b3j0f.aop Documentation

Release 0.7.8

b3j0f

June 14, 2015

1 b3j0f.aop: Aspect Oriented Programming for Python 1
1.1 b3j0f.aop: Aspect Oriented Programming for Python 1
1.2 Changelog 1141
1.3 Indices and tables 1142
1.4 Description 1142
1.5 Links 1142
1.6 Installation 1142
1.7 Features 1142
1.8 Limitations 1143
1.9 Examples 1143
1.10 State of the art 1143
1.11 Perspectives 1145
1.12 Donation 1145

2 Changelog 1147
2.1 0.7.8 (14/06/2015) 1147
2.2 0.7.7 (13/06/2015) 1147
2.3 0.7.6 (13/06/2015) 1147
2.4 0.7.5 (02/06/2015) 1147
2.5 0.7.4 (20/05/2015) 1147

3 Indices and tables 1149

4 Description 1151

5 Links 1153

6 Installation 1155

7 Features 1157

8 Limitations 1159

9 Examples 1161

10 State of the art 1163
10.1 pyaspects 1163
10.2 aspects 1163
10.3 aspect 1164

10.4 spring 1164
10.5 pytilities 1164

11 Perspectives **1165**

12 Donation **1167**

CHAPTER 1

b3j0f.aop: Aspect Oriented Programming for Python

1.1 b3j0f.aop: Aspect Oriented Programming for Python

1.1.1 b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

b3j0f.aop: Aspect Oriented Programming for Python

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
    - less memory consumption in using the __slots__ class property.
    - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
    - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
    - joinpoint matching with function or regex.
    - distributed programming:
        - interception context sharing in order to ease behaviour sharing between advices.
        - uuid for advice identification in order to ease its use in a distributed context.
    - maintenable with well named variables and functions, comments and few lines.
    - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
    - respect of aspects vocabulary in order to ease its use among AOP users.
    - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
    - advices are callable objects.
    - Unit tests for all functions such as examples.
4. Benchmark:
    - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let's get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.

- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
    - less memory consumption in using the __slots__ class property.
    - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
    - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
    - joinpoint matching with function or regex.
    - distributed programming:
        - interception context sharing in order to ease behaviour sharing between advices.
        - uuid for advice identification in order to ease its use in a distributed context.
    - maintenable with well named variables and functions, comments and few lines.
    - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
    - respect of aspects vocabulary in order to ease its use among AOP users.
    - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
    - advices are callable objects.
    - Unit tests for all functions such as examples.
4. Benchmark:
    - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let's get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
   - less memory consumption in using the __slots__ class property.
   - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
   - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
   - joinpoint matching with function or regex.
   - distributed programming:
     - interception context sharing in order to ease behaviour sharing between advices.
     - uuid for advice identification in order to ease its use in a distributed context.
   - maintenable with well named variables and functions, comments and few lines.
   - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
   - respect of aspects vocabulary in order to ease its use among AOP users.
   - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
   - advices are callable objects.
   - Unit tests for all functions such as examples.
4. Benchmark:
   - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let’s get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
    - less memory consumption in using the __slots__ class property.
    - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
    - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
    - joinpoint matching with function or regex.
    - distributed programming:

    - interception context sharing in order to ease behaviour sharing between advices.
    - uuid for advice identification in order to ease its use in a distributed context.
  - maintenable with well named variables and functions, comments and few lines.
  - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
  - respect of aspects vocabulary in order to ease its use among AOP users.
  - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
  - advices are callable objects.
  - Unit tests for all functions such as examples.

4. Benchmark:
   - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let's get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
   - less memory consumption in using the __slots__ class property.
   - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
   - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
   - joinpoint matching with function or regex.
   - distributed programming:
     - interception context sharing in order to ease behaviour sharing between advices.
     - uuid for advice identification in order to ease its use in a distributed context.
   - maintenable with well named variables and functions, comments and few lines.
   - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
   - respect of aspects vocabulary in order to ease its use among AOP users.
   - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
   - advices are callable objects.
   - Unit tests for all functions such as examples.
4. Benchmark:

  - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let’s get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
    - less memory consumption in using the __slots__ class property.
    - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
    - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
    - joinpoint matching with function or regex.
    - distributed programming:
        - interception context sharing in order to ease behaviour sharing between advices.
        - uuid for advice identification in order to ease its use in a distributed context.
    - maintenable with well named variables and functions, comments and few lines.
    - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
    - respect of aspects vocabulary in order to ease its use among AOP users.
    - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
    - advices are callable objects.
    - Unit tests for all functions such as examples.
4. Benchmark:
    - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let's get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
    - less memory consumption in using the __slots__ class property.
    - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
    - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
    - joinpoint matching with function or regex.
    - distributed programming:
        - interception context sharing in order to ease behaviour sharing between advices.
        - uuid for advice identification in order to ease its use in a distributed context.
    - maintenable with well named variables and functions, comments and few lines.
    - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
    - respect of aspects vocabulary in order to ease its use among AOP users.
    - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
    - advices are callable objects.
    - Unit tests for all functions such as examples.
4. Benchmark:
    - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let’s get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
    - less memory consumption in using the __slots__ class property.
    - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
    - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
    - joinpoint matching with function or regex.
    - distributed programming:

      - interception context sharing in order to ease behaviour sharing between advices.
      - uuid for advice identification in order to ease its use in a distributed context.
   - maintenable with well named variables and functions, comments and few lines.
   - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
   - respect of aspects vocabulary in order to ease its use among AOP users.
   - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
   - advices are callable objects.
   - Unit tests for all functions such as examples.

4. Benchmark:
   - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let’s get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
    - less memory consumption in using the __slots__ class property.
    - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
    - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
    - joinpoint matching with function or regex.
    - distributed programming:
        - interception context sharing in order to ease behaviour sharing between advices.
        - uuid for advice identification in order to ease its use in a distributed context.
    - maintenable with well named variables and functions, comments and few lines.
    - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
    - respect of aspects vocabulary in order to ease its use among AOP users.
    - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
    - advices are callable objects.
    - Unit tests for all functions such as examples.
4. Benchmark:

- speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let's get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
    - less memory consumption in using the __slots__ class property.
    - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
    - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
    - joinpoint matching with function or regex.
    - distributed programming:
        - interception context sharing in order to ease behaviour sharing between advices.
        - uuid for advice identification in order to ease its use in a distributed context.
    - maintenable with well named variables and functions, comments and few lines.
    - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
    - respect of aspects vocabulary in order to ease its use among AOP users.
    - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
    - advices are callable objects.
    - Unit tests for all functions such as examples.
4. Benchmark:
    - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let's get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
    - less memory consumption in using the __slots__ class property.
    - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
    - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
    - joinpoint matching with function or regex.
    - distributed programming:
        - interception context sharing in order to ease behaviour sharing between advices.
        - uuid for advice identification in order to ease its use in a distributed context.
    - maintenable with well named variables and functions, comments and few lines.
    - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
    - respect of aspects vocabulary in order to ease its use among AOP users.
    - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
    - advices are callable objects.
    - Unit tests for all functions such as examples.
4. Benchmark:
    - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples

How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let's get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
    - less memory consumption in using the __slots__ class property.
    - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
    - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
    - joinpoint matching with function or regex.
    - distributed programming:

    - interception context sharing in order to ease behaviour sharing between advices.
    - uuid for advice identification in order to ease its use in a distributed context.
- maintenable with well named variables and functions, comments and few lines.
- extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
- respect of aspects vocabulary in order to ease its use among AOP users.
- close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
- advices are callable objects.
- Unit tests for all functions such as examples.

4. Benchmark:
    - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let’s get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
    - less memory consumption in using the __slots__ class property.
    - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
    - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
    - joinpoint matching with function or regex.
    - distributed programming:
        - interception context sharing in order to ease behaviour sharing between advices.
        - uuid for advice identification in order to ease its use in a distributed context.
    - maintenable with well named variables and functions, comments and few lines.
    - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
    - respect of aspects vocabulary in order to ease its use among AOP users.
    - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
    - advices are callable objects.
    - Unit tests for all functions such as examples.
4. Benchmark:

  - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let’s get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
    - less memory consumption in using the __slots__ class property.
    - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
    - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
    - joinpoint matching with function or regex.
    - distributed programming:
        - interception context sharing in order to ease behaviour sharing between advices.
        - uuid for advice identification in order to ease its use in a distributed context.
    - maintenable with well named variables and functions, comments and few lines.
    - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
    - respect of aspects vocabulary in order to ease its use among AOP users.
    - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
    - advices are callable objects.
    - Unit tests for all functions such as examples.
4. Benchmark:
    - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let's get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
    - less memory consumption in using the __slots__ class property.
    - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
    - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
    - joinpoint matching with function or regex.
    - distributed programming:
        - interception context sharing in order to ease behaviour sharing between advices.
        - uuid for advice identification in order to ease its use in a distributed context.
    - maintenable with well named variables and functions, comments and few lines.
    - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
    - respect of aspects vocabulary in order to ease its use among AOP users.
    - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
    - advices are callable objects.
    - Unit tests for all functions such as examples.
4. Benchmark:
    - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let’s get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
   - less memory consumption in using the __slots__ class property.
   - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
   - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
   - joinpoint matching with function or regex.
   - distributed programming:

    - interception context sharing in order to ease behaviour sharing between advices.
    - uuid for advice identification in order to ease its use in a distributed context.
  - maintenable with well named variables and functions, comments and few lines.
  - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
  - respect of aspects vocabulary in order to ease its use among AOP users.
  - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
  - advices are callable objects.
  - Unit tests for all functions such as examples.

4. Benchmark:
  - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let’s get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
    - less memory consumption in using the __slots__ class property.
    - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
    - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
    - joinpoint matching with function or regex.
    - distributed programming:
        - interception context sharing in order to ease behaviour sharing between advices.
        - uuid for advice identification in order to ease its use in a distributed context.
    - maintenable with well named variables and functions, comments and few lines.
    - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
    - respect of aspects vocabulary in order to ease its use among AOP users.
    - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
    - advices are callable objects.
    - Unit tests for all functions such as examples.
4. Benchmark:

- speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let's get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
    - less memory consumption in using the __slots__ class property.
    - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
    - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
    - joinpoint matching with function or regex.
    - distributed programming:
        - interception context sharing in order to ease behaviour sharing between advices.
        - uuid for advice identification in order to ease its use in a distributed context.
    - maintenable with well named variables and functions, comments and few lines.
    - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
    - respect of aspects vocabulary in order to ease its use among AOP users.
    - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
    - advices are callable objects.
    - Unit tests for all functions such as examples.
4. Benchmark:
    - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let's get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
   - less memory consumption in using the __slots__ class property.
   - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
   - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
   - joinpoint matching with function or regex.
   - distributed programming:
     - interception context sharing in order to ease behaviour sharing between advices.
     - uuid for advice identification in order to ease its use in a distributed context.
   - maintenable with well named variables and functions, comments and few lines.
   - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
   - respect of aspects vocabulary in order to ease its use among AOP users.
   - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
   - advices are callable objects.
   - Unit tests for all functions such as examples.
4. Benchmark:
   - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let’s get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
    - less memory consumption in using the __slots__ class property.
    - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
    - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
    - joinpoint matching with function or regex.
    - distributed programming:

        - interception context sharing in order to ease behaviour sharing between advices.
        - uuid for advice identification in order to ease its use in a distributed context.
    - maintenable with well named variables and functions, comments and few lines.
    - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
    - respect of aspects vocabulary in order to ease its use among AOP users.
    - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
    - advices are callable objects.
    - Unit tests for all functions such as examples.

4. Benchmark:
    - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let's get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
   - less memory consumption in using the __slots__ class property.
   - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
   - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
   - joinpoint matching with function or regex.
   - distributed programming:
     - interception context sharing in order to ease behaviour sharing between advices.
     - uuid for advice identification in order to ease its use in a distributed context.
   - maintenable with well named variables and functions, comments and few lines.
   - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
   - respect of aspects vocabulary in order to ease its use among AOP users.
   - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
   - advices are callable objects.
   - Unit tests for all functions such as examples.
4. Benchmark:

- speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let's get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)   # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
    - less memory consumption in using the __slots__ class property.
    - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
    - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
    - joinpoint matching with function or regex.
    - distributed programming:
        - interception context sharing in order to ease behaviour sharing between advices.
        - uuid for advice identification in order to ease its use in a distributed context.
    - maintenable with well named variables and functions, comments and few lines.
    - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
    - respect of aspects vocabulary in order to ease its use among AOP users.
    - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
    - advices are callable objects.
    - Unit tests for all functions such as examples.
4. Benchmark:
    - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let's get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
    - less memory consumption in using the __slots__ class property.
    - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
    - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
    - joinpoint matching with function or regex.
    - distributed programming:
        - interception context sharing in order to ease behaviour sharing between advices.
        - uuid for advice identification in order to ease its use in a distributed context.
    - maintenable with well named variables and functions, comments and few lines.
    - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
    - respect of aspects vocabulary in order to ease its use among AOP users.
    - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
    - advices are callable objects.
    - Unit tests for all functions such as examples.
4. Benchmark:
    - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let’s get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
    - less memory consumption in using the __slots__ class property.
    - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
    - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
    - joinpoint matching with function or regex.
    - distributed programming:

    - interception context sharing in order to ease behaviour sharing between advices.
    - uuid for advice identification in order to ease its use in a distributed context.
  - maintenable with well named variables and functions, comments and few lines.
  - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
  - respect of aspects vocabulary in order to ease its use among AOP users.
  - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
  - advices are callable objects.
  - Unit tests for all functions such as examples.

4. Benchmark:
   - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let’s get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
    - less memory consumption in using the __slots__ class property.
    - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
    - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
    - joinpoint matching with function or regex.
    - distributed programming:
        - interception context sharing in order to ease behaviour sharing between advices.
        - uuid for advice identification in order to ease its use in a distributed context.
    - maintenable with well named variables and functions, comments and few lines.
    - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
    - respect of aspects vocabulary in order to ease its use among AOP users.
    - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
    - advices are callable objects.
    - Unit tests for all functions such as examples.
4. Benchmark:

- speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let's get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
    - less memory consumption in using the __slots__ class property.
    - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
    - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
    - joinpoint matching with function or regex.
    - distributed programming:
        - interception context sharing in order to ease behaviour sharing between advices.
        - uuid for advice identification in order to ease its use in a distributed context.
    - maintenable with well named variables and functions, comments and few lines.
    - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
    - respect of aspects vocabulary in order to ease its use among AOP users.
    - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
    - advices are callable objects.
    - Unit tests for all functions such as examples.
4. Benchmark:
    - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let's get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
    - less memory consumption in using the __slots__ class property.
    - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
    - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
    - joinpoint matching with function or regex.
    - distributed programming:
        - interception context sharing in order to ease behaviour sharing between advices.
        - uuid for advice identification in order to ease its use in a distributed context.
    - maintenable with well named variables and functions, comments and few lines.
    - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
    - respect of aspects vocabulary in order to ease its use among AOP users.
    - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
    - advices are callable objects.
    - Unit tests for all functions such as examples.
4. Benchmark:
    - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let’s get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
    - less memory consumption in using the __slots__ class property.
    - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
    - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
    - joinpoint matching with function or regex.
    - distributed programming:

        - interception context sharing in order to ease behaviour sharing between advices.
        - uuid for advice identification in order to ease its use in a distributed context.
    - maintenable with well named variables and functions, comments and few lines.
    - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
    - respect of aspects vocabulary in order to ease its use among AOP users.
    - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
    - advices are callable objects.
    - Unit tests for all functions such as examples.

4. Benchmark:
    - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples

How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let’s get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
    - less memory consumption in using the __slots__ class property.
    - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
    - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
    - joinpoint matching with function or regex.
    - distributed programming:
        - interception context sharing in order to ease behaviour sharing between advices.
        - uuid for advice identification in order to ease its use in a distributed context.
    - maintenable with well named variables and functions, comments and few lines.
    - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
    - respect of aspects vocabulary in order to ease its use among AOP users.
    - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
    - advices are callable objects.
    - Unit tests for all functions such as examples.
4. Benchmark:

- speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let's get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
   - less memory consumption in using the __slots__ class property.
   - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
   - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
   - joinpoint matching with function or regex.
   - distributed programming:
     - interception context sharing in order to ease behaviour sharing between advices.
     - uuid for advice identification in order to ease its use in a distributed context.
   - maintenable with well named variables and functions, comments and few lines.
   - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
   - respect of aspects vocabulary in order to ease its use among AOP users.
   - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
   - advices are callable objects.
   - Unit tests for all functions such as examples.
4. Benchmark:
   - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let's get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
    - less memory consumption in using the __slots__ class property.
    - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
    - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
    - joinpoint matching with function or regex.
    - distributed programming:
        - interception context sharing in order to ease behaviour sharing between advices.
        - uuid for advice identification in order to ease its use in a distributed context.
    - maintenable with well named variables and functions, comments and few lines.
    - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
    - respect of aspects vocabulary in order to ease its use among AOP users.
    - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
    - advices are callable objects.
    - Unit tests for all functions such as examples.
4. Benchmark:
    - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let's get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
    - less memory consumption in using the __slots__ class property.
    - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
    - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
    - joinpoint matching with function or regex.
    - distributed programming:

    - interception context sharing in order to ease behaviour sharing between advices.
    - uuid for advice identification in order to ease its use in a distributed context.
  - maintenable with well named variables and functions, comments and few lines.
  - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
  - respect of aspects vocabulary in order to ease its use among AOP users.
  - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
  - advices are callable objects.
  - Unit tests for all functions such as examples.

4. Benchmark:
   - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let’s get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
   - less memory consumption in using the __slots__ class property.
   - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
   - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
   - joinpoint matching with function or regex.
   - distributed programming:
     - interception context sharing in order to ease behaviour sharing between advices.
     - uuid for advice identification in order to ease its use in a distributed context.
   - maintenable with well named variables and functions, comments and few lines.
   - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
   - respect of aspects vocabulary in order to ease its use among AOP users.
   - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
   - advices are callable objects.
   - Unit tests for all functions such as examples.
4. Benchmark:

  - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let’s get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
    - less memory consumption in using the __slots__ class property.
    - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
    - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
    - joinpoint matching with function or regex.
    - distributed programming:
        - interception context sharing in order to ease behaviour sharing between advices.
        - uuid for advice identification in order to ease its use in a distributed context.
    - maintenable with well named variables and functions, comments and few lines.
    - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
    - respect of aspects vocabulary in order to ease its use among AOP users.
    - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
    - advices are callable objects.
    - Unit tests for all functions such as examples.
4. Benchmark:
    - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let's get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
    - less memory consumption in using the __slots__ class property.
    - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
    - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
    - joinpoint matching with function or regex.
    - distributed programming:
        - interception context sharing in order to ease behaviour sharing between advices.
        - uuid for advice identification in order to ease its use in a distributed context.
    - maintenable with well named variables and functions, comments and few lines.
    - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
    - respect of aspects vocabulary in order to ease its use among AOP users.
    - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
    - advices are callable objects.
    - Unit tests for all functions such as examples.
4. Benchmark:
    - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let's get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
   - less memory consumption in using the __slots__ class property.
   - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
   - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
   - joinpoint matching with function or regex.
   - distributed programming:

    - interception context sharing in order to ease behaviour sharing between advices.
    - uuid for advice identification in order to ease its use in a distributed context.
  - maintenable with well named variables and functions, comments and few lines.
  - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
  - respect of aspects vocabulary in order to ease its use among AOP users.
  - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
  - advices are callable objects.
  - Unit tests for all functions such as examples.

4. Benchmark:
   - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let's get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
    - less memory consumption in using the __slots__ class property.
    - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
    - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
    - joinpoint matching with function or regex.
    - distributed programming:
        - interception context sharing in order to ease behaviour sharing between advices.
        - uuid for advice identification in order to ease its use in a distributed context.
    - maintenable with well named variables and functions, comments and few lines.
    - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
    - respect of aspects vocabulary in order to ease its use among AOP users.
    - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
    - advices are callable objects.
    - Unit tests for all functions such as examples.
4. Benchmark:

    - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let's get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
   - less memory consumption in using the __slots__ class property.
   - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
   - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
   - joinpoint matching with function or regex.
   - distributed programming:
     - interception context sharing in order to ease behaviour sharing between advices.
     - uuid for advice identification in order to ease its use in a distributed context.
   - maintenable with well named variables and functions, comments and few lines.
   - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
   - respect of aspects vocabulary in order to ease its use among AOP users.
   - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
   - advices are callable objects.
   - Unit tests for all functions such as examples.
4. Benchmark:
   - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let's get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
    - less memory consumption in using the __slots__ class property.
    - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
    - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
    - joinpoint matching with function or regex.
    - distributed programming:
        - interception context sharing in order to ease behaviour sharing between advices.
        - uuid for advice identification in order to ease its use in a distributed context.
    - maintenable with well named variables and functions, comments and few lines.
    - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
    - respect of aspects vocabulary in order to ease its use among AOP users.
    - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
    - advices are callable objects.
    - Unit tests for all functions such as examples.
4. Benchmark:
    - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let’s get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
    - less memory consumption in using the __slots__ class property.
    - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
    - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
    - joinpoint matching with function or regex.
    - distributed programming:

    - interception context sharing in order to ease behaviour sharing between advices.
    - uuid for advice identification in order to ease its use in a distributed context.
  - maintenable with well named variables and functions, comments and few lines.
  - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
  - respect of aspects vocabulary in order to ease its use among AOP users.
  - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
  - advices are callable objects.
  - Unit tests for all functions such as examples.

4. Benchmark:
   - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let's get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
    - less memory consumption in using the __slots__ class property.
    - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
    - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
    - joinpoint matching with function or regex.
    - distributed programming:
        - interception context sharing in order to ease behaviour sharing between advices.
        - uuid for advice identification in order to ease its use in a distributed context.
    - maintenable with well named variables and functions, comments and few lines.
    - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
    - respect of aspects vocabulary in order to ease its use among AOP users.
    - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
    - advices are callable objects.
    - Unit tests for all functions such as examples.
4. Benchmark:

- speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let's get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
    - less memory consumption in using the __slots__ class property.
    - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
    - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
    - joinpoint matching with function or regex.
    - distributed programming:
        - interception context sharing in order to ease behaviour sharing between advices.
        - uuid for advice identification in order to ease its use in a distributed context.
    - maintenable with well named variables and functions, comments and few lines.
    - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
    - respect of aspects vocabulary in order to ease its use among AOP users.
    - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
    - advices are callable objects.
    - Unit tests for all functions such as examples.
4. Benchmark:
    - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let's get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
    - less memory consumption in using the __slots__ class property.
    - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
    - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
    - joinpoint matching with function or regex.
    - distributed programming:
        - interception context sharing in order to ease behaviour sharing between advices.
        - uuid for advice identification in order to ease its use in a distributed context.
    - maintenable with well named variables and functions, comments and few lines.
    - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
    - respect of aspects vocabulary in order to ease its use among AOP users.
    - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
    - advices are callable objects.
    - Unit tests for all functions such as examples.
4. Benchmark:
    - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let's get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
   - less memory consumption in using the __slots__ class property.
   - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
   - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
   - joinpoint matching with function or regex.
   - distributed programming:

    - interception context sharing in order to ease behaviour sharing between advices.
    - uuid for advice identification in order to ease its use in a distributed context.
  - maintenable with well named variables and functions, comments and few lines.
  - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
  - respect of aspects vocabulary in order to ease its use among AOP users.
  - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
  - advices are callable objects.
  - Unit tests for all functions such as examples.

4. Benchmark:
   - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let’s get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
    - less memory consumption in using the __slots__ class property.
    - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
    - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
    - joinpoint matching with function or regex.
    - distributed programming:
        - interception context sharing in order to ease behaviour sharing between advices.
        - uuid for advice identification in order to ease its use in a distributed context.
    - maintenable with well named variables and functions, comments and few lines.
    - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
    - respect of aspects vocabulary in order to ease its use among AOP users.
    - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
    - advices are callable objects.
    - Unit tests for all functions such as examples.
4. Benchmark:

  - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let’s get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)   # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
    - less memory consumption in using the __slots__ class property.
    - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
    - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
    - joinpoint matching with function or regex.
    - distributed programming:
        - interception context sharing in order to ease behaviour sharing between advices.
        - uuid for advice identification in order to ease its use in a distributed context.
    - maintenable with well named variables and functions, comments and few lines.
    - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
    - respect of aspects vocabulary in order to ease its use among AOP users.
    - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
    - advices are callable objects.
    - Unit tests for all functions such as examples.
4. Benchmark:
    - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let's get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
    - less memory consumption in using the __slots__ class property.
    - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
    - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
    - joinpoint matching with function or regex.
    - distributed programming:
        - interception context sharing in order to ease behaviour sharing between advices.
        - uuid for advice identification in order to ease its use in a distributed context.
    - maintenable with well named variables and functions, comments and few lines.
    - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
    - respect of aspects vocabulary in order to ease its use among AOP users.
    - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
    - advices are callable objects.
    - Unit tests for all functions such as examples.
4. Benchmark:
    - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let’s get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
    - less memory consumption in using the __slots__ class property.
    - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
    - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
    - joinpoint matching with function or regex.
    - distributed programming:

    - interception context sharing in order to ease behaviour sharing between advices.
    - uuid for advice identification in order to ease its use in a distributed context.
  - maintenable with well named variables and functions, comments and few lines.
  - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
  - respect of aspects vocabulary in order to ease its use among AOP users.
  - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
  - advices are callable objects.
  - Unit tests for all functions such as examples.

4. Benchmark:
   - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let’s get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
   - less memory consumption in using the __slots__ class property.
   - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
   - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
   - joinpoint matching with function or regex.
   - distributed programming:
     - interception context sharing in order to ease behaviour sharing between advices.
     - uuid for advice identification in order to ease its use in a distributed context.
   - maintenable with well named variables and functions, comments and few lines.
   - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
   - respect of aspects vocabulary in order to ease its use among AOP users.
   - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
   - advices are callable objects.
   - Unit tests for all functions such as examples.
4. Benchmark:

- speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let's get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)   # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
   - less memory consumption in using the __slots__ class property.
   - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
   - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
   - joinpoint matching with function or regex.
   - distributed programming:
     - interception context sharing in order to ease behaviour sharing between advices.
     - uuid for advice identification in order to ease its use in a distributed context.
   - maintenable with well named variables and functions, comments and few lines.
   - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
   - respect of aspects vocabulary in order to ease its use among AOP users.
   - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
   - advices are callable objects.
   - Unit tests for all functions such as examples.
4. Benchmark:
   - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let's get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
    - less memory consumption in using the __slots__ class property.
    - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
    - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
    - joinpoint matching with function or regex.
    - distributed programming:
        - interception context sharing in order to ease behaviour sharing between advices.
        - uuid for advice identification in order to ease its use in a distributed context.
    - maintenable with well named variables and functions, comments and few lines.
    - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
    - respect of aspects vocabulary in order to ease its use among AOP users.
    - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
    - advices are callable objects.
    - Unit tests for all functions such as examples.
4. Benchmark:
    - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let’s get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
   - less memory consumption in using the __slots__ class property.
   - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
   - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
   - joinpoint matching with function or regex.
   - distributed programming:

    - interception context sharing in order to ease behaviour sharing between advices.
    - uuid for advice identification in order to ease its use in a distributed context.
  - maintenable with well named variables and functions, comments and few lines.
  - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
  - respect of aspects vocabulary in order to ease its use among AOP users.
  - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
  - advices are callable objects.
  - Unit tests for all functions such as examples.

4. Benchmark:
  - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let’s get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
    - less memory consumption in using the __slots__ class property.
    - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
    - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
    - joinpoint matching with function or regex.
    - distributed programming:
        - interception context sharing in order to ease behaviour sharing between advices.
        - uuid for advice identification in order to ease its use in a distributed context.
    - maintenable with well named variables and functions, comments and few lines.
    - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
    - respect of aspects vocabulary in order to ease its use among AOP users.
    - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
    - advices are callable objects.
    - Unit tests for all functions such as examples.
4. Benchmark:

- speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let's get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
    - less memory consumption in using the __slots__ class property.
    - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
    - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
    - joinpoint matching with function or regex.
    - distributed programming:
        - interception context sharing in order to ease behaviour sharing between advices.
        - uuid for advice identification in order to ease its use in a distributed context.
    - maintenable with well named variables and functions, comments and few lines.
    - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
    - respect of aspects vocabulary in order to ease its use among AOP users.
    - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
    - advices are callable objects.
    - Unit tests for all functions such as examples.
4. Benchmark:
    - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let's get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
   - less memory consumption in using the __slots__ class property.
   - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
   - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
   - joinpoint matching with function or regex.
   - distributed programming:
     - interception context sharing in order to ease behaviour sharing between advices.
     - uuid for advice identification in order to ease its use in a distributed context.
   - maintenable with well named variables and functions, comments and few lines.
   - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
   - respect of aspects vocabulary in order to ease its use among AOP users.
   - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
   - advices are callable objects.
   - Unit tests for all functions such as examples.
4. Benchmark:
   - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let's get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
    - less memory consumption in using the __slots__ class property.
    - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
    - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
    - joinpoint matching with function or regex.
    - distributed programming:

  - interception context sharing in order to ease behaviour sharing between advices.
  - uuid for advice identification in order to ease its use in a distributed context.
- maintenable with well named variables and functions, comments and few lines.
- extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
- respect of aspects vocabulary in order to ease its use among AOP users.
- close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
- advices are callable objects.
- Unit tests for all functions such as examples.

4. Benchmark:
   - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let’s get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
    - less memory consumption in using the __slots__ class property.
    - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
    - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
    - joinpoint matching with function or regex.
    - distributed programming:
        - interception context sharing in order to ease behaviour sharing between advices.
        - uuid for advice identification in order to ease its use in a distributed context.
    - maintenable with well named variables and functions, comments and few lines.
    - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
    - respect of aspects vocabulary in order to ease its use among AOP users.
    - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
    - advices are callable objects.
    - Unit tests for all functions such as examples.
4. Benchmark:

  - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let’s get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)   # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
    - less memory consumption in using the __slots__ class property.
    - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
    - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
    - joinpoint matching with function or regex.
    - distributed programming:
        - interception context sharing in order to ease behaviour sharing between advices.
        - uuid for advice identification in order to ease its use in a distributed context.
    - maintenable with well named variables and functions, comments and few lines.
    - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
    - respect of aspects vocabulary in order to ease its use among AOP users.
    - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
    - advices are callable objects.
    - Unit tests for all functions such as examples.
4. Benchmark:
    - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let's get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
   - less memory consumption in using the __slots__ class property.
   - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
   - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
   - joinpoint matching with function or regex.
   - distributed programming:
     - interception context sharing in order to ease behaviour sharing between advices.
     - uuid for advice identification in order to ease its use in a distributed context.
   - maintenable with well named variables and functions, comments and few lines.
   - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
   - respect of aspects vocabulary in order to ease its use among AOP users.
   - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
   - advices are callable objects.
   - Unit tests for all functions such as examples.
4. Benchmark:
   - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let’s get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
   - less memory consumption in using the __slots__ class property.
   - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
   - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
   - joinpoint matching with function or regex.
   - distributed programming:

    - interception context sharing in order to ease behaviour sharing between advices.
    - uuid for advice identification in order to ease its use in a distributed context.
  - maintenable with well named variables and functions, comments and few lines.
  - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
  - respect of aspects vocabulary in order to ease its use among AOP users.
  - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
  - advices are callable objects.
  - Unit tests for all functions such as examples.

4. Benchmark:
   - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let’s get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
    - less memory consumption in using the __slots__ class property.
    - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
    - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
    - joinpoint matching with function or regex.
    - distributed programming:
        - interception context sharing in order to ease behaviour sharing between advices.
        - uuid for advice identification in order to ease its use in a distributed context.
    - maintenable with well named variables and functions, comments and few lines.
    - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
    - respect of aspects vocabulary in order to ease its use among AOP users.
    - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
    - advices are callable objects.
    - Unit tests for all functions such as examples.
4. Benchmark:

- speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let's get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
    - less memory consumption in using the __slots__ class property.
    - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
    - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
    - joinpoint matching with function or regex.
    - distributed programming:
        - interception context sharing in order to ease behaviour sharing between advices.
        - uuid for advice identification in order to ease its use in a distributed context.
    - maintenable with well named variables and functions, comments and few lines.
    - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
    - respect of aspects vocabulary in order to ease its use among AOP users.
    - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
    - advices are callable objects.
    - Unit tests for all functions such as examples.
4. Benchmark:
    - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let's get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
    - less memory consumption in using the __slots__ class property.
    - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
    - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
    - joinpoint matching with function or regex.
    - distributed programming:
        - interception context sharing in order to ease behaviour sharing between advices.
        - uuid for advice identification in order to ease its use in a distributed context.
    - maintenable with well named variables and functions, comments and few lines.
    - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
    - respect of aspects vocabulary in order to ease its use among AOP users.
    - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
    - advices are callable objects.
    - Unit tests for all functions such as examples.
4. Benchmark:
    - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let’s get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
   - less memory consumption in using the __slots__ class property.
   - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
   - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
   - joinpoint matching with function or regex.
   - distributed programming:

  - interception context sharing in order to ease behaviour sharing between advices.
  - uuid for advice identification in order to ease its use in a distributed context.
- maintenable with well named variables and functions, comments and few lines.
- extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
- respect of aspects vocabulary in order to ease its use among AOP users.
- close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
- advices are callable objects.
- Unit tests for all functions such as examples.

4. Benchmark:
   - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples

How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let's get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
    - less memory consumption in using the __slots__ class property.
    - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
    - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
    - joinpoint matching with function or regex.
    - distributed programming:
        - interception context sharing in order to ease behaviour sharing between advices.
        - uuid for advice identification in order to ease its use in a distributed context.
    - maintenable with well named variables and functions, comments and few lines.
    - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
    - respect of aspects vocabulary in order to ease its use among AOP users.
    - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
    - advices are callable objects.
    - Unit tests for all functions such as examples.
4. Benchmark:

  - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let’s get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
    - less memory consumption in using the __slots__ class property.
    - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
    - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
    - joinpoint matching with function or regex.
    - distributed programming:
        - interception context sharing in order to ease behaviour sharing between advices.
        - uuid for advice identification in order to ease its use in a distributed context.
    - maintenable with well named variables and functions, comments and few lines.
    - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
    - respect of aspects vocabulary in order to ease its use among AOP users.
    - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
    - advices are callable objects.
    - Unit tests for all functions such as examples.
4. Benchmark:
    - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let's get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
   - less memory consumption in using the __slots__ class property.
   - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
   - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
   - joinpoint matching with function or regex.
   - distributed programming:
     - interception context sharing in order to ease behaviour sharing between advices.
     - uuid for advice identification in order to ease its use in a distributed context.
   - maintenable with well named variables and functions, comments and few lines.
   - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
   - respect of aspects vocabulary in order to ease its use among AOP users.
   - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
   - advices are callable objects.
   - Unit tests for all functions such as examples.
4. Benchmark:
   - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let's get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
    - less memory consumption in using the __slots__ class property.
    - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
    - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
    - joinpoint matching with function or regex.
    - distributed programming:

  - interception context sharing in order to ease behaviour sharing between advices.
  - uuid for advice identification in order to ease its use in a distributed context.
- maintenable with well named variables and functions, comments and few lines.
- extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
- respect of aspects vocabulary in order to ease its use among AOP users.
- close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
- advices are callable objects.
- Unit tests for all functions such as examples.

4. Benchmark:
   - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let’s get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
    - less memory consumption in using the __slots__ class property.
    - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
    - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
    - joinpoint matching with function or regex.
    - distributed programming:
        - interception context sharing in order to ease behaviour sharing between advices.
        - uuid for advice identification in order to ease its use in a distributed context.
    - maintenable with well named variables and functions, comments and few lines.
    - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
    - respect of aspects vocabulary in order to ease its use among AOP users.
    - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
    - advices are callable objects.
    - Unit tests for all functions such as examples.
4. Benchmark:

- speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let’s get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
    - less memory consumption in using the __slots__ class property.
    - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
    - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
    - joinpoint matching with function or regex.
    - distributed programming:
        - interception context sharing in order to ease behaviour sharing between advices.
        - uuid for advice identification in order to ease its use in a distributed context.
    - maintenable with well named variables and functions, comments and few lines.
    - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
    - respect of aspects vocabulary in order to ease its use among AOP users.
    - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
    - advices are callable objects.
    - Unit tests for all functions such as examples.
4. Benchmark:
    - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let's get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
   - less memory consumption in using the __slots__ class property.
   - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
   - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
   - joinpoint matching with function or regex.
   - distributed programming:
     - interception context sharing in order to ease behaviour sharing between advices.
     - uuid for advice identification in order to ease its use in a distributed context.
   - maintenable with well named variables and functions, comments and few lines.
   - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
   - respect of aspects vocabulary in order to ease its use among AOP users.
   - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
   - advices are callable objects.
   - Unit tests for all functions such as examples.
4. Benchmark:
   - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let’s get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
    - less memory consumption in using the __slots__ class property.
    - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
    - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
    - joinpoint matching with function or regex.
    - distributed programming:

    - interception context sharing in order to ease behaviour sharing between advices.
    - uuid for advice identification in order to ease its use in a distributed context.
  - maintenable with well named variables and functions, comments and few lines.
  - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
  - respect of aspects vocabulary in order to ease its use among AOP users.
  - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
  - advices are callable objects.
  - Unit tests for all functions such as examples.

4. Benchmark:
  - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let's get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
   - less memory consumption in using the __slots__ class property.
   - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
   - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
   - joinpoint matching with function or regex.
   - distributed programming:
     - interception context sharing in order to ease behaviour sharing between advices.
     - uuid for advice identification in order to ease its use in a distributed context.
   - maintenable with well named variables and functions, comments and few lines.
   - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
   - respect of aspects vocabulary in order to ease its use among AOP users.
   - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
   - advices are callable objects.
   - Unit tests for all functions such as examples.
4. Benchmark:

- speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let’s get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
    - less memory consumption in using the __slots__ class property.
    - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
    - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
    - joinpoint matching with function or regex.
    - distributed programming:
        - interception context sharing in order to ease behaviour sharing between advices.
        - uuid for advice identification in order to ease its use in a distributed context.
    - maintenable with well named variables and functions, comments and few lines.
    - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
    - respect of aspects vocabulary in order to ease its use among AOP users.
    - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
    - advices are callable objects.
    - Unit tests for all functions such as examples.
4. Benchmark:
    - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let's get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
   - less memory consumption in using the __slots__ class property.
   - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
   - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
   - joinpoint matching with function or regex.
   - distributed programming:
     - interception context sharing in order to ease behaviour sharing between advices.
     - uuid for advice identification in order to ease its use in a distributed context.
   - maintenable with well named variables and functions, comments and few lines.
   - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
   - respect of aspects vocabulary in order to ease its use among AOP users.
   - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
   - advices are callable objects.
   - Unit tests for all functions such as examples.
4. Benchmark:
   - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let’s get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
    - less memory consumption in using the __slots__ class property.
    - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
    - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
    - joinpoint matching with function or regex.
    - distributed programming:

    - interception context sharing in order to ease behaviour sharing between advices.
    - uuid for advice identification in order to ease its use in a distributed context.
  - maintenable with well named variables and functions, comments and few lines.
  - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
  - respect of aspects vocabulary in order to ease its use among AOP users.
  - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
  - advices are callable objects.
  - Unit tests for all functions such as examples.

4. Benchmark:
   - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let’s get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
   - less memory consumption in using the __slots__ class property.
   - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
   - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
   - joinpoint matching with function or regex.
   - distributed programming:
     - interception context sharing in order to ease behaviour sharing between advices.
     - uuid for advice identification in order to ease its use in a distributed context.
   - maintenable with well named variables and functions, comments and few lines.
   - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
   - respect of aspects vocabulary in order to ease its use among AOP users.
   - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
   - advices are callable objects.
   - Unit tests for all functions such as examples.
4. Benchmark:

- speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let's get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)   # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
    - less memory consumption in using the __slots__ class property.
    - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
    - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
    - joinpoint matching with function or regex.
    - distributed programming:
        - interception context sharing in order to ease behaviour sharing between advices.
        - uuid for advice identification in order to ease its use in a distributed context.
    - maintenable with well named variables and functions, comments and few lines.
    - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
    - respect of aspects vocabulary in order to ease its use among AOP users.
    - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
    - advices are callable objects.
    - Unit tests for all functions such as examples.
4. Benchmark:
    - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let's get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
   - less memory consumption in using the __slots__ class property.
   - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
   - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
   - joinpoint matching with function or regex.
   - distributed programming:
     - interception context sharing in order to ease behaviour sharing between advices.
     - uuid for advice identification in order to ease its use in a distributed context.
   - maintenable with well named variables and functions, comments and few lines.
   - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
   - respect of aspects vocabulary in order to ease its use among AOP users.
   - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
   - advices are callable objects.
   - Unit tests for all functions such as examples.
4. Benchmark:
   - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let’s get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
   - less memory consumption in using the __slots__ class property.
   - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
   - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
   - joinpoint matching with function or regex.
   - distributed programming:

    - interception context sharing in order to ease behaviour sharing between advices.
    - uuid for advice identification in order to ease its use in a distributed context.
  - maintenable with well named variables and functions, comments and few lines.
  - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
  - respect of aspects vocabulary in order to ease its use among AOP users.
  - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
  - advices are callable objects.
  - Unit tests for all functions such as examples.

4. Benchmark:
   - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let's get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
    - less memory consumption in using the __slots__ class property.
    - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
    - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
    - joinpoint matching with function or regex.
    - distributed programming:
        - interception context sharing in order to ease behaviour sharing between advices.
        - uuid for advice identification in order to ease its use in a distributed context.
    - maintenable with well named variables and functions, comments and few lines.
    - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
    - respect of aspects vocabulary in order to ease its use among AOP users.
    - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
    - advices are callable objects.
    - Unit tests for all functions such as examples.
4. Benchmark:

- speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let's get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
   - less memory consumption in using the __slots__ class property.
   - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
   - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
   - joinpoint matching with function or regex.
   - distributed programming:
     - interception context sharing in order to ease behaviour sharing between advices.
     - uuid for advice identification in order to ease its use in a distributed context.
   - maintenable with well named variables and functions, comments and few lines.
   - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
   - respect of aspects vocabulary in order to ease its use among AOP users.
   - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
   - advices are callable objects.
   - Unit tests for all functions such as examples.
4. Benchmark:
   - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let's get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
    - less memory consumption in using the __slots__ class property.
    - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
    - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
    - joinpoint matching with function or regex.
    - distributed programming:
        - interception context sharing in order to ease behaviour sharing between advices.
        - uuid for advice identification in order to ease its use in a distributed context.
    - maintenable with well named variables and functions, comments and few lines.
    - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
    - respect of aspects vocabulary in order to ease its use among AOP users.
    - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
    - advices are callable objects.
    - Unit tests for all functions such as examples.
4. Benchmark:
    - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let’s get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
   - less memory consumption in using the __slots__ class property.
   - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
   - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
   - joinpoint matching with function or regex.
   - distributed programming:

    - interception context sharing in order to ease behaviour sharing between advices.
    - uuid for advice identification in order to ease its use in a distributed context.
  - maintenable with well named variables and functions, comments and few lines.
  - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
  - respect of aspects vocabulary in order to ease its use among AOP users.
  - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
  - advices are callable objects.
  - Unit tests for all functions such as examples.

4. Benchmark:
  - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let’s get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
   - less memory consumption in using the __slots__ class property.
   - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
   - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
   - joinpoint matching with function or regex.
   - distributed programming:
     - interception context sharing in order to ease behaviour sharing between advices.
     - uuid for advice identification in order to ease its use in a distributed context.
   - maintenable with well named variables and functions, comments and few lines.
   - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
   - respect of aspects vocabulary in order to ease its use among AOP users.
   - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
   - advices are callable objects.
   - Unit tests for all functions such as examples.
4. Benchmark:

- speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let's get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)   # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
    - less memory consumption in using the __slots__ class property.
    - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
    - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
    - joinpoint matching with function or regex.
    - distributed programming:
        - interception context sharing in order to ease behaviour sharing between advices.
        - uuid for advice identification in order to ease its use in a distributed context.
    - maintenable with well named variables and functions, comments and few lines.
    - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
    - respect of aspects vocabulary in order to ease its use among AOP users.
    - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
    - advices are callable objects.
    - Unit tests for all functions such as examples.
4. Benchmark:
    - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let's get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
    - less memory consumption in using the __slots__ class property.
    - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
    - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
    - joinpoint matching with function or regex.
    - distributed programming:
        - interception context sharing in order to ease behaviour sharing between advices.
        - uuid for advice identification in order to ease its use in a distributed context.
    - maintenable with well named variables and functions, comments and few lines.
    - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
    - respect of aspects vocabulary in order to ease its use among AOP users.
    - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
    - advices are callable objects.
    - Unit tests for all functions such as examples.
4. Benchmark:
    - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let’s get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
    - less memory consumption in using the __slots__ class property.
    - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
    - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
    - joinpoint matching with function or regex.
    - distributed programming:

    - interception context sharing in order to ease behaviour sharing between advices.
    - uuid for advice identification in order to ease its use in a distributed context.
  - maintenable with well named variables and functions, comments and few lines.
  - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
  - respect of aspects vocabulary in order to ease its use among AOP users.
  - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
  - advices are callable objects.
  - Unit tests for all functions such as examples.

4. Benchmark:
  - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let’s get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
   - less memory consumption in using the __slots__ class property.
   - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
   - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
   - joinpoint matching with function or regex.
   - distributed programming:
     - interception context sharing in order to ease behaviour sharing between advices.
     - uuid for advice identification in order to ease its use in a distributed context.
   - maintenable with well named variables and functions, comments and few lines.
   - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
   - respect of aspects vocabulary in order to ease its use among AOP users.
   - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
   - advices are callable objects.
   - Unit tests for all functions such as examples.
4. Benchmark:

- speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let's get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)   # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
    - less memory consumption in using the __slots__ class property.
    - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
    - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
    - joinpoint matching with function or regex.
    - distributed programming:
        - interception context sharing in order to ease behaviour sharing between advices.
        - uuid for advice identification in order to ease its use in a distributed context.
    - maintenable with well named variables and functions, comments and few lines.
    - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
    - respect of aspects vocabulary in order to ease its use among AOP users.
    - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
    - advices are callable objects.
    - Unit tests for all functions such as examples.
4. Benchmark:
    - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let's get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
   - less memory consumption in using the __slots__ class property.
   - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
   - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
   - joinpoint matching with function or regex.
   - distributed programming:
     - interception context sharing in order to ease behaviour sharing between advices.
     - uuid for advice identification in order to ease its use in a distributed context.
   - maintenable with well named variables and functions, comments and few lines.
   - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
   - respect of aspects vocabulary in order to ease its use among AOP users.
   - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
   - advices are callable objects.
   - Unit tests for all functions such as examples.
4. Benchmark:
   - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let’s get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
    - less memory consumption in using the __slots__ class property.
    - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
    - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
    - joinpoint matching with function or regex.
    - distributed programming:

    - interception context sharing in order to ease behaviour sharing between advices.
    - uuid for advice identification in order to ease its use in a distributed context.
  - maintenable with well named variables and functions, comments and few lines.
  - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
  - respect of aspects vocabulary in order to ease its use among AOP users.
  - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
  - advices are callable objects.
  - Unit tests for all functions such as examples.

4. Benchmark:
   - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples

How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let’s get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
    - less memory consumption in using the __slots__ class property.
    - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
    - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
    - joinpoint matching with function or regex.
    - distributed programming:
        - interception context sharing in order to ease behaviour sharing between advices.
        - uuid for advice identification in order to ease its use in a distributed context.
    - maintenable with well named variables and functions, comments and few lines.
    - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
    - respect of aspects vocabulary in order to ease its use among AOP users.
    - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
    - advices are callable objects.
    - Unit tests for all functions such as examples.
4. Benchmark:

- speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let's get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)   # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
   - less memory consumption in using the __slots__ class property.
   - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
   - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
   - joinpoint matching with function or regex.
   - distributed programming:
     - interception context sharing in order to ease behaviour sharing between advices.
     - uuid for advice identification in order to ease its use in a distributed context.
   - maintenable with well named variables and functions, comments and few lines.
   - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
   - respect of aspects vocabulary in order to ease its use among AOP users.
   - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
   - advices are callable objects.
   - Unit tests for all functions such as examples.
4. Benchmark:
   - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let's get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
   - less memory consumption in using the __slots__ class property.
   - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
   - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
   - joinpoint matching with function or regex.
   - distributed programming:
     - interception context sharing in order to ease behaviour sharing between advices.
     - uuid for advice identification in order to ease its use in a distributed context.
   - maintenable with well named variables and functions, comments and few lines.
   - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
   - respect of aspects vocabulary in order to ease its use among AOP users.
   - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
   - advices are callable objects.
   - Unit tests for all functions such as examples.
4. Benchmark:
   - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let’s get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
   - less memory consumption in using the __slots__ class property.
   - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
   - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
   - joinpoint matching with function or regex.
   - distributed programming:

    - interception context sharing in order to ease behaviour sharing between advices.
    - uuid for advice identification in order to ease its use in a distributed context.
  - maintenable with well named variables and functions, comments and few lines.
  - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
  - respect of aspects vocabulary in order to ease its use among AOP users.
  - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
  - advices are callable objects.
  - Unit tests for all functions such as examples.

4. Benchmark:
   - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples

How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let’s get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
   - less memory consumption in using the __slots__ class property.
   - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
   - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
   - joinpoint matching with function or regex.
   - distributed programming:
     - interception context sharing in order to ease behaviour sharing between advices.
     - uuid for advice identification in order to ease its use in a distributed context.
   - maintenable with well named variables and functions, comments and few lines.
   - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
   - respect of aspects vocabulary in order to ease its use among AOP users.
   - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
   - advices are callable objects.
   - Unit tests for all functions such as examples.
4. Benchmark:

- speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let's get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)   # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
    - less memory consumption in using the __slots__ class property.
    - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
    - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
    - joinpoint matching with function or regex.
    - distributed programming:
        - interception context sharing in order to ease behaviour sharing between advices.
        - uuid for advice identification in order to ease its use in a distributed context.
    - maintenable with well named variables and functions, comments and few lines.
    - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
    - respect of aspects vocabulary in order to ease its use among AOP users.
    - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
    - advices are callable objects.
    - Unit tests for all functions such as examples.
4. Benchmark:
    - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let's get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
    - less memory consumption in using the __slots__ class property.
    - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
    - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
    - joinpoint matching with function or regex.
    - distributed programming:
        - interception context sharing in order to ease behaviour sharing between advices.
        - uuid for advice identification in order to ease its use in a distributed context.
    - maintenable with well named variables and functions, comments and few lines.
    - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
    - respect of aspects vocabulary in order to ease its use among AOP users.
    - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
    - advices are callable objects.
    - Unit tests for all functions such as examples.
4. Benchmark:
    - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let’s get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
    - less memory consumption in using the __slots__ class property.
    - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
    - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
    - joinpoint matching with function or regex.
    - distributed programming:

    - interception context sharing in order to ease behaviour sharing between advices.
    - uuid for advice identification in order to ease its use in a distributed context.
  - maintenable with well named variables and functions, comments and few lines.
  - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
  - respect of aspects vocabulary in order to ease its use among AOP users.
  - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
  - advices are callable objects.
  - Unit tests for all functions such as examples.

4. Benchmark:
  - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples

How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let’s get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
    - less memory consumption in using the __slots__ class property.
    - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
    - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
    - joinpoint matching with function or regex.
    - distributed programming:
        - interception context sharing in order to ease behaviour sharing between advices.
        - uuid for advice identification in order to ease its use in a distributed context.
    - maintenable with well named variables and functions, comments and few lines.
    - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
    - respect of aspects vocabulary in order to ease its use among AOP users.
    - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
    - advices are callable objects.
    - Unit tests for all functions such as examples.
4. Benchmark:

- speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let's get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)   # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
    - less memory consumption in using the __slots__ class property.
    - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
    - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
    - joinpoint matching with function or regex.
    - distributed programming:
        - interception context sharing in order to ease behaviour sharing between advices.
        - uuid for advice identification in order to ease its use in a distributed context.
    - maintenable with well named variables and functions, comments and few lines.
    - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
    - respect of aspects vocabulary in order to ease its use among AOP users.
    - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
    - advices are callable objects.
    - Unit tests for all functions such as examples.
4. Benchmark:
    - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let's get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
   - less memory consumption in using the __slots__ class property.
   - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
   - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
   - joinpoint matching with function or regex.
   - distributed programming:
     - interception context sharing in order to ease behaviour sharing between advices.
     - uuid for advice identification in order to ease its use in a distributed context.
   - maintenable with well named variables and functions, comments and few lines.
   - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
   - respect of aspects vocabulary in order to ease its use among AOP users.
   - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
   - advices are callable objects.
   - Unit tests for all functions such as examples.
4. Benchmark:
   - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let’s get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
    - less memory consumption in using the __slots__ class property.
    - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
    - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
    - joinpoint matching with function or regex.
    - distributed programming:

    - interception context sharing in order to ease behaviour sharing between advices.
    - uuid for advice identification in order to ease its use in a distributed context.
  - maintenable with well named variables and functions, comments and few lines.
  - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
  - respect of aspects vocabulary in order to ease its use among AOP users.
  - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
  - advices are callable objects.
  - Unit tests for all functions such as examples.

4. Benchmark:
   - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let’s get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
   - less memory consumption in using the __slots__ class property.
   - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
   - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
   - joinpoint matching with function or regex.
   - distributed programming:
     - interception context sharing in order to ease behaviour sharing between advices.
     - uuid for advice identification in order to ease its use in a distributed context.
   - maintenable with well named variables and functions, comments and few lines.
   - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
   - respect of aspects vocabulary in order to ease its use among AOP users.
   - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
   - advices are callable objects.
   - Unit tests for all functions such as examples.
4. Benchmark:

- speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let's get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
    - less memory consumption in using the __slots__ class property.
    - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
    - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
    - joinpoint matching with function or regex.
    - distributed programming:
        - interception context sharing in order to ease behaviour sharing between advices.
        - uuid for advice identification in order to ease its use in a distributed context.
    - maintenable with well named variables and functions, comments and few lines.
    - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
    - respect of aspects vocabulary in order to ease its use among AOP users.
    - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
    - advices are callable objects.
    - Unit tests for all functions such as examples.
4. Benchmark:
    - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let's get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
   - less memory consumption in using the __slots__ class property.
   - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
   - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
   - joinpoint matching with function or regex.
   - distributed programming:
     - interception context sharing in order to ease behaviour sharing between advices.
     - uuid for advice identification in order to ease its use in a distributed context.
   - maintenable with well named variables and functions, comments and few lines.
   - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
   - respect of aspects vocabulary in order to ease its use among AOP users.
   - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
   - advices are callable objects.
   - Unit tests for all functions such as examples.
4. Benchmark:
   - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let’s get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
   - less memory consumption in using the __slots__ class property.
   - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
   - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
   - joinpoint matching with function or regex.
   - distributed programming:

    - interception context sharing in order to ease behaviour sharing between advices.
    - uuid for advice identification in order to ease its use in a distributed context.
  - maintenable with well named variables and functions, comments and few lines.
  - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
  - respect of aspects vocabulary in order to ease its use among AOP users.
  - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
  - advices are callable objects.
  - Unit tests for all functions such as examples.

4. Benchmark:
   - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let’s get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
   - less memory consumption in using the __slots__ class property.
   - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
   - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
   - joinpoint matching with function or regex.
   - distributed programming:
     - interception context sharing in order to ease behaviour sharing between advices.
     - uuid for advice identification in order to ease its use in a distributed context.
   - maintenable with well named variables and functions, comments and few lines.
   - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
   - respect of aspects vocabulary in order to ease its use among AOP users.
   - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
   - advices are callable objects.
   - Unit tests for all functions such as examples.
4. Benchmark:

- speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let's get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
   - less memory consumption in using the __slots__ class property.
   - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
   - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
   - joinpoint matching with function or regex.
   - distributed programming:
     - interception context sharing in order to ease behaviour sharing between advices.
     - uuid for advice identification in order to ease its use in a distributed context.
   - maintenable with well named variables and functions, comments and few lines.
   - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
   - respect of aspects vocabulary in order to ease its use among AOP users.
   - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
   - advices are callable objects.
   - Unit tests for all functions such as examples.
4. Benchmark:
   - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let’s get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
   - less memory consumption in using the __slots__ class property.
   - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
   - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
   - joinpoint matching with function or regex.
   - distributed programming:
     - interception context sharing in order to ease behaviour sharing between advices.
     - uuid for advice identification in order to ease its use in a distributed context.
   - maintenable with well named variables and functions, comments and few lines.
   - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
   - respect of aspects vocabulary in order to ease its use among AOP users.
   - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
   - advices are callable objects.
   - Unit tests for all functions such as examples.
4. Benchmark:
   - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let's get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
    - less memory consumption in using the __slots__ class property.
    - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
    - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
    - joinpoint matching with function or regex.
    - distributed programming:

    - interception context sharing in order to ease behaviour sharing between advices.
    - uuid for advice identification in order to ease its use in a distributed context.
  - maintenable with well named variables and functions, comments and few lines.
  - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
  - respect of aspects vocabulary in order to ease its use among AOP users.
  - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
  - advices are callable objects.
  - Unit tests for all functions such as examples.

4. Benchmark:
   - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let’s get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
   - less memory consumption in using the __slots__ class property.
   - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
   - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
   - joinpoint matching with function or regex.
   - distributed programming:
     - interception context sharing in order to ease behaviour sharing between advices.
     - uuid for advice identification in order to ease its use in a distributed context.
   - maintenable with well named variables and functions, comments and few lines.
   - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
   - respect of aspects vocabulary in order to ease its use among AOP users.
   - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
   - advices are callable objects.
   - Unit tests for all functions such as examples.
4. Benchmark:

- speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let's get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
    - less memory consumption in using the __slots__ class property.
    - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
    - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
    - joinpoint matching with function or regex.
    - distributed programming:
        - interception context sharing in order to ease behaviour sharing between advices.
        - uuid for advice identification in order to ease its use in a distributed context.
    - maintenable with well named variables and functions, comments and few lines.
    - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
    - respect of aspects vocabulary in order to ease its use among AOP users.
    - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
    - advices are callable objects.
    - Unit tests for all functions such as examples.
4. Benchmark:
    - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let's get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
   - less memory consumption in using the __slots__ class property.
   - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
   - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
   - joinpoint matching with function or regex.
   - distributed programming:
     - interception context sharing in order to ease behaviour sharing between advices.
     - uuid for advice identification in order to ease its use in a distributed context.
   - maintenable with well named variables and functions, comments and few lines.
   - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
   - respect of aspects vocabulary in order to ease its use among AOP users.
   - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
   - advices are callable objects.
   - Unit tests for all functions such as examples.
4. Benchmark:
   - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let's get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
    - less memory consumption in using the __slots__ class property.
    - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
    - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
    - joinpoint matching with function or regex.
    - distributed programming:

        - interception context sharing in order to ease behaviour sharing between advices.
        - uuid for advice identification in order to ease its use in a distributed context.
    - maintenable with well named variables and functions, comments and few lines.
    - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
    - respect of aspects vocabulary in order to ease its use among AOP users.
    - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
    - advices are callable objects.
    - Unit tests for all functions such as examples.

4. Benchmark:
    - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let's get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
   - less memory consumption in using the __slots__ class property.
   - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
   - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
   - joinpoint matching with function or regex.
   - distributed programming:
     - interception context sharing in order to ease behaviour sharing between advices.
     - uuid for advice identification in order to ease its use in a distributed context.
   - maintenable with well named variables and functions, comments and few lines.
   - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
   - respect of aspects vocabulary in order to ease its use among AOP users.
   - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
   - advices are callable objects.
   - Unit tests for all functions such as examples.
4. Benchmark:

- speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let's get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)   # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
    - less memory consumption in using the __slots__ class property.
    - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
    - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
    - joinpoint matching with function or regex.
    - distributed programming:
        - interception context sharing in order to ease behaviour sharing between advices.
        - uuid for advice identification in order to ease its use in a distributed context.
    - maintenable with well named variables and functions, comments and few lines.
    - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
    - respect of aspects vocabulary in order to ease its use among AOP users.
    - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
    - advices are callable objects.
    - Unit tests for all functions such as examples.
4. Benchmark:
    - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let's get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
    - less memory consumption in using the __slots__ class property.
    - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
    - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
    - joinpoint matching with function or regex.
    - distributed programming:
        - interception context sharing in order to ease behaviour sharing between advices.
        - uuid for advice identification in order to ease its use in a distributed context.
    - maintenable with well named variables and functions, comments and few lines.
    - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
    - respect of aspects vocabulary in order to ease its use among AOP users.
    - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
    - advices are callable objects.
    - Unit tests for all functions such as examples.
4. Benchmark:
    - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let’s get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
    - less memory consumption in using the __slots__ class property.
    - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
    - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
    - joinpoint matching with function or regex.
    - distributed programming:

    - interception context sharing in order to ease behaviour sharing between advices.
    - uuid for advice identification in order to ease its use in a distributed context.
  - maintenable with well named variables and functions, comments and few lines.
  - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
  - respect of aspects vocabulary in order to ease its use among AOP users.
  - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
  - advices are callable objects.
  - Unit tests for all functions such as examples.

4. Benchmark:
  - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let's get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
   - less memory consumption in using the __slots__ class property.
   - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
   - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
   - joinpoint matching with function or regex.
   - distributed programming:
     - interception context sharing in order to ease behaviour sharing between advices.
     - uuid for advice identification in order to ease its use in a distributed context.
   - maintenable with well named variables and functions, comments and few lines.
   - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
   - respect of aspects vocabulary in order to ease its use among AOP users.
   - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
   - advices are callable objects.
   - Unit tests for all functions such as examples.
4. Benchmark:

  - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let's get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
    - less memory consumption in using the __slots__ class property.
    - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
    - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
    - joinpoint matching with function or regex.
    - distributed programming:
        - interception context sharing in order to ease behaviour sharing between advices.
        - uuid for advice identification in order to ease its use in a distributed context.
    - maintenable with well named variables and functions, comments and few lines.
    - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
    - respect of aspects vocabulary in order to ease its use among AOP users.
    - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
    - advices are callable objects.
    - Unit tests for all functions such as examples.
4. Benchmark:
    - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let's get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
    - less memory consumption in using the __slots__ class property.
    - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
    - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
    - joinpoint matching with function or regex.
    - distributed programming:
        - interception context sharing in order to ease behaviour sharing between advices.
        - uuid for advice identification in order to ease its use in a distributed context.
    - maintenable with well named variables and functions, comments and few lines.
    - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
    - respect of aspects vocabulary in order to ease its use among AOP users.
    - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
    - advices are callable objects.
    - Unit tests for all functions such as examples.
4. Benchmark:
    - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let’s get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
    - less memory consumption in using the __slots__ class property.
    - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
    - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
    - joinpoint matching with function or regex.
    - distributed programming:

    - interception context sharing in order to ease behaviour sharing between advices.
    - uuid for advice identification in order to ease its use in a distributed context.
  - maintenable with well named variables and functions, comments and few lines.
  - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
  - respect of aspects vocabulary in order to ease its use among AOP users.
  - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
  - advices are callable objects.
  - Unit tests for all functions such as examples.

4. Benchmark:
  - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples

How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let's get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
    - less memory consumption in using the __slots__ class property.
    - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
    - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
    - joinpoint matching with function or regex.
    - distributed programming:
        - interception context sharing in order to ease behaviour sharing between advices.
        - uuid for advice identification in order to ease its use in a distributed context.
    - maintenable with well named variables and functions, comments and few lines.
    - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
    - respect of aspects vocabulary in order to ease its use among AOP users.
    - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
    - advices are callable objects.
    - Unit tests for all functions such as examples.
4. Benchmark:

- speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let's get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
    - less memory consumption in using the __slots__ class property.
    - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
    - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
    - joinpoint matching with function or regex.
    - distributed programming:
        - interception context sharing in order to ease behaviour sharing between advices.
        - uuid for advice identification in order to ease its use in a distributed context.
    - maintenable with well named variables and functions, comments and few lines.
    - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
    - respect of aspects vocabulary in order to ease its use among AOP users.
    - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
    - advices are callable objects.
    - Unit tests for all functions such as examples.
4. Benchmark:
    - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let's get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
   - less memory consumption in using the __slots__ class property.
   - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
   - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
   - joinpoint matching with function or regex.
   - distributed programming:
     - interception context sharing in order to ease behaviour sharing between advices.
     - uuid for advice identification in order to ease its use in a distributed context.
   - maintenable with well named variables and functions, comments and few lines.
   - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
   - respect of aspects vocabulary in order to ease its use among AOP users.
   - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
   - advices are callable objects.
   - Unit tests for all functions such as examples.
4. Benchmark:
   - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let’s get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
    - less memory consumption in using the __slots__ class property.
    - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
    - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
    - joinpoint matching with function or regex.
    - distributed programming:

    - interception context sharing in order to ease behaviour sharing between advices.
    - uuid for advice identification in order to ease its use in a distributed context.
  - maintenable with well named variables and functions, comments and few lines.
  - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
  - respect of aspects vocabulary in order to ease its use among AOP users.
  - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
  - advices are callable objects.
  - Unit tests for all functions such as examples.
4. Benchmark:
   - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let’s get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
    - less memory consumption in using the __slots__ class property.
    - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
    - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
    - joinpoint matching with function or regex.
    - distributed programming:
        - interception context sharing in order to ease behaviour sharing between advices.
        - uuid for advice identification in order to ease its use in a distributed context.
    - maintenable with well named variables and functions, comments and few lines.
    - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
    - respect of aspects vocabulary in order to ease its use among AOP users.
    - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
    - advices are callable objects.
    - Unit tests for all functions such as examples.
4. Benchmark:

- speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let’s get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
    - less memory consumption in using the __slots__ class property.
    - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
    - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
    - joinpoint matching with function or regex.
    - distributed programming:
        - interception context sharing in order to ease behaviour sharing between advices.
        - uuid for advice identification in order to ease its use in a distributed context.
    - maintenable with well named variables and functions, comments and few lines.
    - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
    - respect of aspects vocabulary in order to ease its use among AOP users.
    - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
    - advices are callable objects.
    - Unit tests for all functions such as examples.
4. Benchmark:
    - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let’s get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
    - less memory consumption in using the __slots__ class property.
    - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
    - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
    - joinpoint matching with function or regex.
    - distributed programming:
        - interception context sharing in order to ease behaviour sharing between advices.
        - uuid for advice identification in order to ease its use in a distributed context.
    - maintenable with well named variables and functions, comments and few lines.
    - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
    - respect of aspects vocabulary in order to ease its use among AOP users.
    - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
    - advices are callable objects.
    - Unit tests for all functions such as examples.
4. Benchmark:
    - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let’s get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
   - less memory consumption in using the __slots__ class property.
   - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
   - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
   - joinpoint matching with function or regex.
   - distributed programming:

    - interception context sharing in order to ease behaviour sharing between advices.
    - uuid for advice identification in order to ease its use in a distributed context.
  - maintenable with well named variables and functions, comments and few lines.
  - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
  - respect of aspects vocabulary in order to ease its use among AOP users.
  - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
  - advices are callable objects.
  - Unit tests for all functions such as examples.

4. Benchmark:
   - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let’s get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
    - less memory consumption in using the __slots__ class property.
    - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
    - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
    - joinpoint matching with function or regex.
    - distributed programming:
        - interception context sharing in order to ease behaviour sharing between advices.
        - uuid for advice identification in order to ease its use in a distributed context.
    - maintenable with well named variables and functions, comments and few lines.
    - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
    - respect of aspects vocabulary in order to ease its use among AOP users.
    - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
    - advices are callable objects.
    - Unit tests for all functions such as examples.
4. Benchmark:

- speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let's get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
    - less memory consumption in using the __slots__ class property.
    - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
    - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
    - joinpoint matching with function or regex.
    - distributed programming:
        - interception context sharing in order to ease behaviour sharing between advices.
        - uuid for advice identification in order to ease its use in a distributed context.
    - maintenable with well named variables and functions, comments and few lines.
    - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
    - respect of aspects vocabulary in order to ease its use among AOP users.
    - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
    - advices are callable objects.
    - Unit tests for all functions such as examples.
4. Benchmark:
    - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let's get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
   - less memory consumption in using the __slots__ class property.
   - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
   - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
   - joinpoint matching with function or regex.
   - distributed programming:
     - interception context sharing in order to ease behaviour sharing between advices.
     - uuid for advice identification in order to ease its use in a distributed context.
   - maintenable with well named variables and functions, comments and few lines.
   - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
   - respect of aspects vocabulary in order to ease its use among AOP users.
   - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
   - advices are callable objects.
   - Unit tests for all functions such as examples.
4. Benchmark:
   - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let's get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
    - less memory consumption in using the __slots__ class property.
    - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
    - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
    - joinpoint matching with function or regex.
    - distributed programming:

    - interception context sharing in order to ease behaviour sharing between advices.
    - uuid for advice identification in order to ease its use in a distributed context.
  - maintenable with well named variables and functions, comments and few lines.
  - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
  - respect of aspects vocabulary in order to ease its use among AOP users.
  - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
  - advices are callable objects.
  - Unit tests for all functions such as examples.

4. Benchmark:
   - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples

How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let’s get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
    - less memory consumption in using the __slots__ class property.
    - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
    - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
    - joinpoint matching with function or regex.
    - distributed programming:
        - interception context sharing in order to ease behaviour sharing between advices.
        - uuid for advice identification in order to ease its use in a distributed context.
    - maintenable with well named variables and functions, comments and few lines.
    - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
    - respect of aspects vocabulary in order to ease its use among AOP users.
    - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
    - advices are callable objects.
    - Unit tests for all functions such as examples.
4. Benchmark:

- speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let's get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
    - less memory consumption in using the __slots__ class property.
    - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
    - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
    - joinpoint matching with function or regex.
    - distributed programming:
        - interception context sharing in order to ease behaviour sharing between advices.
        - uuid for advice identification in order to ease its use in a distributed context.
    - maintenable with well named variables and functions, comments and few lines.
    - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
    - respect of aspects vocabulary in order to ease its use among AOP users.
    - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
    - advices are callable objects.
    - Unit tests for all functions such as examples.
4. Benchmark:
    - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let's get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
   - less memory consumption in using the __slots__ class property.
   - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
   - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
   - joinpoint matching with function or regex.
   - distributed programming:
     - interception context sharing in order to ease behaviour sharing between advices.
     - uuid for advice identification in order to ease its use in a distributed context.
   - maintenable with well named variables and functions, comments and few lines.
   - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
   - respect of aspects vocabulary in order to ease its use among AOP users.
   - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
   - advices are callable objects.
   - Unit tests for all functions such as examples.
4. Benchmark:
   - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let’s get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
    - less memory consumption in using the __slots__ class property.
    - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
    - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
    - joinpoint matching with function or regex.
    - distributed programming:

  - interception context sharing in order to ease behaviour sharing between advices.
  - uuid for advice identification in order to ease its use in a distributed context.
- maintenable with well named variables and functions, comments and few lines.
- extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
- respect of aspects vocabulary in order to ease its use among AOP users.
- close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
- advices are callable objects.
- Unit tests for all functions such as examples.

4. Benchmark:
   - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples

How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let’s get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
    - less memory consumption in using the __slots__ class property.
    - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
    - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
    - joinpoint matching with function or regex.
    - distributed programming:
        - interception context sharing in order to ease behaviour sharing between advices.
        - uuid for advice identification in order to ease its use in a distributed context.
    - maintenable with well named variables and functions, comments and few lines.
    - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
    - respect of aspects vocabulary in order to ease its use among AOP users.
    - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
    - advices are callable objects.
    - Unit tests for all functions such as examples.
4. Benchmark:

- speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let's get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
    - less memory consumption in using the __slots__ class property.
    - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
    - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
    - joinpoint matching with function or regex.
    - distributed programming:
        - interception context sharing in order to ease behaviour sharing between advices.
        - uuid for advice identification in order to ease its use in a distributed context.
    - maintenable with well named variables and functions, comments and few lines.
    - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
    - respect of aspects vocabulary in order to ease its use among AOP users.
    - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
    - advices are callable objects.
    - Unit tests for all functions such as examples.
4. Benchmark:
    - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let's get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
    - less memory consumption in using the __slots__ class property.
    - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
    - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
    - joinpoint matching with function or regex.
    - distributed programming:
        - interception context sharing in order to ease behaviour sharing between advices.
        - uuid for advice identification in order to ease its use in a distributed context.
    - maintenable with well named variables and functions, comments and few lines.
    - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
    - respect of aspects vocabulary in order to ease its use among AOP users.
    - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
    - advices are callable objects.
    - Unit tests for all functions such as examples.
4. Benchmark:
    - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let’s get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
    - less memory consumption in using the __slots__ class property.
    - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
    - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
    - joinpoint matching with function or regex.
    - distributed programming:

    - interception context sharing in order to ease behaviour sharing between advices.
    - uuid for advice identification in order to ease its use in a distributed context.
  - maintenable with well named variables and functions, comments and few lines.
  - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
  - respect of aspects vocabulary in order to ease its use among AOP users.
  - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
  - advices are callable objects.
  - Unit tests for all functions such as examples.

4. Benchmark:
   - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let’s get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
    - less memory consumption in using the __slots__ class property.
    - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
    - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
    - joinpoint matching with function or regex.
    - distributed programming:
        - interception context sharing in order to ease behaviour sharing between advices.
        - uuid for advice identification in order to ease its use in a distributed context.
    - maintenable with well named variables and functions, comments and few lines.
    - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
    - respect of aspects vocabulary in order to ease its use among AOP users.
    - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
    - advices are callable objects.
    - Unit tests for all functions such as examples.
4. Benchmark:

- speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let’s get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
    - less memory consumption in using the __slots__ class property.
    - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
    - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
    - joinpoint matching with function or regex.
    - distributed programming:
        - interception context sharing in order to ease behaviour sharing between advices.
        - uuid for advice identification in order to ease its use in a distributed context.
    - maintenable with well named variables and functions, comments and few lines.
    - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
    - respect of aspects vocabulary in order to ease its use among AOP users.
    - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
    - advices are callable objects.
    - Unit tests for all functions such as examples.
4. Benchmark:
    - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let's get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
    - less memory consumption in using the __slots__ class property.
    - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
    - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
    - joinpoint matching with function or regex.
    - distributed programming:
        - interception context sharing in order to ease behaviour sharing between advices.
        - uuid for advice identification in order to ease its use in a distributed context.
    - maintenable with well named variables and functions, comments and few lines.
    - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
    - respect of aspects vocabulary in order to ease its use among AOP users.
    - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
    - advices are callable objects.
    - Unit tests for all functions such as examples.
4. Benchmark:
    - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let’s get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
    - less memory consumption in using the __slots__ class property.
    - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
    - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
    - joinpoint matching with function or regex.
    - distributed programming:

    - interception context sharing in order to ease behaviour sharing between advices.
    - uuid for advice identification in order to ease its use in a distributed context.
  - maintenable with well named variables and functions, comments and few lines.
  - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
  - respect of aspects vocabulary in order to ease its use among AOP users.
  - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
  - advices are callable objects.
  - Unit tests for all functions such as examples.

4. Benchmark:
   - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let's get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
    - less memory consumption in using the __slots__ class property.
    - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
    - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
    - joinpoint matching with function or regex.
    - distributed programming:
        - interception context sharing in order to ease behaviour sharing between advices.
        - uuid for advice identification in order to ease its use in a distributed context.
    - maintenable with well named variables and functions, comments and few lines.
    - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
    - respect of aspects vocabulary in order to ease its use among AOP users.
    - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
    - advices are callable objects.
    - Unit tests for all functions such as examples.
4. Benchmark:

- speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let's get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)   # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
    - less memory consumption in using the __slots__ class property.
    - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
    - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
    - joinpoint matching with function or regex.
    - distributed programming:
        - interception context sharing in order to ease behaviour sharing between advices.
        - uuid for advice identification in order to ease its use in a distributed context.
    - maintenable with well named variables and functions, comments and few lines.
    - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
    - respect of aspects vocabulary in order to ease its use among AOP users.
    - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
    - advices are callable objects.
    - Unit tests for all functions such as examples.
4. Benchmark:
    - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let's get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
    - less memory consumption in using the __slots__ class property.
    - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
    - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
    - joinpoint matching with function or regex.
    - distributed programming:
        - interception context sharing in order to ease behaviour sharing between advices.
        - uuid for advice identification in order to ease its use in a distributed context.
    - maintenable with well named variables and functions, comments and few lines.
    - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
    - respect of aspects vocabulary in order to ease its use among AOP users.
    - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
    - advices are callable objects.
    - Unit tests for all functions such as examples.
4. Benchmark:
    - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let's get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
    - less memory consumption in using the __slots__ class property.
    - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
    - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
    - joinpoint matching with function or regex.
    - distributed programming:

    - interception context sharing in order to ease behaviour sharing between advices.
    - uuid for advice identification in order to ease its use in a distributed context.
  - maintenable with well named variables and functions, comments and few lines.
  - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
  - respect of aspects vocabulary in order to ease its use among AOP users.
  - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
  - advices are callable objects.
  - Unit tests for all functions such as examples.

4. Benchmark:

  - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let’s get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
    - less memory consumption in using the __slots__ class property.
    - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
    - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
    - joinpoint matching with function or regex.
    - distributed programming:
        - interception context sharing in order to ease behaviour sharing between advices.
        - uuid for advice identification in order to ease its use in a distributed context.
    - maintenable with well named variables and functions, comments and few lines.
    - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
    - respect of aspects vocabulary in order to ease its use among AOP users.
    - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
    - advices are callable objects.
    - Unit tests for all functions such as examples.
4. Benchmark:

- speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let's get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
    - less memory consumption in using the __slots__ class property.
    - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
    - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
    - joinpoint matching with function or regex.
    - distributed programming:
        - interception context sharing in order to ease behaviour sharing between advices.
        - uuid for advice identification in order to ease its use in a distributed context.
    - maintenable with well named variables and functions, comments and few lines.
    - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
    - respect of aspects vocabulary in order to ease its use among AOP users.
    - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
    - advices are callable objects.
    - Unit tests for all functions such as examples.
4. Benchmark:
    - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let's get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
   - less memory consumption in using the __slots__ class property.
   - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
   - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
   - joinpoint matching with function or regex.
   - distributed programming:
     - interception context sharing in order to ease behaviour sharing between advices.
     - uuid for advice identification in order to ease its use in a distributed context.
   - maintenable with well named variables and functions, comments and few lines.
   - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
   - respect of aspects vocabulary in order to ease its use among AOP users.
   - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
   - advices are callable objects.
   - Unit tests for all functions such as examples.
4. Benchmark:
   - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let's get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
    - less memory consumption in using the __slots__ class property.
    - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
    - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
    - joinpoint matching with function or regex.
    - distributed programming:

        - interception context sharing in order to ease behaviour sharing between advices.
        - uuid for advice identification in order to ease its use in a distributed context.
    - maintenable with well named variables and functions, comments and few lines.
    - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
    - respect of aspects vocabulary in order to ease its use among AOP users.
    - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
    - advices are callable objects.
    - Unit tests for all functions such as examples.

4. Benchmark:
    - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let’s get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
   - less memory consumption in using the __slots__ class property.
   - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
   - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
   - joinpoint matching with function or regex.
   - distributed programming:
     - interception context sharing in order to ease behaviour sharing between advices.
     - uuid for advice identification in order to ease its use in a distributed context.
   - maintenable with well named variables and functions, comments and few lines.
   - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
   - respect of aspects vocabulary in order to ease its use among AOP users.
   - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
   - advices are callable objects.
   - Unit tests for all functions such as examples.
4. Benchmark:

- speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let's get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
   - less memory consumption in using the __slots__ class property.
   - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
   - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
   - joinpoint matching with function or regex.
   - distributed programming:
     - interception context sharing in order to ease behaviour sharing between advices.
     - uuid for advice identification in order to ease its use in a distributed context.
   - maintenable with well named variables and functions, comments and few lines.
   - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
   - respect of aspects vocabulary in order to ease its use among AOP users.
   - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
   - advices are callable objects.
   - Unit tests for all functions such as examples.
4. Benchmark:
   - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let's get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
    - less memory consumption in using the __slots__ class property.
    - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
    - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
    - joinpoint matching with function or regex.
    - distributed programming:
        - interception context sharing in order to ease behaviour sharing between advices.
        - uuid for advice identification in order to ease its use in a distributed context.
    - maintenable with well named variables and functions, comments and few lines.
    - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
    - respect of aspects vocabulary in order to ease its use among AOP users.
    - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
    - advices are callable objects.
    - Unit tests for all functions such as examples.
4. Benchmark:
    - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let’s get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
    - less memory consumption in using the __slots__ class property.
    - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
    - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
    - joinpoint matching with function or regex.
    - distributed programming:

        - interception context sharing in order to ease behaviour sharing between advices.
        - uuid for advice identification in order to ease its use in a distributed context.
    - maintenable with well named variables and functions, comments and few lines.
    - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
    - respect of aspects vocabulary in order to ease its use among AOP users.
    - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
    - advices are callable objects.
    - Unit tests for all functions such as examples.
4. Benchmark:
    - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples

How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let’s get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
    - less memory consumption in using the __slots__ class property.
    - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
    - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
    - joinpoint matching with function or regex.
    - distributed programming:
        - interception context sharing in order to ease behaviour sharing between advices.
        - uuid for advice identification in order to ease its use in a distributed context.
    - maintenable with well named variables and functions, comments and few lines.
    - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
    - respect of aspects vocabulary in order to ease its use among AOP users.
    - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
    - advices are callable objects.
    - Unit tests for all functions such as examples.
4. Benchmark:

- speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let’s get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
    - less memory consumption in using the __slots__ class property.
    - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
    - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
    - joinpoint matching with function or regex.
    - distributed programming:
        - interception context sharing in order to ease behaviour sharing between advices.
        - uuid for advice identification in order to ease its use in a distributed context.
    - maintenable with well named variables and functions, comments and few lines.
    - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
    - respect of aspects vocabulary in order to ease its use among AOP users.
    - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
    - advices are callable objects.
    - Unit tests for all functions such as examples.
4. Benchmark:
    - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let's get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
    - less memory consumption in using the __slots__ class property.
    - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
    - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
    - joinpoint matching with function or regex.
    - distributed programming:
        - interception context sharing in order to ease behaviour sharing between advices.
        - uuid for advice identification in order to ease its use in a distributed context.
    - maintenable with well named variables and functions, comments and few lines.
    - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
    - respect of aspects vocabulary in order to ease its use among AOP users.
    - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
    - advices are callable objects.
    - Unit tests for all functions such as examples.
4. Benchmark:
    - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let’s get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
    - less memory consumption in using the __slots__ class property.
    - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
    - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
    - joinpoint matching with function or regex.
    - distributed programming:

  - interception context sharing in order to ease behaviour sharing between advices.
  - uuid for advice identification in order to ease its use in a distributed context.
- maintenable with well named variables and functions, comments and few lines.
- extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
- respect of aspects vocabulary in order to ease its use among AOP users.
- close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
- advices are callable objects.
- Unit tests for all functions such as examples.

4. Benchmark:
   - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples

How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let’s get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
   - less memory consumption in using the __slots__ class property.
   - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
   - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
   - joinpoint matching with function or regex.
   - distributed programming:
     - interception context sharing in order to ease behaviour sharing between advices.
     - uuid for advice identification in order to ease its use in a distributed context.
   - maintenable with well named variables and functions, comments and few lines.
   - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
   - respect of aspects vocabulary in order to ease its use among AOP users.
   - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
   - advices are callable objects.
   - Unit tests for all functions such as examples.
4. Benchmark:

- speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let's get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)   # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
    - less memory consumption in using the __slots__ class property.
    - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
    - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
    - joinpoint matching with function or regex.
    - distributed programming:
        - interception context sharing in order to ease behaviour sharing between advices.
        - uuid for advice identification in order to ease its use in a distributed context.
    - maintenable with well named variables and functions, comments and few lines.
    - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
    - respect of aspects vocabulary in order to ease its use among AOP users.
    - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
    - advices are callable objects.
    - Unit tests for all functions such as examples.
4. Benchmark:
    - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let's get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
    - less memory consumption in using the __slots__ class property.
    - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
    - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
    - joinpoint matching with function or regex.
    - distributed programming:
        - interception context sharing in order to ease behaviour sharing between advices.
        - uuid for advice identification in order to ease its use in a distributed context.
    - maintenable with well named variables and functions, comments and few lines.
    - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
    - respect of aspects vocabulary in order to ease its use among AOP users.
    - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
    - advices are callable objects.
    - Unit tests for all functions such as examples.
4. Benchmark:
    - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples

How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let’s get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
    - less memory consumption in using the __slots__ class property.
    - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
    - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
    - joinpoint matching with function or regex.
    - distributed programming:

  - interception context sharing in order to ease behaviour sharing between advices.
  - uuid for advice identification in order to ease its use in a distributed context.
- maintenable with well named variables and functions, comments and few lines.
- extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
- respect of aspects vocabulary in order to ease its use among AOP users.
- close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
- advices are callable objects.
- Unit tests for all functions such as examples.

4. Benchmark:
   - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples

How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let's get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
    - less memory consumption in using the __slots__ class property.
    - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
    - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
    - joinpoint matching with function or regex.
    - distributed programming:
        - interception context sharing in order to ease behaviour sharing between advices.
        - uuid for advice identification in order to ease its use in a distributed context.
    - maintenable with well named variables and functions, comments and few lines.
    - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
    - respect of aspects vocabulary in order to ease its use among AOP users.
    - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
    - advices are callable objects.
    - Unit tests for all functions such as examples.
4. Benchmark:

  - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let's get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
    - less memory consumption in using the __slots__ class property.
    - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
    - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
    - joinpoint matching with function or regex.
    - distributed programming:
        - interception context sharing in order to ease behaviour sharing between advices.
        - uuid for advice identification in order to ease its use in a distributed context.
    - maintenable with well named variables and functions, comments and few lines.
    - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
    - respect of aspects vocabulary in order to ease its use among AOP users.
    - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
    - advices are callable objects.
    - Unit tests for all functions such as examples.
4. Benchmark:
    - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let’s get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
    - less memory consumption in using the __slots__ class property.
    - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
    - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
    - joinpoint matching with function or regex.
    - distributed programming:
        - interception context sharing in order to ease behaviour sharing between advices.
        - uuid for advice identification in order to ease its use in a distributed context.
    - maintenable with well named variables and functions, comments and few lines.
    - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
    - respect of aspects vocabulary in order to ease its use among AOP users.
    - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
    - advices are callable objects.
    - Unit tests for all functions such as examples.
4. Benchmark:
    - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let’s get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
    - less memory consumption in using the __slots__ class property.
    - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
    - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
    - joinpoint matching with function or regex.
    - distributed programming:

  - interception context sharing in order to ease behaviour sharing between advices.
  - uuid for advice identification in order to ease its use in a distributed context.
- maintenable with well named variables and functions, comments and few lines.
- extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
- respect of aspects vocabulary in order to ease its use among AOP users.
- close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
- advices are callable objects.
- Unit tests for all functions such as examples.

4. Benchmark:
   - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let’s get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
   - less memory consumption in using the __slots__ class property.
   - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
   - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
   - joinpoint matching with function or regex.
   - distributed programming:
     - interception context sharing in order to ease behaviour sharing between advices.
     - uuid for advice identification in order to ease its use in a distributed context.
   - maintenable with well named variables and functions, comments and few lines.
   - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
   - respect of aspects vocabulary in order to ease its use among AOP users.
   - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
   - advices are callable objects.
   - Unit tests for all functions such as examples.
4. Benchmark:

  - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let's get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
    - less memory consumption in using the __slots__ class property.
    - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
    - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
    - joinpoint matching with function or regex.
    - distributed programming:
        - interception context sharing in order to ease behaviour sharing between advices.
        - uuid for advice identification in order to ease its use in a distributed context.
    - maintenable with well named variables and functions, comments and few lines.
    - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
    - respect of aspects vocabulary in order to ease its use among AOP users.
    - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
    - advices are callable objects.
    - Unit tests for all functions such as examples.
4. Benchmark:
    - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let's get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
    - less memory consumption in using the __slots__ class property.
    - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
    - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
    - joinpoint matching with function or regex.
    - distributed programming:
        - interception context sharing in order to ease behaviour sharing between advices.
        - uuid for advice identification in order to ease its use in a distributed context.
    - maintenable with well named variables and functions, comments and few lines.
    - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
    - respect of aspects vocabulary in order to ease its use among AOP users.
    - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
    - advices are callable objects.
    - Unit tests for all functions such as examples.
4. Benchmark:
    - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let’s get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
    - less memory consumption in using the __slots__ class property.
    - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
    - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
    - joinpoint matching with function or regex.
    - distributed programming:

    - interception context sharing in order to ease behaviour sharing between advices.
    - uuid for advice identification in order to ease its use in a distributed context.
  - maintenable with well named variables and functions, comments and few lines.
  - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
  - respect of aspects vocabulary in order to ease its use among AOP users.
  - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
  - advices are callable objects.
  - Unit tests for all functions such as examples.

4. Benchmark:
   - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let’s get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
   - less memory consumption in using the __slots__ class property.
   - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
   - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
   - joinpoint matching with function or regex.
   - distributed programming:
     - interception context sharing in order to ease behaviour sharing between advices.
     - uuid for advice identification in order to ease its use in a distributed context.
   - maintenable with well named variables and functions, comments and few lines.
   - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
   - respect of aspects vocabulary in order to ease its use among AOP users.
   - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
   - advices are callable objects.
   - Unit tests for all functions such as examples.
4. Benchmark:

   - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let's get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
    - less memory consumption in using the __slots__ class property.
    - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
    - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
    - joinpoint matching with function or regex.
    - distributed programming:
        - interception context sharing in order to ease behaviour sharing between advices.
        - uuid for advice identification in order to ease its use in a distributed context.
    - maintenable with well named variables and functions, comments and few lines.
    - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
    - respect of aspects vocabulary in order to ease its use among AOP users.
    - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
    - advices are callable objects.
    - Unit tests for all functions such as examples.
4. Benchmark:
    - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let's get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
   - less memory consumption in using the __slots__ class property.
   - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
   - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
   - joinpoint matching with function or regex.
   - distributed programming:
     - interception context sharing in order to ease behaviour sharing between advices.
     - uuid for advice identification in order to ease its use in a distributed context.
   - maintenable with well named variables and functions, comments and few lines.
   - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
   - respect of aspects vocabulary in order to ease its use among AOP users.
   - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
   - advices are callable objects.
   - Unit tests for all functions such as examples.
4. Benchmark:
   - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let’s get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
    - less memory consumption in using the __slots__ class property.
    - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
    - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
    - joinpoint matching with function or regex.
    - distributed programming:

    - interception context sharing in order to ease behaviour sharing between advices.
    - uuid for advice identification in order to ease its use in a distributed context.
  - maintenable with well named variables and functions, comments and few lines.
  - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
  - respect of aspects vocabulary in order to ease its use among AOP users.
  - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
  - advices are callable objects.
  - Unit tests for all functions such as examples.

4. Benchmark:
   - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let’s get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
   - less memory consumption in using the __slots__ class property.
   - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
   - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
   - joinpoint matching with function or regex.
   - distributed programming:
     - interception context sharing in order to ease behaviour sharing between advices.
     - uuid for advice identification in order to ease its use in a distributed context.
   - maintenable with well named variables and functions, comments and few lines.
   - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
   - respect of aspects vocabulary in order to ease its use among AOP users.
   - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
   - advices are callable objects.
   - Unit tests for all functions such as examples.
4. Benchmark:

- speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let's get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)   # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
   - less memory consumption in using the __slots__ class property.
   - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
   - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
   - joinpoint matching with function or regex.
   - distributed programming:
     - interception context sharing in order to ease behaviour sharing between advices.
     - uuid for advice identification in order to ease its use in a distributed context.
   - maintenable with well named variables and functions, comments and few lines.
   - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
   - respect of aspects vocabulary in order to ease its use among AOP users.
   - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
   - advices are callable objects.
   - Unit tests for all functions such as examples.
4. Benchmark:
   - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let's get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
    - less memory consumption in using the __slots__ class property.
    - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
    - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
    - joinpoint matching with function or regex.
    - distributed programming:
        - interception context sharing in order to ease behaviour sharing between advices.
        - uuid for advice identification in order to ease its use in a distributed context.
    - maintenable with well named variables and functions, comments and few lines.
    - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
    - respect of aspects vocabulary in order to ease its use among AOP users.
    - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
    - advices are callable objects.
    - Unit tests for all functions such as examples.
4. Benchmark:
    - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let’s get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
    - less memory consumption in using the __slots__ class property.
    - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
    - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
    - joinpoint matching with function or regex.
    - distributed programming:

        - interception context sharing in order to ease behaviour sharing between advices.
        - uuid for advice identification in order to ease its use in a distributed context.
    - maintenable with well named variables and functions, comments and few lines.
    - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
    - respect of aspects vocabulary in order to ease its use among AOP users.
    - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
    - advices are callable objects.
    - Unit tests for all functions such as examples.

4. Benchmark:
    - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let’s get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
   - less memory consumption in using the __slots__ class property.
   - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
   - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
   - joinpoint matching with function or regex.
   - distributed programming:
     - interception context sharing in order to ease behaviour sharing between advices.
     - uuid for advice identification in order to ease its use in a distributed context.
   - maintenable with well named variables and functions, comments and few lines.
   - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
   - respect of aspects vocabulary in order to ease its use among AOP users.
   - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
   - advices are callable objects.
   - Unit tests for all functions such as examples.
4. Benchmark:

- speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let's get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
    - less memory consumption in using the __slots__ class property.
    - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
    - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
    - joinpoint matching with function or regex.
    - distributed programming:
        - interception context sharing in order to ease behaviour sharing between advices.
        - uuid for advice identification in order to ease its use in a distributed context.
    - maintenable with well named variables and functions, comments and few lines.
    - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
    - respect of aspects vocabulary in order to ease its use among AOP users.
    - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
    - advices are callable objects.
    - Unit tests for all functions such as examples.
4. Benchmark:
    - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let's get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
   - less memory consumption in using the __slots__ class property.
   - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
   - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
   - joinpoint matching with function or regex.
   - distributed programming:
     - interception context sharing in order to ease behaviour sharing between advices.
     - uuid for advice identification in order to ease its use in a distributed context.
   - maintenable with well named variables and functions, comments and few lines.
   - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
   - respect of aspects vocabulary in order to ease its use among AOP users.
   - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
   - advices are callable objects.
   - Unit tests for all functions such as examples.
4. Benchmark:
   - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let’s get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
    - less memory consumption in using the __slots__ class property.
    - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
    - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
    - joinpoint matching with function or regex.
    - distributed programming:

        - interception context sharing in order to ease behaviour sharing between advices.
        - uuid for advice identification in order to ease its use in a distributed context.
    - maintenable with well named variables and functions, comments and few lines.
    - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
    - respect of aspects vocabulary in order to ease its use among AOP users.
    - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
    - advices are callable objects.
    - Unit tests for all functions such as examples.

4. Benchmark:
    - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let's get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
    - less memory consumption in using the __slots__ class property.
    - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
    - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
    - joinpoint matching with function or regex.
    - distributed programming:
        - interception context sharing in order to ease behaviour sharing between advices.
        - uuid for advice identification in order to ease its use in a distributed context.
    - maintenable with well named variables and functions, comments and few lines.
    - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
    - respect of aspects vocabulary in order to ease its use among AOP users.
    - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
    - advices are callable objects.
    - Unit tests for all functions such as examples.
4. Benchmark:

  - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let's get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
   - less memory consumption in using the __slots__ class property.
   - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
   - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
   - joinpoint matching with function or regex.
   - distributed programming:
     - interception context sharing in order to ease behaviour sharing between advices.
     - uuid for advice identification in order to ease its use in a distributed context.
   - maintenable with well named variables and functions, comments and few lines.
   - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
   - respect of aspects vocabulary in order to ease its use among AOP users.
   - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
   - advices are callable objects.
   - Unit tests for all functions such as examples.
4. Benchmark:
   - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let's get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
    - less memory consumption in using the __slots__ class property.
    - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
    - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
    - joinpoint matching with function or regex.
    - distributed programming:
        - interception context sharing in order to ease behaviour sharing between advices.
        - uuid for advice identification in order to ease its use in a distributed context.
    - maintenable with well named variables and functions, comments and few lines.
    - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
    - respect of aspects vocabulary in order to ease its use among AOP users.
    - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
    - advices are callable objects.
    - Unit tests for all functions such as examples.
4. Benchmark:
    - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let's get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
    - less memory consumption in using the __slots__ class property.
    - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
    - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
    - joinpoint matching with function or regex.
    - distributed programming:

    - interception context sharing in order to ease behaviour sharing between advices.
    - uuid for advice identification in order to ease its use in a distributed context.
  - maintenable with well named variables and functions, comments and few lines.
  - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
  - respect of aspects vocabulary in order to ease its use among AOP users.
  - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
  - advices are callable objects.
  - Unit tests for all functions such as examples.

4. Benchmark:
   - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let’s get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
    - less memory consumption in using the __slots__ class property.
    - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
    - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
    - joinpoint matching with function or regex.
    - distributed programming:
        - interception context sharing in order to ease behaviour sharing between advices.
        - uuid for advice identification in order to ease its use in a distributed context.
    - maintenable with well named variables and functions, comments and few lines.
    - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
    - respect of aspects vocabulary in order to ease its use among AOP users.
    - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
    - advices are callable objects.
    - Unit tests for all functions such as examples.
4. Benchmark:

- speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let’s get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
   - less memory consumption in using the __slots__ class property.
   - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
   - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
   - joinpoint matching with function or regex.
   - distributed programming:
     - interception context sharing in order to ease behaviour sharing between advices.
     - uuid for advice identification in order to ease its use in a distributed context.
   - maintenable with well named variables and functions, comments and few lines.
   - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
   - respect of aspects vocabulary in order to ease its use among AOP users.
   - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
   - advices are callable objects.
   - Unit tests for all functions such as examples.
4. Benchmark:
   - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let's get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
    - less memory consumption in using the __slots__ class property.
    - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
    - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
    - joinpoint matching with function or regex.
    - distributed programming:
        - interception context sharing in order to ease behaviour sharing between advices.
        - uuid for advice identification in order to ease its use in a distributed context.
    - maintenable with well named variables and functions, comments and few lines.
    - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
    - respect of aspects vocabulary in order to ease its use among AOP users.
    - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
    - advices are callable objects.
    - Unit tests for all functions such as examples.
4. Benchmark:
    - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let’s get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
    - less memory consumption in using the __slots__ class property.
    - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
    - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
    - joinpoint matching with function or regex.
    - distributed programming:

    - interception context sharing in order to ease behaviour sharing between advices.
    - uuid for advice identification in order to ease its use in a distributed context.
  - maintenable with well named variables and functions, comments and few lines.
  - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
  - respect of aspects vocabulary in order to ease its use among AOP users.
  - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
  - advices are callable objects.
  - Unit tests for all functions such as examples.

4. Benchmark:
  - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let’s get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
   - less memory consumption in using the __slots__ class property.
   - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
   - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
   - joinpoint matching with function or regex.
   - distributed programming:
     - interception context sharing in order to ease behaviour sharing between advices.
     - uuid for advice identification in order to ease its use in a distributed context.
   - maintenable with well named variables and functions, comments and few lines.
   - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
   - respect of aspects vocabulary in order to ease its use among AOP users.
   - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
   - advices are callable objects.
   - Unit tests for all functions such as examples.
4. Benchmark:

- speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let's get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)   # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
   - less memory consumption in using the __slots__ class property.
   - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
   - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
   - joinpoint matching with function or regex.
   - distributed programming:
     - interception context sharing in order to ease behaviour sharing between advices.
     - uuid for advice identification in order to ease its use in a distributed context.
   - maintenable with well named variables and functions, comments and few lines.
   - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
   - respect of aspects vocabulary in order to ease its use among AOP users.
   - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
   - advices are callable objects.
   - Unit tests for all functions such as examples.
4. Benchmark:
   - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let's get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
    - less memory consumption in using the __slots__ class property.
    - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
    - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
    - joinpoint matching with function or regex.
    - distributed programming:
        - interception context sharing in order to ease behaviour sharing between advices.
        - uuid for advice identification in order to ease its use in a distributed context.
    - maintenable with well named variables and functions, comments and few lines.
    - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
    - respect of aspects vocabulary in order to ease its use among AOP users.
    - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
    - advices are callable objects.
    - Unit tests for all functions such as examples.
4. Benchmark:
    - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let’s get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
   - less memory consumption in using the __slots__ class property.
   - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
   - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
   - joinpoint matching with function or regex.
   - distributed programming:

  - interception context sharing in order to ease behaviour sharing between advices.
  - uuid for advice identification in order to ease its use in a distributed context.
- maintenable with well named variables and functions, comments and few lines.
- extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
- respect of aspects vocabulary in order to ease its use among AOP users.
- close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
- advices are callable objects.
- Unit tests for all functions such as examples.

4. Benchmark:
   - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let’s get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
   - less memory consumption in using the __slots__ class property.
   - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
   - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
   - joinpoint matching with function or regex.
   - distributed programming:
     - interception context sharing in order to ease behaviour sharing between advices.
     - uuid for advice identification in order to ease its use in a distributed context.
   - maintenable with well named variables and functions, comments and few lines.
   - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
   - respect of aspects vocabulary in order to ease its use among AOP users.
   - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
   - advices are callable objects.
   - Unit tests for all functions such as examples.
4. Benchmark:

- speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let's get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
    - less memory consumption in using the __slots__ class property.
    - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
    - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
    - joinpoint matching with function or regex.
    - distributed programming:
        - interception context sharing in order to ease behaviour sharing between advices.
        - uuid for advice identification in order to ease its use in a distributed context.
    - maintenable with well named variables and functions, comments and few lines.
    - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
    - respect of aspects vocabulary in order to ease its use among AOP users.
    - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
    - advices are callable objects.
    - Unit tests for all functions such as examples.
4. Benchmark:
    - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let's get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
    - less memory consumption in using the __slots__ class property.
    - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
    - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
    - joinpoint matching with function or regex.
    - distributed programming:
        - interception context sharing in order to ease behaviour sharing between advices.
        - uuid for advice identification in order to ease its use in a distributed context.
    - maintenable with well named variables and functions, comments and few lines.
    - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
    - respect of aspects vocabulary in order to ease its use among AOP users.
    - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
    - advices are callable objects.
    - Unit tests for all functions such as examples.
4. Benchmark:
    - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let’s get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
    - less memory consumption in using the __slots__ class property.
    - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
    - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
    - joinpoint matching with function or regex.
    - distributed programming:

    - interception context sharing in order to ease behaviour sharing between advices.
    - uuid for advice identification in order to ease its use in a distributed context.
  - maintenable with well named variables and functions, comments and few lines.
  - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
  - respect of aspects vocabulary in order to ease its use among AOP users.
  - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
  - advices are callable objects.
  - Unit tests for all functions such as examples.

4. Benchmark:
   - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let’s get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
    - less memory consumption in using the __slots__ class property.
    - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
    - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
    - joinpoint matching with function or regex.
    - distributed programming:
        - interception context sharing in order to ease behaviour sharing between advices.
        - uuid for advice identification in order to ease its use in a distributed context.
    - maintenable with well named variables and functions, comments and few lines.
    - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
    - respect of aspects vocabulary in order to ease its use among AOP users.
    - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
    - advices are callable objects.
    - Unit tests for all functions such as examples.
4. Benchmark:

  - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let's get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
    - less memory consumption in using the __slots__ class property.
    - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
    - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
    - joinpoint matching with function or regex.
    - distributed programming:
        - interception context sharing in order to ease behaviour sharing between advices.
        - uuid for advice identification in order to ease its use in a distributed context.
    - maintenable with well named variables and functions, comments and few lines.
    - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
    - respect of aspects vocabulary in order to ease its use among AOP users.
    - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
    - advices are callable objects.
    - Unit tests for all functions such as examples.
4. Benchmark:
    - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let's get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
   - less memory consumption in using the __slots__ class property.
   - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
   - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
   - joinpoint matching with function or regex.
   - distributed programming:
     - interception context sharing in order to ease behaviour sharing between advices.
     - uuid for advice identification in order to ease its use in a distributed context.
   - maintenable with well named variables and functions, comments and few lines.
   - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
   - respect of aspects vocabulary in order to ease its use among AOP users.
   - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
   - advices are callable objects.
   - Unit tests for all functions such as examples.
4. Benchmark:
   - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let’s get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
   - less memory consumption in using the __slots__ class property.
   - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
   - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
   - joinpoint matching with function or regex.
   - distributed programming:

    - interception context sharing in order to ease behaviour sharing between advices.
    - uuid for advice identification in order to ease its use in a distributed context.
  - maintenable with well named variables and functions, comments and few lines.
  - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
  - respect of aspects vocabulary in order to ease its use among AOP users.
  - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
  - advices are callable objects.
  - Unit tests for all functions such as examples.

4. Benchmark:
   - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples

How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let’s get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
   - less memory consumption in using the __slots__ class property.
   - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
   - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
   - joinpoint matching with function or regex.
   - distributed programming:
     - interception context sharing in order to ease behaviour sharing between advices.
     - uuid for advice identification in order to ease its use in a distributed context.
   - maintenable with well named variables and functions, comments and few lines.
   - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
   - respect of aspects vocabulary in order to ease its use among AOP users.
   - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
   - advices are callable objects.
   - Unit tests for all functions such as examples.
4. Benchmark:

- speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let's get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
    - less memory consumption in using the __slots__ class property.
    - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
    - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
    - joinpoint matching with function or regex.
    - distributed programming:
        - interception context sharing in order to ease behaviour sharing between advices.
        - uuid for advice identification in order to ease its use in a distributed context.
    - maintenable with well named variables and functions, comments and few lines.
    - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
    - respect of aspects vocabulary in order to ease its use among AOP users.
    - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
    - advices are callable objects.
    - Unit tests for all functions such as examples.
4. Benchmark:
    - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let's get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
    - less memory consumption in using the __slots__ class property.
    - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
    - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
    - joinpoint matching with function or regex.
    - distributed programming:
        - interception context sharing in order to ease behaviour sharing between advices.
        - uuid for advice identification in order to ease its use in a distributed context.
    - maintenable with well named variables and functions, comments and few lines.
    - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
    - respect of aspects vocabulary in order to ease its use among AOP users.
    - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
    - advices are callable objects.
    - Unit tests for all functions such as examples.
4. Benchmark:
    - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples

How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let’s get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
    - less memory consumption in using the __slots__ class property.
    - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
    - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
    - joinpoint matching with function or regex.
    - distributed programming:

    - interception context sharing in order to ease behaviour sharing between advices.
    - uuid for advice identification in order to ease its use in a distributed context.
  - maintenable with well named variables and functions, comments and few lines.
  - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
  - respect of aspects vocabulary in order to ease its use among AOP users.
  - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
  - advices are callable objects.
  - Unit tests for all functions such as examples.

4. Benchmark:
   - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let’s get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
    - less memory consumption in using the __slots__ class property.
    - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
    - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
    - joinpoint matching with function or regex.
    - distributed programming:
        - interception context sharing in order to ease behaviour sharing between advices.
        - uuid for advice identification in order to ease its use in a distributed context.
    - maintenable with well named variables and functions, comments and few lines.
    - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
    - respect of aspects vocabulary in order to ease its use among AOP users.
    - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
    - advices are callable objects.
    - Unit tests for all functions such as examples.
4. Benchmark:

  - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let's get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)   # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
    - less memory consumption in using the __slots__ class property.
    - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
    - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
    - joinpoint matching with function or regex.
    - distributed programming:
        - interception context sharing in order to ease behaviour sharing between advices.
        - uuid for advice identification in order to ease its use in a distributed context.
    - maintenable with well named variables and functions, comments and few lines.
    - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
    - respect of aspects vocabulary in order to ease its use among AOP users.
    - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
    - advices are callable objects.
    - Unit tests for all functions such as examples.
4. Benchmark:
    - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let's get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
    - less memory consumption in using the __slots__ class property.
    - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
    - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
    - joinpoint matching with function or regex.
    - distributed programming:
        - interception context sharing in order to ease behaviour sharing between advices.
        - uuid for advice identification in order to ease its use in a distributed context.
    - maintenable with well named variables and functions, comments and few lines.
    - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
    - respect of aspects vocabulary in order to ease its use among AOP users.
    - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
    - advices are callable objects.
    - Unit tests for all functions such as examples.
4. Benchmark:
    - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let's get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
    - less memory consumption in using the __slots__ class property.
    - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
    - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
    - joinpoint matching with function or regex.
    - distributed programming:

  - interception context sharing in order to ease behaviour sharing between advices.
  - uuid for advice identification in order to ease its use in a distributed context.
- maintenable with well named variables and functions, comments and few lines.
- extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
- respect of aspects vocabulary in order to ease its use among AOP users.
- close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
- advices are callable objects.
- Unit tests for all functions such as examples.

4. Benchmark:
   - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples

How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let’s get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
    - less memory consumption in using the __slots__ class property.
    - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
    - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
    - joinpoint matching with function or regex.
    - distributed programming:
        - interception context sharing in order to ease behaviour sharing between advices.
        - uuid for advice identification in order to ease its use in a distributed context.
    - maintenable with well named variables and functions, comments and few lines.
    - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
    - respect of aspects vocabulary in order to ease its use among AOP users.
    - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
    - advices are callable objects.
    - Unit tests for all functions such as examples.
4. Benchmark:

- speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let's get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)   # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
    - less memory consumption in using the __slots__ class property.
    - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
    - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
    - joinpoint matching with function or regex.
    - distributed programming:
        - interception context sharing in order to ease behaviour sharing between advices.
        - uuid for advice identification in order to ease its use in a distributed context.
    - maintenable with well named variables and functions, comments and few lines.
    - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
    - respect of aspects vocabulary in order to ease its use among AOP users.
    - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
    - advices are callable objects.
    - Unit tests for all functions such as examples.
4. Benchmark:
    - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let's get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
    - less memory consumption in using the __slots__ class property.
    - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
    - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
    - joinpoint matching with function or regex.
    - distributed programming:
        - interception context sharing in order to ease behaviour sharing between advices.
        - uuid for advice identification in order to ease its use in a distributed context.
    - maintenable with well named variables and functions, comments and few lines.
    - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
    - respect of aspects vocabulary in order to ease its use among AOP users.
    - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
    - advices are callable objects.
    - Unit tests for all functions such as examples.
4. Benchmark:
    - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let’s get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
   - less memory consumption in using the __slots__ class property.
   - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
   - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
   - joinpoint matching with function or regex.
   - distributed programming:

- interception context sharing in order to ease behaviour sharing between advices.
    - uuid for advice identification in order to ease its use in a distributed context.
- maintenable with well named variables and functions, comments and few lines.
- extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
- respect of aspects vocabulary in order to ease its use among AOP users.
- close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
- advices are callable objects.
- Unit tests for all functions such as examples.

4. Benchmark:
    - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let's get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
    - less memory consumption in using the __slots__ class property.
    - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
    - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
    - joinpoint matching with function or regex.
    - distributed programming:
        - interception context sharing in order to ease behaviour sharing between advices.
        - uuid for advice identification in order to ease its use in a distributed context.
    - maintenable with well named variables and functions, comments and few lines.
    - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
    - respect of aspects vocabulary in order to ease its use among AOP users.
    - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
    - advices are callable objects.
    - Unit tests for all functions such as examples.
4. Benchmark:

  - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let's get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
    - less memory consumption in using the __slots__ class property.
    - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
    - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
    - joinpoint matching with function or regex.
    - distributed programming:
        - interception context sharing in order to ease behaviour sharing between advices.
        - uuid for advice identification in order to ease its use in a distributed context.
    - maintenable with well named variables and functions, comments and few lines.
    - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
    - respect of aspects vocabulary in order to ease its use among AOP users.
    - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
    - advices are callable objects.
    - Unit tests for all functions such as examples.
4. Benchmark:
    - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let's get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
    - less memory consumption in using the __slots__ class property.
    - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
    - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
    - joinpoint matching with function or regex.
    - distributed programming:
        - interception context sharing in order to ease behaviour sharing between advices.
        - uuid for advice identification in order to ease its use in a distributed context.
    - maintenable with well named variables and functions, comments and few lines.
    - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
    - respect of aspects vocabulary in order to ease its use among AOP users.
    - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
    - advices are callable objects.
    - Unit tests for all functions such as examples.
4. Benchmark:
    - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let’s get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
    - less memory consumption in using the __slots__ class property.
    - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
    - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
    - joinpoint matching with function or regex.
    - distributed programming:

    - interception context sharing in order to ease behaviour sharing between advices.
    - uuid for advice identification in order to ease its use in a distributed context.
  - maintenable with well named variables and functions, comments and few lines.
  - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
  - respect of aspects vocabulary in order to ease its use among AOP users.
  - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
  - advices are callable objects.
  - Unit tests for all functions such as examples.

4. Benchmark:
   - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let's get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
    - less memory consumption in using the __slots__ class property.
    - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
    - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
    - joinpoint matching with function or regex.
    - distributed programming:
        - interception context sharing in order to ease behaviour sharing between advices.
        - uuid for advice identification in order to ease its use in a distributed context.
    - maintenable with well named variables and functions, comments and few lines.
    - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
    - respect of aspects vocabulary in order to ease its use among AOP users.
    - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
    - advices are callable objects.
    - Unit tests for all functions such as examples.
4. Benchmark:

  - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let’s get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
    - less memory consumption in using the __slots__ class property.
    - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
    - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
    - joinpoint matching with function or regex.
    - distributed programming:
        - interception context sharing in order to ease behaviour sharing between advices.
        - uuid for advice identification in order to ease its use in a distributed context.
    - maintenable with well named variables and functions, comments and few lines.
    - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
    - respect of aspects vocabulary in order to ease its use among AOP users.
    - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
    - advices are callable objects.
    - Unit tests for all functions such as examples.
4. Benchmark:
    - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let's get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
   - less memory consumption in using the __slots__ class property.
   - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
   - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
   - joinpoint matching with function or regex.
   - distributed programming:
     - interception context sharing in order to ease behaviour sharing between advices.
     - uuid for advice identification in order to ease its use in a distributed context.
   - maintenable with well named variables and functions, comments and few lines.
   - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
   - respect of aspects vocabulary in order to ease its use among AOP users.
   - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
   - advices are callable objects.
   - Unit tests for all functions such as examples.
4. Benchmark:
   - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let’s get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
    - less memory consumption in using the __slots__ class property.
    - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
    - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
    - joinpoint matching with function or regex.
    - distributed programming:

    - interception context sharing in order to ease behaviour sharing between advices.
    - uuid for advice identification in order to ease its use in a distributed context.
  - maintenable with well named variables and functions, comments and few lines.
  - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
  - respect of aspects vocabulary in order to ease its use among AOP users.
  - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
  - advices are callable objects.
  - Unit tests for all functions such as examples.

4. Benchmark:
   - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let’s get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
   - less memory consumption in using the __slots__ class property.
   - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
   - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
   - joinpoint matching with function or regex.
   - distributed programming:
     - interception context sharing in order to ease behaviour sharing between advices.
     - uuid for advice identification in order to ease its use in a distributed context.
   - maintenable with well named variables and functions, comments and few lines.
   - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
   - respect of aspects vocabulary in order to ease its use among AOP users.
   - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
   - advices are callable objects.
   - Unit tests for all functions such as examples.
4. Benchmark:

- speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let's get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)   # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
    - less memory consumption in using the __slots__ class property.
    - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
    - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
    - joinpoint matching with function or regex.
    - distributed programming:
        - interception context sharing in order to ease behaviour sharing between advices.
        - uuid for advice identification in order to ease its use in a distributed context.
    - maintenable with well named variables and functions, comments and few lines.
    - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
    - respect of aspects vocabulary in order to ease its use among AOP users.
    - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
    - advices are callable objects.
    - Unit tests for all functions such as examples.
4. Benchmark:
    - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let's get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
    - less memory consumption in using the __slots__ class property.
    - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
    - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
    - joinpoint matching with function or regex.
    - distributed programming:
        - interception context sharing in order to ease behaviour sharing between advices.
        - uuid for advice identification in order to ease its use in a distributed context.
    - maintenable with well named variables and functions, comments and few lines.
    - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
    - respect of aspects vocabulary in order to ease its use among AOP users.
    - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
    - advices are callable objects.
    - Unit tests for all functions such as examples.
4. Benchmark:
    - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let’s get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
    - less memory consumption in using the __slots__ class property.
    - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
    - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
    - joinpoint matching with function or regex.
    - distributed programming:

    - interception context sharing in order to ease behaviour sharing between advices.
    - uuid for advice identification in order to ease its use in a distributed context.
  - maintenable with well named variables and functions, comments and few lines.
  - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
  - respect of aspects vocabulary in order to ease its use among AOP users.
  - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
  - advices are callable objects.
  - Unit tests for all functions such as examples.

4. Benchmark:
   - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let’s get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
    - less memory consumption in using the __slots__ class property.
    - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
    - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
    - joinpoint matching with function or regex.
    - distributed programming:
        - interception context sharing in order to ease behaviour sharing between advices.
        - uuid for advice identification in order to ease its use in a distributed context.
    - maintenable with well named variables and functions, comments and few lines.
    - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
    - respect of aspects vocabulary in order to ease its use among AOP users.
    - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
    - advices are callable objects.
    - Unit tests for all functions such as examples.
4. Benchmark:

- speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let's get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
   - less memory consumption in using the __slots__ class property.
   - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
   - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
   - joinpoint matching with function or regex.
   - distributed programming:
     - interception context sharing in order to ease behaviour sharing between advices.
     - uuid for advice identification in order to ease its use in a distributed context.
   - maintenable with well named variables and functions, comments and few lines.
   - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
   - respect of aspects vocabulary in order to ease its use among AOP users.
   - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
   - advices are callable objects.
   - Unit tests for all functions such as examples.
4. Benchmark:
   - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let's get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
    - less memory consumption in using the __slots__ class property.
    - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
    - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
    - joinpoint matching with function or regex.
    - distributed programming:
        - interception context sharing in order to ease behaviour sharing between advices.
        - uuid for advice identification in order to ease its use in a distributed context.
    - maintenable with well named variables and functions, comments and few lines.
    - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
    - respect of aspects vocabulary in order to ease its use among AOP users.
    - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
    - advices are callable objects.
    - Unit tests for all functions such as examples.
4. Benchmark:
    - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let’s get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
    - less memory consumption in using the __slots__ class property.
    - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
    - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
    - joinpoint matching with function or regex.
    - distributed programming:

    - interception context sharing in order to ease behaviour sharing between advices.
    - uuid for advice identification in order to ease its use in a distributed context.
  - maintenable with well named variables and functions, comments and few lines.
  - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
  - respect of aspects vocabulary in order to ease its use among AOP users.
  - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
  - advices are callable objects.
  - Unit tests for all functions such as examples.

4. Benchmark:
   - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let’s get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
    - less memory consumption in using the __slots__ class property.
    - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
    - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
    - joinpoint matching with function or regex.
    - distributed programming:
        - interception context sharing in order to ease behaviour sharing between advices.
        - uuid for advice identification in order to ease its use in a distributed context.
    - maintenable with well named variables and functions, comments and few lines.
    - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
    - respect of aspects vocabulary in order to ease its use among AOP users.
    - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
    - advices are callable objects.
    - Unit tests for all functions such as examples.
4. Benchmark:

- speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let's get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
   - less memory consumption in using the __slots__ class property.
   - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
   - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
   - joinpoint matching with function or regex.
   - distributed programming:
     - interception context sharing in order to ease behaviour sharing between advices.
     - uuid for advice identification in order to ease its use in a distributed context.
   - maintenable with well named variables and functions, comments and few lines.
   - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
   - respect of aspects vocabulary in order to ease its use among AOP users.
   - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
   - advices are callable objects.
   - Unit tests for all functions such as examples.
4. Benchmark:
   - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let's get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
   - less memory consumption in using the __slots__ class property.
   - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
   - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
   - joinpoint matching with function or regex.
   - distributed programming:
     - interception context sharing in order to ease behaviour sharing between advices.
     - uuid for advice identification in order to ease its use in a distributed context.
   - maintenable with well named variables and functions, comments and few lines.
   - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
   - respect of aspects vocabulary in order to ease its use among AOP users.
   - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
   - advices are callable objects.
   - Unit tests for all functions such as examples.
4. Benchmark:
   - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let’s get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
    - less memory consumption in using the __slots__ class property.
    - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
    - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
    - joinpoint matching with function or regex.
    - distributed programming:

    - interception context sharing in order to ease behaviour sharing between advices.
    - uuid for advice identification in order to ease its use in a distributed context.
  - maintenable with well named variables and functions, comments and few lines.
  - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
  - respect of aspects vocabulary in order to ease its use among AOP users.
  - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
  - advices are callable objects.
  - Unit tests for all functions such as examples.

4. Benchmark:
  - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let’s get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
   - less memory consumption in using the __slots__ class property.
   - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
   - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
   - joinpoint matching with function or regex.
   - distributed programming:
     - interception context sharing in order to ease behaviour sharing between advices.
     - uuid for advice identification in order to ease its use in a distributed context.
   - maintenable with well named variables and functions, comments and few lines.
   - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
   - respect of aspects vocabulary in order to ease its use among AOP users.
   - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
   - advices are callable objects.
   - Unit tests for all functions such as examples.
4. Benchmark:

- speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let's get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
   - less memory consumption in using the __slots__ class property.
   - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
   - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
   - joinpoint matching with function or regex.
   - distributed programming:
     - interception context sharing in order to ease behaviour sharing between advices.
     - uuid for advice identification in order to ease its use in a distributed context.
   - maintenable with well named variables and functions, comments and few lines.
   - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
   - respect of aspects vocabulary in order to ease its use among AOP users.
   - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
   - advices are callable objects.
   - Unit tests for all functions such as examples.
4. Benchmark:
   - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let's get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
    - less memory consumption in using the __slots__ class property.
    - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
    - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
    - joinpoint matching with function or regex.
    - distributed programming:
        - interception context sharing in order to ease behaviour sharing between advices.
        - uuid for advice identification in order to ease its use in a distributed context.
    - maintenable with well named variables and functions, comments and few lines.
    - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
    - respect of aspects vocabulary in order to ease its use among AOP users.
    - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
    - advices are callable objects.
    - Unit tests for all functions such as examples.
4. Benchmark:
    - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let’s get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
    - less memory consumption in using the __slots__ class property.
    - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
    - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
    - joinpoint matching with function or regex.
    - distributed programming:

        - interception context sharing in order to ease behaviour sharing between advices.
        - uuid for advice identification in order to ease its use in a distributed context.
    - maintenable with well named variables and functions, comments and few lines.
    - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
    - respect of aspects vocabulary in order to ease its use among AOP users.
    - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
    - advices are callable objects.
    - Unit tests for all functions such as examples.

4. Benchmark:
    - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let's get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
    - less memory consumption in using the __slots__ class property.
    - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
    - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
    - joinpoint matching with function or regex.
    - distributed programming:
        - interception context sharing in order to ease behaviour sharing between advices.
        - uuid for advice identification in order to ease its use in a distributed context.
    - maintenable with well named variables and functions, comments and few lines.
    - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
    - respect of aspects vocabulary in order to ease its use among AOP users.
    - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
    - advices are callable objects.
    - Unit tests for all functions such as examples.
4. Benchmark:

- speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let's get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
   - less memory consumption in using the __slots__ class property.
   - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
   - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
   - joinpoint matching with function or regex.
   - distributed programming:
     - interception context sharing in order to ease behaviour sharing between advices.
     - uuid for advice identification in order to ease its use in a distributed context.
   - maintenable with well named variables and functions, comments and few lines.
   - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
   - respect of aspects vocabulary in order to ease its use among AOP users.
   - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
   - advices are callable objects.
   - Unit tests for all functions such as examples.
4. Benchmark:
   - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let's get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
    - less memory consumption in using the __slots__ class property.
    - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
    - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
    - joinpoint matching with function or regex.
    - distributed programming:
        - interception context sharing in order to ease behaviour sharing between advices.
        - uuid for advice identification in order to ease its use in a distributed context.
    - maintenable with well named variables and functions, comments and few lines.
    - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
    - respect of aspects vocabulary in order to ease its use among AOP users.
    - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
    - advices are callable objects.
    - Unit tests for all functions such as examples.
4. Benchmark:
    - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let’s get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
    - less memory consumption in using the __slots__ class property.
    - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
    - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
    - joinpoint matching with function or regex.
    - distributed programming:

    - interception context sharing in order to ease behaviour sharing between advices.
    - uuid for advice identification in order to ease its use in a distributed context.
  - maintenable with well named variables and functions, comments and few lines.
  - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
  - respect of aspects vocabulary in order to ease its use among AOP users.
  - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
  - advices are callable objects.
  - Unit tests for all functions such as examples.

4. Benchmark:
   - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples

How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let’s get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
   - less memory consumption in using the __slots__ class property.
   - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
   - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
   - joinpoint matching with function or regex.
   - distributed programming:
     - interception context sharing in order to ease behaviour sharing between advices.
     - uuid for advice identification in order to ease its use in a distributed context.
   - maintenable with well named variables and functions, comments and few lines.
   - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
   - respect of aspects vocabulary in order to ease its use among AOP users.
   - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
   - advices are callable objects.
   - Unit tests for all functions such as examples.
4. Benchmark:

  - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let's get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)   # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
    - less memory consumption in using the __slots__ class property.
    - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
    - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
    - joinpoint matching with function or regex.
    - distributed programming:
        - interception context sharing in order to ease behaviour sharing between advices.
        - uuid for advice identification in order to ease its use in a distributed context.
    - maintenable with well named variables and functions, comments and few lines.
    - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
    - respect of aspects vocabulary in order to ease its use among AOP users.
    - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
    - advices are callable objects.
    - Unit tests for all functions such as examples.
4. Benchmark:
    - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let's get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
    - less memory consumption in using the __slots__ class property.
    - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
    - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
    - joinpoint matching with function or regex.
    - distributed programming:
        - interception context sharing in order to ease behaviour sharing between advices.
        - uuid for advice identification in order to ease its use in a distributed context.
    - maintenable with well named variables and functions, comments and few lines.
    - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
    - respect of aspects vocabulary in order to ease its use among AOP users.
    - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
    - advices are callable objects.
    - Unit tests for all functions such as examples.
4. Benchmark:
    - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let's get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
    - less memory consumption in using the __slots__ class property.
    - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
    - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
    - joinpoint matching with function or regex.
    - distributed programming:

    - interception context sharing in order to ease behaviour sharing between advices.
    - uuid for advice identification in order to ease its use in a distributed context.
  - maintenable with well named variables and functions, comments and few lines.
  - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
  - respect of aspects vocabulary in order to ease its use among AOP users.
  - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
  - advices are callable objects.
  - Unit tests for all functions such as examples.

4. Benchmark:
   - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let’s get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
   - less memory consumption in using the __slots__ class property.
   - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
   - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
   - joinpoint matching with function or regex.
   - distributed programming:
     - interception context sharing in order to ease behaviour sharing between advices.
     - uuid for advice identification in order to ease its use in a distributed context.
   - maintenable with well named variables and functions, comments and few lines.
   - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
   - respect of aspects vocabulary in order to ease its use among AOP users.
   - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
   - advices are callable objects.
   - Unit tests for all functions such as examples.
4. Benchmark:

- speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let's get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)   # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
   - less memory consumption in using the __slots__ class property.
   - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
   - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
   - joinpoint matching with function or regex.
   - distributed programming:
     - interception context sharing in order to ease behaviour sharing between advices.
     - uuid for advice identification in order to ease its use in a distributed context.
   - maintenable with well named variables and functions, comments and few lines.
   - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
   - respect of aspects vocabulary in order to ease its use among AOP users.
   - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
   - advices are callable objects.
   - Unit tests for all functions such as examples.
4. Benchmark:
   - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let's get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
    - less memory consumption in using the __slots__ class property.
    - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
    - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
    - joinpoint matching with function or regex.
    - distributed programming:
        - interception context sharing in order to ease behaviour sharing between advices.
        - uuid for advice identification in order to ease its use in a distributed context.
    - maintenable with well named variables and functions, comments and few lines.
    - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
    - respect of aspects vocabulary in order to ease its use among AOP users.
    - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
    - advices are callable objects.
    - Unit tests for all functions such as examples.
4. Benchmark:
    - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let’s get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
    - less memory consumption in using the __slots__ class property.
    - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
    - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
    - joinpoint matching with function or regex.
    - distributed programming:

  - interception context sharing in order to ease behaviour sharing between advices.
  - uuid for advice identification in order to ease its use in a distributed context.
- maintenable with well named variables and functions, comments and few lines.
- extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
- respect of aspects vocabulary in order to ease its use among AOP users.
- close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
- advices are callable objects.
- Unit tests for all functions such as examples.

4. Benchmark:
   - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let's get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
   - less memory consumption in using the __slots__ class property.
   - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
   - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
   - joinpoint matching with function or regex.
   - distributed programming:
     - interception context sharing in order to ease behaviour sharing between advices.
     - uuid for advice identification in order to ease its use in a distributed context.
   - maintenable with well named variables and functions, comments and few lines.
   - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
   - respect of aspects vocabulary in order to ease its use among AOP users.
   - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
   - advices are callable objects.
   - Unit tests for all functions such as examples.
4. Benchmark:

- speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let's get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
   - less memory consumption in using the __slots__ class property.
   - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
   - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
   - joinpoint matching with function or regex.
   - distributed programming:
     - interception context sharing in order to ease behaviour sharing between advices.
     - uuid for advice identification in order to ease its use in a distributed context.
   - maintenable with well named variables and functions, comments and few lines.
   - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
   - respect of aspects vocabulary in order to ease its use among AOP users.
   - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
   - advices are callable objects.
   - Unit tests for all functions such as examples.
4. Benchmark:
   - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let's get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
    - less memory consumption in using the __slots__ class property.
    - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
    - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
    - joinpoint matching with function or regex.
    - distributed programming:
        - interception context sharing in order to ease behaviour sharing between advices.
        - uuid for advice identification in order to ease its use in a distributed context.
    - maintenable with well named variables and functions, comments and few lines.
    - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
    - respect of aspects vocabulary in order to ease its use among AOP users.
    - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
    - advices are callable objects.
    - Unit tests for all functions such as examples.
4. Benchmark:
    - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let’s get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
    - less memory consumption in using the __slots__ class property.
    - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
    - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
    - joinpoint matching with function or regex.
    - distributed programming:

      - interception context sharing in order to ease behaviour sharing between advices.
      - uuid for advice identification in order to ease its use in a distributed context.
   - maintenable with well named variables and functions, comments and few lines.
   - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
   - respect of aspects vocabulary in order to ease its use among AOP users.
   - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
   - advices are callable objects.
   - Unit tests for all functions such as examples.

4. Benchmark:
   - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples

How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let’s get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
    - less memory consumption in using the __slots__ class property.
    - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
    - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
    - joinpoint matching with function or regex.
    - distributed programming:
        - interception context sharing in order to ease behaviour sharing between advices.
        - uuid for advice identification in order to ease its use in a distributed context.
    - maintenable with well named variables and functions, comments and few lines.
    - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
    - respect of aspects vocabulary in order to ease its use among AOP users.
    - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
    - advices are callable objects.
    - Unit tests for all functions such as examples.
4. Benchmark:

- speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let’s get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
    - less memory consumption in using the __slots__ class property.
    - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
    - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
    - joinpoint matching with function or regex.
    - distributed programming:
        - interception context sharing in order to ease behaviour sharing between advices.
        - uuid for advice identification in order to ease its use in a distributed context.
    - maintenable with well named variables and functions, comments and few lines.
    - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
    - respect of aspects vocabulary in order to ease its use among AOP users.
    - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
    - advices are callable objects.
    - Unit tests for all functions such as examples.
4. Benchmark:
    - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let's get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
   - less memory consumption in using the __slots__ class property.
   - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
   - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
   - joinpoint matching with function or regex.
   - distributed programming:
     - interception context sharing in order to ease behaviour sharing between advices.
     - uuid for advice identification in order to ease its use in a distributed context.
   - maintenable with well named variables and functions, comments and few lines.
   - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
   - respect of aspects vocabulary in order to ease its use among AOP users.
   - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
   - advices are callable objects.
   - Unit tests for all functions such as examples.
4. Benchmark:
   - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let's get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
   - less memory consumption in using the __slots__ class property.
   - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
   - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
   - joinpoint matching with function or regex.
   - distributed programming:

    - interception context sharing in order to ease behaviour sharing between advices.
    - uuid for advice identification in order to ease its use in a distributed context.
  - maintenable with well named variables and functions, comments and few lines.
  - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
  - respect of aspects vocabulary in order to ease its use among AOP users.
  - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
  - advices are callable objects.
  - Unit tests for all functions such as examples.

4. Benchmark:
   - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let’s get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
    - less memory consumption in using the __slots__ class property.
    - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
    - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
    - joinpoint matching with function or regex.
    - distributed programming:
        - interception context sharing in order to ease behaviour sharing between advices.
        - uuid for advice identification in order to ease its use in a distributed context.
    - maintenable with well named variables and functions, comments and few lines.
    - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
    - respect of aspects vocabulary in order to ease its use among AOP users.
    - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
    - advices are callable objects.
    - Unit tests for all functions such as examples.
4. Benchmark:

  - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let's get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)   # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
   - less memory consumption in using the __slots__ class property.
   - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
   - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
   - joinpoint matching with function or regex.
   - distributed programming:
     - interception context sharing in order to ease behaviour sharing between advices.
     - uuid for advice identification in order to ease its use in a distributed context.
   - maintenable with well named variables and functions, comments and few lines.
   - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
   - respect of aspects vocabulary in order to ease its use among AOP users.
   - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
   - advices are callable objects.
   - Unit tests for all functions such as examples.
4. Benchmark:
   - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let's get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
   - less memory consumption in using the __slots__ class property.
   - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
   - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
   - joinpoint matching with function or regex.
   - distributed programming:
     - interception context sharing in order to ease behaviour sharing between advices.
     - uuid for advice identification in order to ease its use in a distributed context.
   - maintenable with well named variables and functions, comments and few lines.
   - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
   - respect of aspects vocabulary in order to ease its use among AOP users.
   - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
   - advices are callable objects.
   - Unit tests for all functions such as examples.
4. Benchmark:
   - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let’s get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
    - less memory consumption in using the __slots__ class property.
    - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
    - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
    - joinpoint matching with function or regex.
    - distributed programming:

    - interception context sharing in order to ease behaviour sharing between advices.
    - uuid for advice identification in order to ease its use in a distributed context.
  - maintenable with well named variables and functions, comments and few lines.
  - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
  - respect of aspects vocabulary in order to ease its use among AOP users.
  - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
  - advices are callable objects.
  - Unit tests for all functions such as examples.

4. Benchmark:
   - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let’s get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
   - less memory consumption in using the __slots__ class property.
   - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
   - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
   - joinpoint matching with function or regex.
   - distributed programming:
     - interception context sharing in order to ease behaviour sharing between advices.
     - uuid for advice identification in order to ease its use in a distributed context.
   - maintenable with well named variables and functions, comments and few lines.
   - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
   - respect of aspects vocabulary in order to ease its use among AOP users.
   - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
   - advices are callable objects.
   - Unit tests for all functions such as examples.
4. Benchmark:

- speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let's get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
    - less memory consumption in using the __slots__ class property.
    - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
    - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
    - joinpoint matching with function or regex.
    - distributed programming:
        - interception context sharing in order to ease behaviour sharing between advices.
        - uuid for advice identification in order to ease its use in a distributed context.
    - maintenable with well named variables and functions, comments and few lines.
    - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
    - respect of aspects vocabulary in order to ease its use among AOP users.
    - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
    - advices are callable objects.
    - Unit tests for all functions such as examples.
4. Benchmark:
    - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let's get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
    - less memory consumption in using the __slots__ class property.
    - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
    - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
    - joinpoint matching with function or regex.
    - distributed programming:
        - interception context sharing in order to ease behaviour sharing between advices.
        - uuid for advice identification in order to ease its use in a distributed context.
    - maintenable with well named variables and functions, comments and few lines.
    - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
    - respect of aspects vocabulary in order to ease its use among AOP users.
    - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
    - advices are callable objects.
    - Unit tests for all functions such as examples.
4. Benchmark:
    - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let’s get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
    - less memory consumption in using the __slots__ class property.
    - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
    - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
    - joinpoint matching with function or regex.
    - distributed programming:

    - interception context sharing in order to ease behaviour sharing between advices.
    - uuid for advice identification in order to ease its use in a distributed context.
  - maintenable with well named variables and functions, comments and few lines.
  - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
  - respect of aspects vocabulary in order to ease its use among AOP users.
  - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
  - advices are callable objects.
  - Unit tests for all functions such as examples.

4. Benchmark:
   - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let's get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
   - less memory consumption in using the __slots__ class property.
   - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
   - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
   - joinpoint matching with function or regex.
   - distributed programming:
     - interception context sharing in order to ease behaviour sharing between advices.
     - uuid for advice identification in order to ease its use in a distributed context.
   - maintenable with well named variables and functions, comments and few lines.
   - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
   - respect of aspects vocabulary in order to ease its use among AOP users.
   - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
   - advices are callable objects.
   - Unit tests for all functions such as examples.
4. Benchmark:

- speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let's get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)   # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
   - less memory consumption in using the __slots__ class property.
   - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
   - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
   - joinpoint matching with function or regex.
   - distributed programming:
     - interception context sharing in order to ease behaviour sharing between advices.
     - uuid for advice identification in order to ease its use in a distributed context.
   - maintenable with well named variables and functions, comments and few lines.
   - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
   - respect of aspects vocabulary in order to ease its use among AOP users.
   - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
   - advices are callable objects.
   - Unit tests for all functions such as examples.
4. Benchmark:
   - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let's get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
   - less memory consumption in using the __slots__ class property.
   - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
   - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
   - joinpoint matching with function or regex.
   - distributed programming:
     - interception context sharing in order to ease behaviour sharing between advices.
     - uuid for advice identification in order to ease its use in a distributed context.
   - maintenable with well named variables and functions, comments and few lines.
   - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
   - respect of aspects vocabulary in order to ease its use among AOP users.
   - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
   - advices are callable objects.
   - Unit tests for all functions such as examples.
4. Benchmark:
   - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let’s get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
    - less memory consumption in using the __slots__ class property.
    - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
    - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
    - joinpoint matching with function or regex.
    - distributed programming:

    - interception context sharing in order to ease behaviour sharing between advices.
    - uuid for advice identification in order to ease its use in a distributed context.
  - maintenable with well named variables and functions, comments and few lines.
  - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
  - respect of aspects vocabulary in order to ease its use among AOP users.
  - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
  - advices are callable objects.
  - Unit tests for all functions such as examples.

4. Benchmark:
   - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let's get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
   - less memory consumption in using the __slots__ class property.
   - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
   - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
   - joinpoint matching with function or regex.
   - distributed programming:
     - interception context sharing in order to ease behaviour sharing between advices.
     - uuid for advice identification in order to ease its use in a distributed context.
   - maintenable with well named variables and functions, comments and few lines.
   - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
   - respect of aspects vocabulary in order to ease its use among AOP users.
   - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
   - advices are callable objects.
   - Unit tests for all functions such as examples.
4. Benchmark:

  - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let's get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)   # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
    - less memory consumption in using the __slots__ class property.
    - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
    - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
    - joinpoint matching with function or regex.
    - distributed programming:
        - interception context sharing in order to ease behaviour sharing between advices.
        - uuid for advice identification in order to ease its use in a distributed context.
    - maintenable with well named variables and functions, comments and few lines.
    - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
    - respect of aspects vocabulary in order to ease its use among AOP users.
    - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
    - advices are callable objects.
    - Unit tests for all functions such as examples.
4. Benchmark:
    - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let's get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
   - less memory consumption in using the __slots__ class property.
   - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
   - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
   - joinpoint matching with function or regex.
   - distributed programming:
     - interception context sharing in order to ease behaviour sharing between advices.
     - uuid for advice identification in order to ease its use in a distributed context.
   - maintenable with well named variables and functions, comments and few lines.
   - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
   - respect of aspects vocabulary in order to ease its use among AOP users.
   - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
   - advices are callable objects.
   - Unit tests for all functions such as examples.
4. Benchmark:
   - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let’s get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
    - less memory consumption in using the __slots__ class property.
    - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
    - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
    - joinpoint matching with function or regex.
    - distributed programming:

    - interception context sharing in order to ease behaviour sharing between advices.
    - uuid for advice identification in order to ease its use in a distributed context.
  - maintenable with well named variables and functions, comments and few lines.
  - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
  - respect of aspects vocabulary in order to ease its use among AOP users.
  - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
  - advices are callable objects.
  - Unit tests for all functions such as examples.

4. Benchmark:
   - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples

How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let’s get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
    - less memory consumption in using the __slots__ class property.
    - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
    - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
    - joinpoint matching with function or regex.
    - distributed programming:
        - interception context sharing in order to ease behaviour sharing between advices.
        - uuid for advice identification in order to ease its use in a distributed context.
    - maintenable with well named variables and functions, comments and few lines.
    - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
    - respect of aspects vocabulary in order to ease its use among AOP users.
    - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
    - advices are callable objects.
    - Unit tests for all functions such as examples.
4. Benchmark:

- speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let's get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
   - less memory consumption in using the __slots__ class property.
   - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
   - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
   - joinpoint matching with function or regex.
   - distributed programming:
     - interception context sharing in order to ease behaviour sharing between advices.
     - uuid for advice identification in order to ease its use in a distributed context.
   - maintenable with well named variables and functions, comments and few lines.
   - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
   - respect of aspects vocabulary in order to ease its use among AOP users.
   - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
   - advices are callable objects.
   - Unit tests for all functions such as examples.
4. Benchmark:
   - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let's get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
   - less memory consumption in using the __slots__ class property.
   - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
   - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
   - joinpoint matching with function or regex.
   - distributed programming:
     - interception context sharing in order to ease behaviour sharing between advices.
     - uuid for advice identification in order to ease its use in a distributed context.
   - maintenable with well named variables and functions, comments and few lines.
   - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
   - respect of aspects vocabulary in order to ease its use among AOP users.
   - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
   - advices are callable objects.
   - Unit tests for all functions such as examples.
4. Benchmark:
   - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let’s get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
    - less memory consumption in using the __slots__ class property.
    - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
    - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
    - joinpoint matching with function or regex.
    - distributed programming:

    - interception context sharing in order to ease behaviour sharing between advices.
    - uuid for advice identification in order to ease its use in a distributed context.
  - maintenable with well named variables and functions, comments and few lines.
  - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
  - respect of aspects vocabulary in order to ease its use among AOP users.
  - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
  - advices are callable objects.
  - Unit tests for all functions such as examples.

4. Benchmark:

  - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let's get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
   - less memory consumption in using the __slots__ class property.
   - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
   - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
   - joinpoint matching with function or regex.
   - distributed programming:
     - interception context sharing in order to ease behaviour sharing between advices.
     - uuid for advice identification in order to ease its use in a distributed context.
   - maintenable with well named variables and functions, comments and few lines.
   - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
   - respect of aspects vocabulary in order to ease its use among AOP users.
   - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
   - advices are callable objects.
   - Unit tests for all functions such as examples.
4. Benchmark:

- speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let's get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)   # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
    - less memory consumption in using the __slots__ class property.
    - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
    - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
    - joinpoint matching with function or regex.
    - distributed programming:
        - interception context sharing in order to ease behaviour sharing between advices.
        - uuid for advice identification in order to ease its use in a distributed context.
    - maintenable with well named variables and functions, comments and few lines.
    - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
    - respect of aspects vocabulary in order to ease its use among AOP users.
    - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
    - advices are callable objects.
    - Unit tests for all functions such as examples.
4. Benchmark:
    - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let's get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
    - less memory consumption in using the __slots__ class property.
    - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
    - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
    - joinpoint matching with function or regex.
    - distributed programming:
        - interception context sharing in order to ease behaviour sharing between advices.
        - uuid for advice identification in order to ease its use in a distributed context.
    - maintenable with well named variables and functions, comments and few lines.
    - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
    - respect of aspects vocabulary in order to ease its use among AOP users.
    - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
    - advices are callable objects.
    - Unit tests for all functions such as examples.
4. Benchmark:
    - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let’s get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
    - less memory consumption in using the __slots__ class property.
    - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
    - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
    - joinpoint matching with function or regex.
    - distributed programming:

    - interception context sharing in order to ease behaviour sharing between advices.
    - uuid for advice identification in order to ease its use in a distributed context.
  - maintenable with well named variables and functions, comments and few lines.
  - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
  - respect of aspects vocabulary in order to ease its use among AOP users.
  - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
  - advices are callable objects.
  - Unit tests for all functions such as examples.

4. Benchmark:
   - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples

How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let’s get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
   - less memory consumption in using the __slots__ class property.
   - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
   - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
   - joinpoint matching with function or regex.
   - distributed programming:
     - interception context sharing in order to ease behaviour sharing between advices.
     - uuid for advice identification in order to ease its use in a distributed context.
   - maintenable with well named variables and functions, comments and few lines.
   - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
   - respect of aspects vocabulary in order to ease its use among AOP users.
   - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
   - advices are callable objects.
   - Unit tests for all functions such as examples.
4. Benchmark:

- speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let's get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
    - less memory consumption in using the __slots__ class property.
    - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
    - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
    - joinpoint matching with function or regex.
    - distributed programming:
        - interception context sharing in order to ease behaviour sharing between advices.
        - uuid for advice identification in order to ease its use in a distributed context.
    - maintenable with well named variables and functions, comments and few lines.
    - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
    - respect of aspects vocabulary in order to ease its use among AOP users.
    - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
    - advices are callable objects.
    - Unit tests for all functions such as examples.
4. Benchmark:
    - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let's get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
    - less memory consumption in using the __slots__ class property.
    - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
    - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
    - joinpoint matching with function or regex.
    - distributed programming:
        - interception context sharing in order to ease behaviour sharing between advices.
        - uuid for advice identification in order to ease its use in a distributed context.
    - maintenable with well named variables and functions, comments and few lines.
    - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
    - respect of aspects vocabulary in order to ease its use among AOP users.
    - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
    - advices are callable objects.
    - Unit tests for all functions such as examples.
4. Benchmark:
    - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let’s get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
    - less memory consumption in using the __slots__ class property.
    - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
    - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
    - joinpoint matching with function or regex.
    - distributed programming:

    - interception context sharing in order to ease behaviour sharing between advices.
    - uuid for advice identification in order to ease its use in a distributed context.
  - maintenable with well named variables and functions, comments and few lines.
  - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
  - respect of aspects vocabulary in order to ease its use among AOP users.
  - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
  - advices are callable objects.
  - Unit tests for all functions such as examples.

4. Benchmark:
  - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let’s get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
    - less memory consumption in using the __slots__ class property.
    - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
    - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
    - joinpoint matching with function or regex.
    - distributed programming:
        - interception context sharing in order to ease behaviour sharing between advices.
        - uuid for advice identification in order to ease its use in a distributed context.
    - maintenable with well named variables and functions, comments and few lines.
    - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
    - respect of aspects vocabulary in order to ease its use among AOP users.
    - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
    - advices are callable objects.
    - Unit tests for all functions such as examples.
4. Benchmark:

  - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let's get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)   # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
    - less memory consumption in using the __slots__ class property.
    - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
    - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
    - joinpoint matching with function or regex.
    - distributed programming:
        - interception context sharing in order to ease behaviour sharing between advices.
        - uuid for advice identification in order to ease its use in a distributed context.
    - maintenable with well named variables and functions, comments and few lines.
    - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
    - respect of aspects vocabulary in order to ease its use among AOP users.
    - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
    - advices are callable objects.
    - Unit tests for all functions such as examples.
4. Benchmark:
    - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let's get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
    - less memory consumption in using the __slots__ class property.
    - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
    - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
    - joinpoint matching with function or regex.
    - distributed programming:
        - interception context sharing in order to ease behaviour sharing between advices.
        - uuid for advice identification in order to ease its use in a distributed context.
    - maintenable with well named variables and functions, comments and few lines.
    - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
    - respect of aspects vocabulary in order to ease its use among AOP users.
    - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
    - advices are callable objects.
    - Unit tests for all functions such as examples.
4. Benchmark:
    - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let’s get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
    - less memory consumption in using the __slots__ class property.
    - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
    - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
    - joinpoint matching with function or regex.
    - distributed programming:

- interception context sharing in order to ease behaviour sharing between advices.
- uuid for advice identification in order to ease its use in a distributed context.

- maintenable with well named variables and functions, comments and few lines.
- extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
- respect of aspects vocabulary in order to ease its use among AOP users.
- close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
- advices are callable objects.
- Unit tests for all functions such as examples.

4. Benchmark:
    - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let’s get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
    - less memory consumption in using the __slots__ class property.
    - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
    - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
    - joinpoint matching with function or regex.
    - distributed programming:
        - interception context sharing in order to ease behaviour sharing between advices.
        - uuid for advice identification in order to ease its use in a distributed context.
    - maintenable with well named variables and functions, comments and few lines.
    - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
    - respect of aspects vocabulary in order to ease its use among AOP users.
    - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
    - advices are callable objects.
    - Unit tests for all functions such as examples.
4. Benchmark:

- speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let's get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)   # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description

This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation

pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
    - less memory consumption in using the __slots__ class property.
    - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
    - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
    - joinpoint matching with function or regex.
    - distributed programming:
        - interception context sharing in order to ease behaviour sharing between advices.
        - uuid for advice identification in order to ease its use in a distributed context.
    - maintenable with well named variables and functions, comments and few lines.
    - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
    - respect of aspects vocabulary in order to ease its use among AOP users.
    - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
    - advices are callable objects.
    - Unit tests for all functions such as examples.
4. Benchmark:
    - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples

How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let's get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art

Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

Indices and tables

- genindex
- modindex
- search

Description

This project is an Aspect Oriented Programming library for python with reflective concerns.

Links

- Homepage
- PyPI
- Documentation

Installation

pip install b3j0f.aop

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
    - less memory consumption in using the __slots__ class property.
    - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
    - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
    - joinpoint matching with function or regex.
    - distributed programming:
        - interception context sharing in order to ease behaviour sharing between advices.
        - uuid for advice identification in order to ease its use in a distributed context.
    - maintenable with well named variables and functions, comments and few lines.
    - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
    - respect of aspects vocabulary in order to ease its use among AOP users.
    - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
    - advices are callable objects.
    - Unit tests for all functions such as examples.
4. Benchmark:
    - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples

How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let's get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art

Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.

- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

Donation

1.1.2 Changelog

0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

0.7.7 (13/06/2015)

- fix references shields.io badges.

0.7.6 (13/06/2015)

- use shields.io badge.

0.7.5 (02/06/2015)

- update README with a better state of the art.

0.7.4 (20/05/2015)

- add wheel package.

1.1.3 Indices and tables

- genindex
- modindex
- search

1.1.4 Description

This project is an Aspect Oriented Programming library for python with reflective concerns.

1.1.5 Links

- Homepage
- PyPI
- Documentation

1.1.6 Installation

pip install b3j0f.aop

1.1.7 Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
    - less memory consumption in using the __slots__ class property.
    - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
    - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
    - joinpoint matching with function or regex.
    - distributed programming:
        - interception context sharing in order to ease behaviour sharing between advices.
        - uuid for advice identification in order to ease its use in a distributed context.
    - maintenable with well named variables and functions, comments and few lines.
    - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
    - respect of aspects vocabulary in order to ease its use among AOP users.
    - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
    - advices are callable objects.
    - Unit tests for all functions such as examples.
4. Benchmark:
    - speed execution

1.1.8 Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

1.1.9 Examples

How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let’s get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

1.1.10 State of the art

Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.

- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

1.1.11 Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

1.1.12 Donation

1.2 Changelog

1.2.1 0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

1.2.2 0.7.7 (13/06/2015)

- fix references shields.io badges.

1.2.3 0.7.6 (13/06/2015)

- use shields.io badge.

1.2.4 0.7.5 (02/06/2015)

- update README with a better state of the art.

1.2.5 0.7.4 (20/05/2015)

- add wheel package.

1.3 Indices and tables

- genindex
- modindex
- search

1.4 Description

This project is an Aspect Oriented Programming library for python with reflective concerns.

1.5 Links

- Homepage
- PyPI
- Documentation

1.6 Installation

pip install b3j0f.aop

1.7 Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
    - less memory consumption in using the __slots__ class property.
    - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
    - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
    - joinpoint matching with function or regex.
    - distributed programming:
        - interception context sharing in order to ease behaviour sharing between advices.
        - uuid for advice identification in order to ease its use in a distributed context.
    - maintenable with well named variables and functions, comments and few lines.
    - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
    - respect of aspects vocabulary in order to ease its use among AOP users.

   - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
   - advices are callable objects.
   - Unit tests for all functions such as examples.

4. Benchmark:
   - speed execution

1.8 Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

1.9 Examples

How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let's get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

1.10 State of the art

Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

1.10.1 pyaspects

weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

1.10.2 aspects

weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

1.10.3 aspect

strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

1.10.4 spring

strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

1.10.5 pytilities

strenghts

- Very complex and full library for doing aspects and other things.

weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

1.11 Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

1.12 Donation

CHAPTER 2

Changelog

2.1 0.7.8 (14/06/2015)

- resolve documentation hosted by readthedocs.

2.2 0.7.7 (13/06/2015)

- fix references shields.io badges.

2.3 0.7.6 (13/06/2015)

- use shields.io badge.

2.4 0.7.5 (02/06/2015)

- update README with a better state of the art.

2.5 0.7.4 (20/05/2015)

- add wheel package.

CHAPTER 3

Indices and tables

- genindex
- modindex
- search

CHAPTER 4

Description

This project is an Aspect Oriented Programming library for python with reflective concerns.

CHAPTER 5

Links

- Homepage
- PyPI
- Documentation

CHAPTER 6

Installation

pip install b3j0f.aop

CHAPTER 7

Features

1. Free and unlimited access: no limits to sharing of ideas and knowledges with the license MIT.
2. Performance:
    - less memory consumption in using the __slots__ class property.
    - less time on (un-)weaving and advice application improvement with binary python encoding and in using constants var in code.
    - (dis/en)abling advices without remove them in using dedicated Advice class.
3. Easy to use:
    - joinpoint matching with function or regex.
    - distributed programming:
        - interception context sharing in order to ease behaviour sharing between advices.
        - uuid for advice identification in order to ease its use in a distributed context.
    - maintenable with well named variables and functions, comments and few lines.
    - extensible through pythonic code (PEP8), same logic to function code interception and concern modularisation with one module by joinpoint or advice.
    - respect of aspects vocabulary in order to ease its use among AOP users.
    - close to callable python objects in weaving all types of callable elements such as (built-in) functions, (built-in) class, (built-in) methods, callable objects, etc.
    - advices are callable objects.
    - Unit tests for all functions such as examples.
4. Benchmark:
    - speed execution

Limitations

- Do not weave advices on readonly instance methods (where class use __slots__ attribute).

Examples

How to change the behaviour of min by max ?

```
>>> from b3j0f.aop import weave, is_intercepted
>>> double_advice = lambda joinpoint: joinpoint.proceed() * 2
>>> weave(target=min, advices=double_advice)
>>> min(6, 7)
12
```

How to check if a function is intercepted ?

```
>>> from b3j0f.aop import is_intercepted
>>> is_intercepted(min)
True
```

Ok, let's get back its previous behaviour ...

```
>>> from b3j0f.aop import unweave
>>> unweave(min)
>>> min(6, 7)
6
>>> is_intercepted(min)
False
```

And with an annotation ?

```
>>> from b3j0f.aop import weave_on
>>> weave_on(advices=double_advice)(min)
>>> min(6, 7)
12
>>> is_intercepted(min)
True
>>> unweave(min)  # do not forget to unweave if weaving is useless ;)
```

Enjoy ...

State of the art

Related to improving criteria points (1. Free and unlimited access, etc.), a state of the art is provided here.

Library	Url	License	Execution	Use	Benchmark	Compatibility
b3j0f.aop	https://github.com/b3j0f/aop	MIT	4/5	4/5	4/5	4/5 (>=2.6)
pyaspects	http://tinyurl.com/n7ccof5	GPL 2	4/5	2/5	2/5	2/5
aspects	http://tinyurl.com/obp8t2v	LGPL 2.1	2/5	2/5	2/5	2/5
aspect	http://tinyurl.com/lpd87bd	BSD	2/5	1/5	1/5	1/5
spring	http://tinyurl.com/dmkpj3	Apache	4/5	2/5	3/5	2/5
pytilities	http://tinyurl.com/q49ulr5	GPL 3	1/5	1/5	1/5	1/5

10.1 pyaspects

10.1.1 weaknesses

- Not functional approach: Aspect class definition.
- Side effects: Not close to python API.
- Not optimized Weaving and Time execution: use classes and generic methods.
- Not maintenable: poor comments.
- open-source and use limitations: GPL 2.
- limited in weave filtering.

10.2 aspects

10.2.1 weaknesses

- open-source and use limitations: LGPL 2.1.
- more difficulties to understand code with no respect of the AOP vocabulary, packaged into one module.
- limited in weave filtering.

10.3 aspect

10.3.1 strengths

- invert the AOP in decorating advices with joinpoint instead of weaving advices on joinpoint.
- open-source and no use limitations: BSD.

10.3.2 weaknesses

- Simple and functional approach with use of python tools.
- maintenable: commented in respect of the PEP8.
- limited in weave filtering.

10.4 spring

10.4.1 strengths

- a very powerful library dedicated to develop strong systems based on component based software engineering.
- unittests.
- huge community.

10.4.2 weaknesses

- require to understand a lot of concepts and install an heavy library before doing a simple interception with AOP concerns.

10.5 pytilities

10.5.1 strenghts

- Very complex and full library for doing aspects and other things.

10.5.2 weaknesses

- open-source and use limitations: GPL 3.
- not maintenable: missing documentations and not respect of the PEP8.
- Executon time is not optimized with several classes used with generic getters without using __slots__. The only one optimization comes from the yield which requires from users to use it in their own advices (which must be a class).

CHAPTER 11

Perspectives

- wait feedbacks during 6 months before passing it to a stable version.
- Cython implementation.

CHAPTER 12

Donation