Pylama Documentation

Release 7.6.6

Kirill Klenov

Nov 09, 2018

Contents

Welcome to Pylama's documentation.

Code audit tool for Python and JavaScript. Pylama wraps these tools:

- pycodestyle (formerly pep8) © 2012-2013, Florent Xicluna;
- pydocstyle (formerly pep257 by Vladimir Keleshev) © 2014, Amir Rachum;
- PyFlakes © 2005-2013, Kevin Watters;
- Mccabe © Ned Batchelder;
- Pylint © 2013, Logilab (should be installed 'pylama_pylint' module);
- Radon © Michele Lacchia
- gjslint © The Closure Linter Authors (should be installed 'pylama_gjslint' module);
- eradicate © Steven Myint;

Contents

CHAPTER 1

Requirements:

- Python (2.7, 3.4, 3.5, 3.6, 3.7)
- To use JavaScript checker (`gjslint`) you need to install `python-gflags` with `pip install python-gflags`.
- If your tests are failing on Win platform you are missing: `curses` - http://www.lfd.uci.edu/~gohlke/pythonlibs/ (The curses library supplies a terminal-independent screen-painting and keyboard-handling facility for text-based terminals)

CHAPTER 2

Installation:

Pylama could be installed using pip: ::

```
$ pip install pylama
```

CHAPTER 3

Quickstart

Pylama is easy to use and really fun for checking code quality. Just run *pylama* and get common output from all pylama plugins (pycodestyle, PyFlakes and etc)

Recursive check the current directory.

```
$ pylama
```

Recursive check a path.

```
$ pylama <path_to_directory_or_file>
```

Ignore errors

```
$ pylama -i W,E501
```

Note: You could choose a group erros *D*,'E1' and etc or special errors *C0312*

Choose code checkers

```
$ pylama -l "pycodestyle,mccabe"
```

Choose code checkers for JavaScript:

```
$ pylama --linters=gjslint --ignore=E:0010 <path_to_directory_or_file>
```

CHAPTER 4

Set Pylama (checkers) options

4.1 Command line options

```
$ pylama --help

usage: pylama [-h] [--verbose] [--version]
              [--format {pep8,pycodestyle,pylint,parsable}] [--select SELECT]
              [--sort SORT] [--linters LINTERS] [--ignore IGNORE]
              [--skip SKIP] [--report REPORT] [--hook] [--concurrent]
              [--options FILE] [--force] [--abspath]
              [paths [paths ...]]

Code audit tool for python.

positional arguments:
  paths                 Paths to files or directories for code check.

optional arguments:
  -h, --help            show this help message and exit
  --verbose, -v         Verbose mode.
  --version             show program's version number and exit
  --format {pep8,pycodestyle,pylint,parsable}, -f {pep8,pycodestyle,pylint,parsable}
                        Choose errors format (pycodestyle, pylint, parsable).
  --select SELECT, -s SELECT
                        Select errors and warnings. (comma-separated list)
  --sort SORT           Sort result by error types. Ex. E,W,D
  --linters LINTERS, -l LINTERS
                        Select linters. (comma-separated). Choices are mccabe,
                        pep257,pydocstyle,pep8,pycodestyle,pyflakes,pylint,iso
                        rt.
  --ignore IGNORE, -i IGNORE
                        Ignore errors and warnings. (comma-separated)
  --skip SKIP           Skip files by masks (comma-separated, Ex.
                        */messages.py)
```

(continues on next page)

(continued from previous page)

```
  --report REPORT, -r REPORT
                        Send report to file [REPORT]
  --hook                Install Git (Mercurial) hook.
  --concurrent, --async
                        Enable async mode. Useful for checking a lot of files.
                        Unsupported with pylint.
  --options FILE, -o FILE
                        Specify configuration file. Looks for pylama.ini,
                        setup.cfg, tox.ini, or pytest.ini in the current
                        directory (default: None).
  --force, -F           Force code checking (if linter doesn't allow)
  --abspath, -a         Use absolute paths in output.
```

4.2 File modelines

You can set options for **Pylama** inside a source file. Use pylama *modeline* for this.

Format:

```
# pylama:{name1}={value1}:{name2}={value2}:...
```

```
.. Somethere in code
# pylama:ignore=W:select=W301
```

Disable code checking for current file:

```
.. Somethere in code
# pylama:skip=1
```

Those options have a higher priority.

4.3 Skip lines (noqa)

Just add *# noqa* in end of line to ignore.

```
def urgent_fuction():
    unused_var = 'No errors here' # noqa
```

4.4 Configuration file

Pylama looks for a configuration file in the current directory.

The program searches for the first matching ini-style configuration file in the directories of command line argument. Pylama looks for the configuration in this order:

```
pylama.ini
setup.cfg
tox.ini
pytest.ini
```

The "–option" / "-o" argument can be used to specify a configuration file.

Pylama searches for sections whose names start with *pylama*.

The "pylama" section configures global options like *linters* and *skip*.

```
[pylama]
format = pylint
skip = */.tox/*,*/.env/*
linters = pylint,mccabe
ignore = F0401,C0111,E731
```

4.5 Set Code-checkers' options

You could set options for special code checker with pylama configurations.

```
[pylama:pyflakes]
builtins = _

[pylama:pycodestyle]
max_line_length = 100

[pylama:pylint]
max_line_length = 100
disable = R
```

See code-checkers' documentation for more info.

4.6 Set options for file (group of files)

You could set options for special file (group of files) with sections:

The options have a higher priority than in the *pylama* section.

```
[pylama:*/pylama/main.py]
ignore = C901,R0914,W0212
select = R

[pylama:*/tests.py]
ignore = C0110

[pylama:*/setup.py]
skip = 1
```

CHAPTER 5

Pytest integration

Pylama has Pytest support. The package automatically registers itself as a pytest plugin during installation. Pylama also supports *pytest_cache* plugin.

Check files with pylama

```
pytest --pylama ...
```

Recommended way to set pylama options when using pytest — configuration files (see below).

CHAPTER 6

Writing a linter

You can write a custom extension for Pylama. Custom linter should be a python module. Name should be like 'pylama_<name>'.

In 'setup.py', 'pylama.linter' entry point should be defined.

```
setup(
    # ...
    entry_points={
        'pylama.linter': ['linternname = pylama_linternname.main:Linter'],
    }
    # ...
)
```

'Linter' should be instance of 'pylama.lint.Linter' class. Must implement two methods:

'allow' takes a path and returns true if linter can check this file for errors. 'run' takes a path and meta keywords params and returns a list of errors.

6.1 Example:

Just a virtual 'WOW' checker.

setup.py:

```
setup(
    name='pylama_wow',
    install_requires=[ 'setuptools' ],
    entry_points={
        'pylama.linter': ['wow = pylama_wow.main:Linter'],
    }
    # ...
)
```

pylama_wow.py:

```
from pylama.lint import Linter as BaseLinter

class Linter(BaseLinter):

    def allow(self, path):
        return 'wow' in path

    def run(self, path, **meta):
        with open(path) as f:
            if 'wow' in f.read():
                return [{
                    lnum: 0,
                    col: 0,
                    text: 'Wow has been finded.',
                    type: 'WOW'
                }]
```

6.2 Run pylama from python code

```
from pylama.main import check_path, parse_options

# Use and/or modify 0 or more of the options defined as keys in the variable my_
↪redefined_options below.
# To use defaults for any option, remove that key completely.
my_redefined_options = {
    'linters': ['pep257', 'pydocstyle', 'pycodestyle', 'pyflakes' ...],
    'ignore': ['D203', 'D213', 'D406', 'D407', 'D413' ...],
    'select': ['R1705' ...],
    'sort': 'F,E,W,C,D,...',
    'skip': '*__init__.py,*/test/*.py,...',
    'async': True,
    'force': True
    ...
}
# relative path of the directory in which pylama should check
my_path = '...'

options = parse_options([my_path], **my_redefined_options)
errors = check_path(options, rootdir='.')
```

6.3 Bug tracker

If you have any suggestions, bug reports or annoyances please report them to the issue tracker at https://github.com/klen/pylama/issues

6.4 Contributing

Development of *pylama* happens at GitHub: https://github.com/klen/pylama

6.4.1 Contributors

See AUTHORS.

6.5 License

Licensed under a BSD license.