

---

# **MiniBench Documentation**

*Release 0.1.3.dev*

**Axel Haustant**

November 21, 2015



<b>1</b>	<b>Quickstart</b>	<b>3</b>
<b>2</b>	<b>Writing benchmarks</b>	<b>5</b>
2.1	Basics . . . . .	5
2.2	Documenting . . . . .	5
2.3	Hooks . . . . .	6
<b>3</b>	<b>Console Client</b>	<b>9</b>
3.1	Override Times . . . . .	9
3.2	Export reports . . . . .	9
3.3	Run against a reference . . . . .	9
3.4	Debug mode . . . . .	10
3.5	Bash completion . . . . .	10
<b>4</b>	<b>Reporters</b>	<b>11</b>
<b>5</b>	<b>API</b>	<b>13</b>
5.1	Core . . . . .	13
5.2	Reporters . . . . .	14
<b>6</b>	<b>Changelog</b>	<b>17</b>
6.1	Current . . . . .	17
6.2	0.1.2 (2015-11-21) . . . . .	17
6.3	0.1.1 . . . . .	17
6.4	0.1.0 . . . . .	17
<b>7</b>	<b>Contributing</b>	<b>19</b>
7.1	Submitting issues . . . . .	19
7.2	Submitting patches (bugfix, features, ...) . . . . .	19
<b>8</b>	<b>Indices and tables</b>	<b>21</b>
	<b>Python Module Index</b>	<b>23</b>



MiniBench provides a simple framework for benchmarking following the `unittest` module pattern.

You can install `minibench` with `pip`:

```
$ pip install minibench
```

Then, you just have to write a `.bench.py` file.

```
# fake.bench.py
from minibench import Benchmark

class FakeBenchmark(Benchmark):
    '''Fake benchmark'''
    def bench_fake(self):
        '''Run my bench'''
        # Do something
```

Then run it with the `bench` command

```
$ bench
>>> Fake benchmark (x5)
Run my bench ..... (0.1234s)
```

Contents:



---

**Quickstart**

---

You can install minibench with pip:

```
$ pip install minibench
```

Write your benchmarks as you would write your unittests. Just create `.bench.py` file.

```
# fake.bench.py
from minibench import Benchmark

class FakeBenchmark(Benchmark):
    '''Fake benchmark'''
    def bench_fake(self):
        '''Run my bench'''
        # Do something

    def bench_another(self):
        '''Another bench'''
        # Do something
```

Then run it with the `bench` command

```
$ bench
>>> Fake benchmark (x5)
Run my bench ..... (0.1234s / 0.1233s)
Another bench ..... (0.1234s / 0.1233s)
```





---

## Writing benchmarks

---

### 2.1 Basics

Writing a benchmark is as simple as extending `Benchmark`. Each method will be run `times` times. A benchmark method should start with `bench_`.

```
from minibench import Benchmark

class SumBenchmark(Benchmark):
    times = 1000

    def bench_sum(self):
        sum(x for x in range(5))

    def bench_consecutive_add(self):
        total = 0
        for x in range(5):
            total += x
```

```
$ bench examples/sum.bench.py
Running 1 benchmark
-----
>>> Sum benchmark (x1000)
Consecutive add..... 0.00142s / 0.00000s
Sum..... 0.00245s / 0.00000s
Done
```

### 2.2 Documenting

Documenting your benchmark is as simple as writing explicit docstrings. Only the first line will be kept.

```
from minibench import Benchmark

class SumBenchmark(Benchmark):
    '''
    A simple sum benchmark

    This will be ignored.
```

```
'''
times = 1000

def bench_sum(self):
    '''Sum using sum()'''
    sum(x for x in range(5))

def bench_consecutive_add(self):
    '''Sum using +='''
    total = 0
    for x in range(5):
        total += x
```

```
$ bench examples/sum.bench.py
Running 1 benchmark
-----
>>> A simple sum benchmark (x1000)
Sum using sum()..... 0.00142s / 0.00000s
Sum using +=..... 0.00245s / 0.00000s
Done
```

## 2.3 Hooks

The *Benchmark* provide some hooks as methods:

- *before\_class()*: executed once before each class
- *before()*: executed once before each method
- *before\_each()*: executed before each method call
- *after\_class()*: executed once after each class
- *after()*: executed once after each method
- *after\_each()*: executed after each method call

```
from minibench import Benchmark

class MyBench(Benchmark):

    def before_class(self):
        # Will be executed once before all class methods
        pass

    def before(self):
        # Will be executed once before each method
        pass

    def before_each(self):
        # Will be executed before each method call
        pass

    def after_class(self):
        # Will be executed once after all class methods
        pass

    def after(self):
```

```
# Will be executed once after each method  
pass  
  
def after_each(self):  
# Will be executed after each method call  
pass
```



---

## Console Client

---

MiniBench comes with a console client *bench*.

This client takes a list of glob patterns as parameters. If none is provided, it default on `**/*.bench.py`.

```
$ bench
$ bench benchmarks/*.bench.py
$ bench benches/*.bench tests/*.bench
```

### 3.1 Override Times

You can overrides how many times a methods are called with the `--times` option

```
$ bench -t 10000
$ bench --times 1000
```

### 3.2 Export reports

You can export the result summary to the following formats:

- JSON
- CSV
- reStructuredText
- Markdown

```
$ bench --json out.json --csv out.csv
$ bench --rst out.rst --md out.md
```

### 3.3 Run against a reference

MiniBench provides an easy to compare results against a previous JSON report with the `--ref` option.

```
$ bench --json out.json -t 100
Running 1 benchmark
-----
>>> Glob benchmark (x100)
```

```
Fnmatch..... 1.61257s / 0.01613s
Glob..... 2.02383s / 0.02024s
Re..... 1.39118s / 0.01391s
Done

$ bench --ref out.json -t 100
Running 1 benchmark
-----
>>> Glob benchmark (x100)
Fnmatch..... 1.60748s / 0.01607s (-0.31%)
Glob..... 1.97594s / 0.01976s (-2.36%)
Re..... 1.48161s / 0.01482s (+6.50%)
Done

$ bench --ref out.json -t 100 --unit seconds
Running 1 benchmark
-----
>>> Glob benchmark (x100)
Fnmatch..... 1.60748s / 0.01607s (-0.00508s / -0.00005s)
Glob..... 1.97594s / 0.01976s (-0.04788s / -0.00048s)
Re..... 1.48161s / 0.01482s (+0.09043s / +0.00090s)
Done
```

## 3.4 Debug mode

By default MiniBench does not stop on error nor display it. If you want to stop on first error and display it, you should run in debug mode with the `-d` or the `--debug` option.

```
$ bench -d
$ bench --debug
```

In debug mode, it will not continue to execute a failing method and it will display the raised error.

## 3.5 Bash completion

A bash completion script is provided in the minibench github repository: [bench-complete.sh](#).

---

## Reporters

---

Reporters provide a simple way of formatting your benchmark reports. They all inherit from *BaseReporter*.





---

This part of the documentation lists the full API reference of all public classes and functions.

## 5.1 Core

**class** `minibench.Benchmark` (*times=None, prefix=u'bench\_', debug=False, before=None, before\_each=None, after=None, after\_each=None, \*\*kwargs*)

Base class for all benchmark suites

**times** = The number of iteration to run each method

**after** ()

Hook called once after each method

**after\_class** ()

Hook called after each class

**after\_each** ()

Hook called after each method once

**before** ()

Hook called once before each method

**before\_class** ()

Hook called before each class

**before\_each** ()

Hook called before each method

**label**

A human readable label

**label\_for** (*name*)

Get a human readable label for a method given its name

**run** ()

Collect all tests to run and run them.

Each method will be run `Benchmark.times`.

**class** `minibench.RunResult` (*duration, success, result*)

Store a single method execution result

**class** `minibench.BenchmarkRunner` (*\*filenames, \*\*kwargs*)

Collect all benchmarks and run them

### Parameters

- **filenames** (*string*) – the benchmark files names
- **reporters** (*list*) – the reporters classes or instance to run
- **debug** (*bool*) – Run in debug mode if `True`

**load\_from\_module** (*module*)

Load all benchmarks from a given module

**load\_module** (*filename*)

Load a benchmark module from file

**run** (*\*\*kwargs*)

Run all benchmarks.

Extras kwargs are passed to benchmarks constructors.

## 5.2 Reporters

**class** `minibench.BaseReporter` (*precision=5, \*\*kwargs*)

Base class for all reporters

**after\_class** (*bench*)

Hook called once after each benchmark class

**after\_method** (*bench, method*)

Hook called once after each benchmark method

**before\_class** (*bench*)

Hook called once before each benchmark class

**before\_method** (*bench, method*)

Hook called once before each benchmark method

**end** ()

Hook called once on run end

**key** (*bench*)

Generate a report key from a benchmark instance

**progress** (*bench, method, times*)

Hook called after each benchmark method call

**start** ()

Hook called once on run start

**summary** ()

Compute the execution summary

**class** `minibench.FileReporter` (*filename, \*\*kwargs*)

A reporter dumping results into a file

**Parameters** **filename** (*string*) – the output file name

**end** ()

Dump the report into the output file.

If the file directory does not exist, it will be created. The open file is then given as parameter to `output()`.

**line** (*text=u''*)

A simple helper to write line with ‘ ‘

**output** (*out*)

Serialize the report into the open file.

Child classes should implement this method.

**Parameters** **out** (*file*) – an open file object to serialize into.

**class** `minibench.JsonReporter` (*filename, \*\*kwargs*)

A reporter dumping results into a JSON file

**Parameters** **filename** (*string*) – the output file name

**class** `minibench.CsvReporter` (*filename, \*\*kwargs*)

A reporter dumping results into a CSV file

The CSV will have the following format:

Benchmark	Method	Times	Total (s)	Average (s)
-----------	--------	-------	-----------	-------------

It uses ; character as delimiter and “ as delimiter. All Strings are quoted.

**Parameters** **filename** (*string*) – the output file name

**class** `minibench.MarkdownReporter` (*filename, \*\*kwargs*)

A reporter rendering result as a markdown table.

Each benchmark will be rendered as a table with the following format:

Method	Times	Total (s)	Average (s)
--------	-------	-----------	-------------

**Parameters** **filename** (*string*) – the output file name

**class** `minibench.RstReporter` (*filename, \*\*kwargs*)

A reporter rendering result as a reStructuredText table

Each benchmark will be rendered as a table with the following format:

Method	Times	Total (s)	Average (s)
--------	-------	-----------	-------------

**Parameters** **filename** (*string*) – the output file name

**class** `minibench.cli.CliReporter` (*ref=None, debug=False, unit='%', \*\*kwargs*)

A reporter that display running benchmarks with ANSI colors



---

## Changelog

---

### 6.1 Current

- Nothing yet

### 6.2 0.1.2 (2015-11-21)

- Improve diff from reference and added `-u/--unit` option
- Properly handle precision and added `-p/--precision` option

### 6.3 0.1.1

- Fix Python 3 Wheel packaging

### 6.4 0.1.0

- Initial release



---

## Contributing

---

MiniBench is open-source and very open to contributions.

### 7.1 Submitting issues

Issues are contributions in a way so don't hesitate to submit reports on the [official bugtracker](#).

Provide as much informations as possible to specify the issues:

- the minibench version
- a stacktrace
- installed applications list
- a code sample to reproduce the issue
- ...

### 7.2 Submitting patches (bugfix, features, ...)

If you want to contribute some code:

1. fork the [official minibench repository](#)
2. create a branch with an explicit name (like `my-new-feature` or `issue-XX`)
3. do your work in it
4. rebase it on the master branch from the official repository (cleanup your history by performing an interactive rebase)
5. submit your pull-request

There are some rules to follow:

- your contribution should be documented (if needed)
- your contribution should be tested and the test suite should pass successfully
- your code should be mostly PEP8 compatible with a 120 characters line length
- your contribution should support both Python 2 and 3 (use `tox` to test)

You need to install some dependencies to develop on minibench:

```
$ pip install -e .[test,dev]
```

An `Invoke tasks.py` is provided to simplify the common tasks:

```
$ inv -l
Available tasks:

all          Run tests, reports and packaging
clean       Cleanup all build artifacts
completion  Generate bash completion script
cover       Run tests suite with coverage
dist        Package for distribution
doc         Build the documentation
qa          Run a quality report
test        Run tests suite
tox         Run test in all Python versions
```

To ensure everything is fine before submission, use `tox`. It will run the test suite on all the supported Python version and ensure the documentation is generating.

```
$ tox
```

You also need to ensure your code is compliant with the minibench coding standards:

```
$ inv qa
```



---

## Indices and tables

---

- `genindex`
- `modindex`
- `search`



**m**

minibench, 13



**A**

after() (minibench.Benchmark method), 13  
after\_class() (minibench.BaseReporter method), 14  
after\_class() (minibench.Benchmark method), 13  
after\_each() (minibench.Benchmark method), 13  
after\_method() (minibench.BaseReporter method), 14

**B**

BaseReporter (class in minibench), 14  
before() (minibench.Benchmark method), 13  
before\_class() (minibench.BaseReporter method), 14  
before\_class() (minibench.Benchmark method), 13  
before\_each() (minibench.Benchmark method), 13  
before\_method() (minibench.BaseReporter method), 14  
Benchmark (class in minibench), 13  
BenchmarkRunner (class in minibench), 13

**C**

CliReporter (class in minibench.cli), 15  
CsvReporter (class in minibench), 15

**E**

end() (minibench.BaseReporter method), 14  
end() (minibench.FileReporter method), 14

**F**

FileReporter (class in minibench), 14

**J**

JsonReporter (class in minibench), 15

**K**

key() (minibench.BaseReporter method), 14

**L**

label (minibench.Benchmark attribute), 13  
label\_for() (minibench.Benchmark method), 13  
line() (minibench.FileReporter method), 14  
load\_from\_module() (minibench.BenchmarkRunner method), 14

load\_module() (minibench.BenchmarkRunner method), 14

**M**

MarkdownReporter (class in minibench), 15  
minibench (module), 13

**O**

output() (minibench.FileReporter method), 15

**P**

progress() (minibench.BaseReporter method), 14

**R**

RstReporter (class in minibench), 15  
run() (minibench.Benchmark method), 13  
run() (minibench.BenchmarkRunner method), 14  
RunResult (class in minibench), 13

**S**

start() (minibench.BaseReporter method), 14  
summary() (minibench.BaseReporter method), 14

**T**

times (minibench.Benchmark attribute), 13