

Module: time

```
</> import time (python3 doc)
```

```
</> time.time(): returns the time in seconds since the epoch as a floating point number
```

The *epoch* is OS-dependent, and is given by `time.gmtime(0)`

Module: timeit

TODO

Methodology : Decorator and print()

```
def timing_func(f):
    def wrapper(*args, **kwargs):
        tic = time.time()
        res = f(*args, **kwargs)
        toc = time.time()
        print('{0} running time: {1} secs'.format-
(f.__name__, toc-tic))
        return res
```

Use: Decorator `@timing_func` on functions to be timed, which is syntactic sugar for `some_func = timing_func(some_func)`

Note: To preserve attributes of `some_func`, use `@wraps` decorator from the `functools` module on `wrapper`

Methodology: *nix time command

```
$ /usr/bin/time -p python my_module.py
real 12.37
user 12.15
sys 0.09
```

Note: use system `/usr/bin/time` (man page) rather than shell `time`, as the former comes with a `--verbose` option

real: wall clock or elapsed time

user: amount of time the CPU spent on your task outside of kernel functions

sys: time spent in kernel functions

Useful for: segregating time `my_module.py` spends in CPU, from time spent on other kernel-level tasks, or other background processes

Module: line_profiler

Porting from Python2 to Python3 explained in [this stackoverflow thread](#)

Module: cProfile

C

By [deleted]
cheatography.com/deleted-26304/

Not published yet.
Last updated 11th May, 2016.
Page 1 of 1.

Sponsored by **CrosswordCheats.com**
Learn to solve cryptic crosswords!
<http://crosswordcheats.com>