cProfile

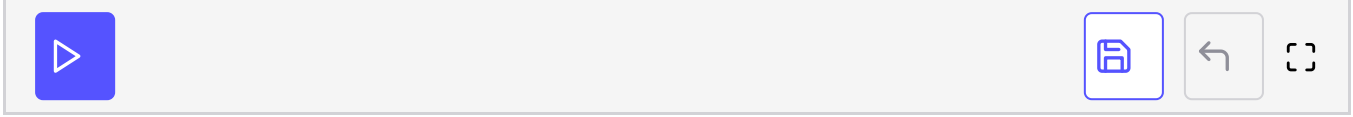
Python comes with its own code profilers built-in. There is the **profile** module and the **cProfile** module. The profile module is pure Python, but it will add a lot of overhead to anything you profile, so it's usually recommended that you go with cProfile, which has a similar interface but is much faster.

We're not going to go into a lot of detail about this module in this chapter, but let's look at a couple of fun examples so you get a taste for what it can do.

```
import cProfile
cProfile.run("[x for x in range(1500)]")
#         4 function calls in 0.001 seconds

#   Ordered by: standard name

#   ncalls  tottime  percall  cumtime  percall filename:lineno(function)
#        1    0.000    0.000    0.000    0.000 <string>:1(<listcomp>)
#        1    0.000    0.000    0.000    0.000 <string>:1(<module>)
#        1    0.001    0.001    0.001    0.001 {built-in method builtins.exec}
#        1    0.000    0.000    0.000    0.000 {method 'disable' of '_lsprof.Profiler' object
```



Let's break this down a bit. The first line shows that there were 4 function calls. The next line tells us how the results are ordered. According to the documentation, standard name refers to the far right column. There are a number of columns here.

> - **ncalls** is the number of calls made.
> - **tottime** is a total of the time spent in the given function.
> - **percall** refers to the quotient of tottime divided by ncalls
> - **cumtime** is the cumulative time spent in this and all subfunctions. It's even accurate for recursive functions!
> - The second **percall** column is the quotient of cumtime divided by

primitive calls

- **filename:lineno(function)** provides the respective data of each function

You can call cProfile on the command line in much the same way as we did with the timeit module. The main difference is that you would pass a Python script to it instead of just passing a snippet. Here's an example call:

```
python -m cProfile test.py
```

Give it a try on one of your own modules or try it on one of Python's modules to see how it works.