# System Development with Python: Week 8

Christopher Barker

UW Continuing Education

May 14, 2013

## Table of Contents

## Performance Testing

"Premature optimization is the root of all evil"

– Donald Knuth

## Profiling/timing

You can't optimize your code without knowing where the bottlenecks are.

Smarter people than me have said they they are almost always wrong when they try to logically determine where the slow code is. (I know I am)

... and how to speed it up

## time.clock()

The really easy way:

```
import time

start = time.clock()
  ... do_some_stuff ...
print "It took %f seconds to run"%(time.clock - start)
```

It works, it's easy, and it gives a gross approximation

(use `time.clock()`, rather than `time.time()`)

## timeit

The good way:

```
import timeit

timeit.timeit( statement, setup=some_stuff)
```

It's kind of a pain, but gives meaningful results.
(can also be called on the command line)

`http://docs.python.org/library/timeit.html`

(`code/timing.py`)

## %timeit

The easy and good way:

`ipython`:

```
In [52]: import timing

In [53]: %timeit timing.primes_stupid(5)
100000 loops, best of 3: 10.9 us per loop
```

Takes care of the setup/namespace stuff for you

`http://ipython.org/ipython-doc/dev/interactive/tutorial.html`

## Profiling

A profiler is a tool that describes the run time performance of a program, providing a variety of statistics

Helpful when you don't yet know where your bottlenecks are

The python profiler

```
python -m cProfile profile_example.py
```

spews some stats

`http://docs.python.org/library/profile.html`

## python profiler

What you get:

| | |
|---|---|
| ncalls | the number of calls. |
| tottime | the total time spent in the given function (and excluding time made in calls to sub-functions), |
| percall | the quotient of tottime divided by ncalls |
| cumtime | the total time spent in this and all subfunctions (from invocation till exit). This figure is accurate even for recursive functions. |
| percall | the quotient of cumtime divided by primitive calls |

(demo: `python -m cProfile profile_example.py`)

## python profiler

You can also dump to a file:

```
$ python -m cProfile -o profile_dump profile_example.py
```

This gives you a binary file you can examine with `pstats`:

demo: `$ python -m pstats`

## pstats

### Running pstats

```
$
$ python -m pstats
Welcome to the profile statistics browser.
% read profile_dump
profile_dump% stats
Wed Aug 29 16:21:39 2012    profile_dump

         51403 function calls in 0.032 seconds

   Random listing order was used

   ncalls  tottime  percall  cumtime  percall filename:lineno(fu
    51200    0.006    0.000    0.006    0.000 {method 'append' o
        1    0.000    0.000    0.032    0.032 profile_example.py
        1    0.001    0.001    0.032    0.032 profile_example.py
      100    0.022    0.000    0.027    0.000 profile_example.py
      100    0.003    0.000    0.031    0.000 profile_example.py
```

## pstats commands

Commands:

| | |
|---|---|
| help | help on pstats or particular command |
| stats | print the profile statistics |
| sort | sort by various data fields |
| strip | strips the leading path info from file names |
| callers | Print callers statistics |
| callees | Print callees statistics |
| quit | quits |

Each has options to customize output

## automating profile stats

`cProfile` and `pstats` are also modules

So you can script collection of profiles and stats

`http://docs.python.org/library/profile.html`

## "Run Snake Run"

For a visual look at your profiling results:

`http://www.vrplumber.com/programming/runsnakerun/`

(pretty cool stuff!)

## Performance Tips

Some common python performance issues:

`http://wiki.python.org/moin/PythonSpeed/PerformanceTips/`

(some nifty profiling tools described there, too)

## LAB

### Profiling lab

- run `timeit` on some code of yours (or timing.py, or..)
- run iPython's %timeit on the same code.
- try to make the factorial code in timing.py faster, and time the difference.
- write some code that tests one of the performance issues in: `http://wiki.python.org/moin/PythonSpeed/PerformanceTips` use one of the `timeits` to see if you can make a difference.
- try the profile tutorial at: `http://pysnippet.blogspot.com/2009/12/profiling-your-python-code.html`

## Wrap up

I hope you have an idea how to profile and time your code.

Try it on a part of your project

## Next Week:

Student Presentations

## Homework

Profile your project

Performance tune part of it

Get ready to present

## Project Time!

Map out what you're going to present