Part 1: Profiling

Bookmark this page

Part 1: Profiling

At some point, perhaps even during this curriculum, you may have heard someone say that comprehensions are fast. In a certain general sense perhaps they are, but in this context — performance — speed should strictly be considered a relative metric. In other words, the question is not whether comprehensions are fast in general, but whether a comprehension is fast relative to some other construct in the specific context of the programming problem you are trying to address.

To carry this thought further, consider some of the alternatives to comprehensions. Do you need to instantiate a list (or dictionary) immediately? If not, perhaps your use case would be better served by map and filter. Map returns a lazy iterator in Python 3.x, which is to say that when it is called it does not instantiate an entire list as does a comprehension. Instead, a map object yields up subsequent values on subsequent calls, thereby spreading computational cost over a larger timespan. Perhaps that is enough to make your graphic user interface or ReST API more responsive. Speaking of yield, would a generator be better than a comprehension for your use case? If you strictly need a sequence generated by an algorithm, it may well be, both in terms of memory consumption and in terms of raw speed.

However, knowing your options in terms of programming constructs is not the last word on understanding the performance of your program. And gaining a view into the performance of your program in isolation does not necessarily translate into an understanding of its performance in a production system incurring high, bursting loads, experiencing network latencies, blocking on calls to a message queue, or retrieving data from a database running on a separate machine.

Keep all of this in mind, because one of the themes of this lesson is that you should be skeptical of what you think you know until you have observed a system in a representative context.

Profiling & Timing

In its broadest sense, profiling is observing and measuring a system, an individual program or a code snippet in a context. The goal is to understand performance and resource consumption characteristics. The context should be representative, and when possible it should avoid the introduction of extraneous artifacts. For instance, when profiling code snippets of alternate algorithms, the same python interpreter and version should be used. When profiling your code on alternate interpreters, a single machine should be used. If a program depends on user input or on data from extraneous systems, mock or simulate these vectors in a problematically reproducible fashion so that multiple runs of the program are not differentially influenced. In short, avoid confounding factors and control for your variables.

At a high level, before resorting to tools specific to Python, operating systems provide performance monitoring tools such as Activity Monitor (OS X), top (Linux & OS X), and Performance Monitor (Windows). These tools can be a good first place to look for memory, processor or network hogs.

Time

GNU Time, or something very close to it, is available on all major operating systems. It measures the duration of processes and can be used to clock virtually any program with a command line interface.

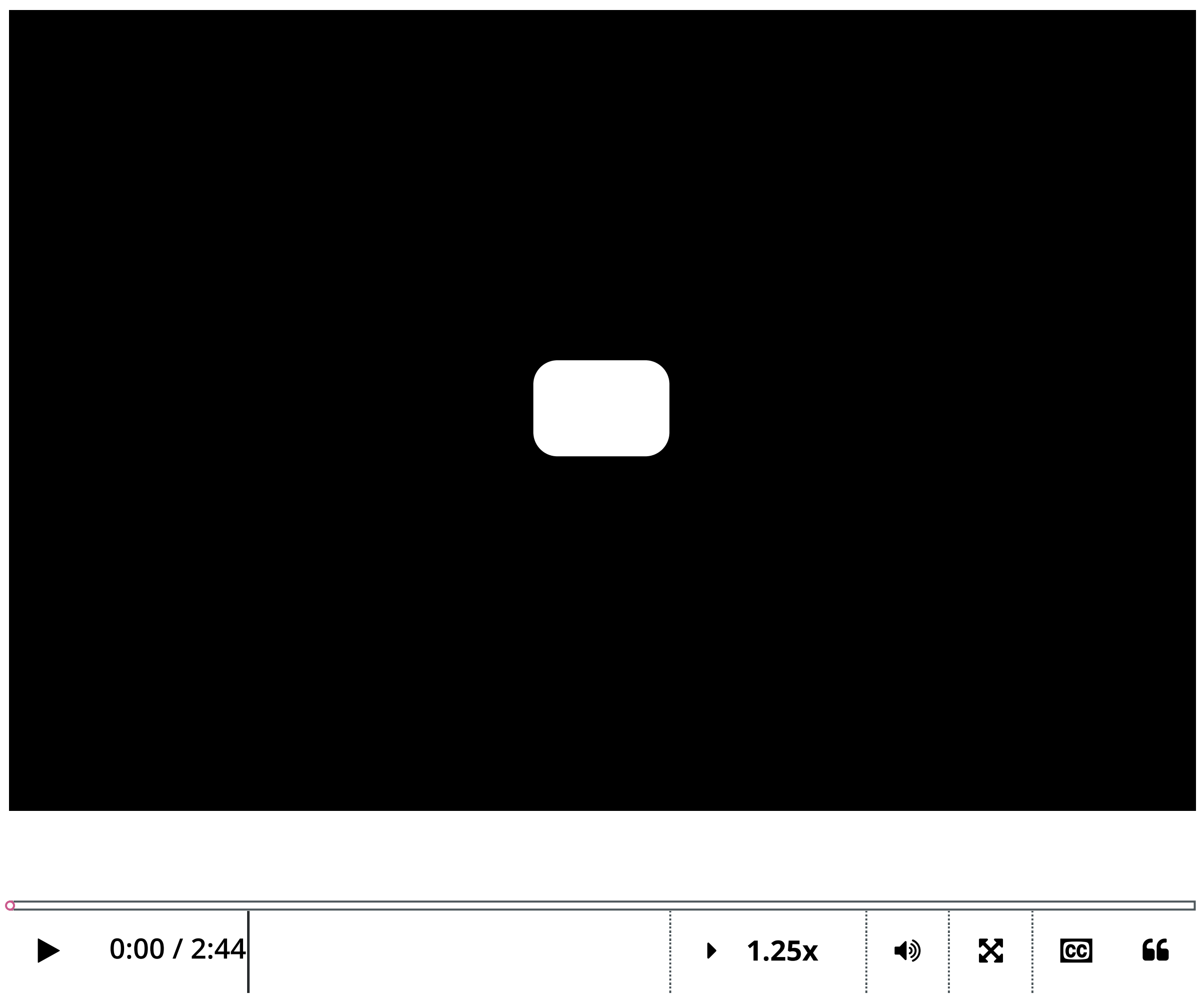
https://www.gnu.org/software/time/

```
$ time -p find ~/Documents
real 3.02
user 0.15
sys 0.62
```

Gnu-time installed via Homebrew on Mac OS X for more depth.

```
$ brew install gnu-time
$ gtime --verbose find ~/Documents
    Command being timed: "find /Users/demo-user/Documents"
    User time (seconds): 0.06
    System time (seconds): 0.27
    Percent of CPU this job got: 62%
    Elapsed (wall clock) time (h:mm:ss or m:ss): 0:00.54
    Average shared text size (kbytes): 0
    Average unshared data size (kbytes): 0
    Average stack size (kbytes): 0
    Average total size (kbytes): 0
    Maximum resident set size (kbytes): 1292
    Average resident set size (kbytes): 0
    Major (requiring I/O) page faults: 0
    Minor (reclaiming a frame) page faults: 449
    Voluntary context switches: 1384
    Involuntary context switches: 1028
    Swaps: 0
    File system inputs: 0
    File system outputs: 0
    Socket messages sent: 0
    Socket messages received: 0
    Signals delivered: 0
    Page size (bytes): 4096
    Exit status: 0
```

Video 1 of 2: Time



Start of transcript. Skip to the end.

[MUSIC PLAYING]

Profiling and performance.

We're going to be looking at techniques to judge the efficiency of your code

and even looking at some of the built-ins

to use them as examples to see how they fare against each other.

These first couple of things that I want to show you are--

you're not even really getting into Python yet.

Video
Download video file

Transcripts
Download SubRip (.srt) file
Download Text (.txt) file

timeit

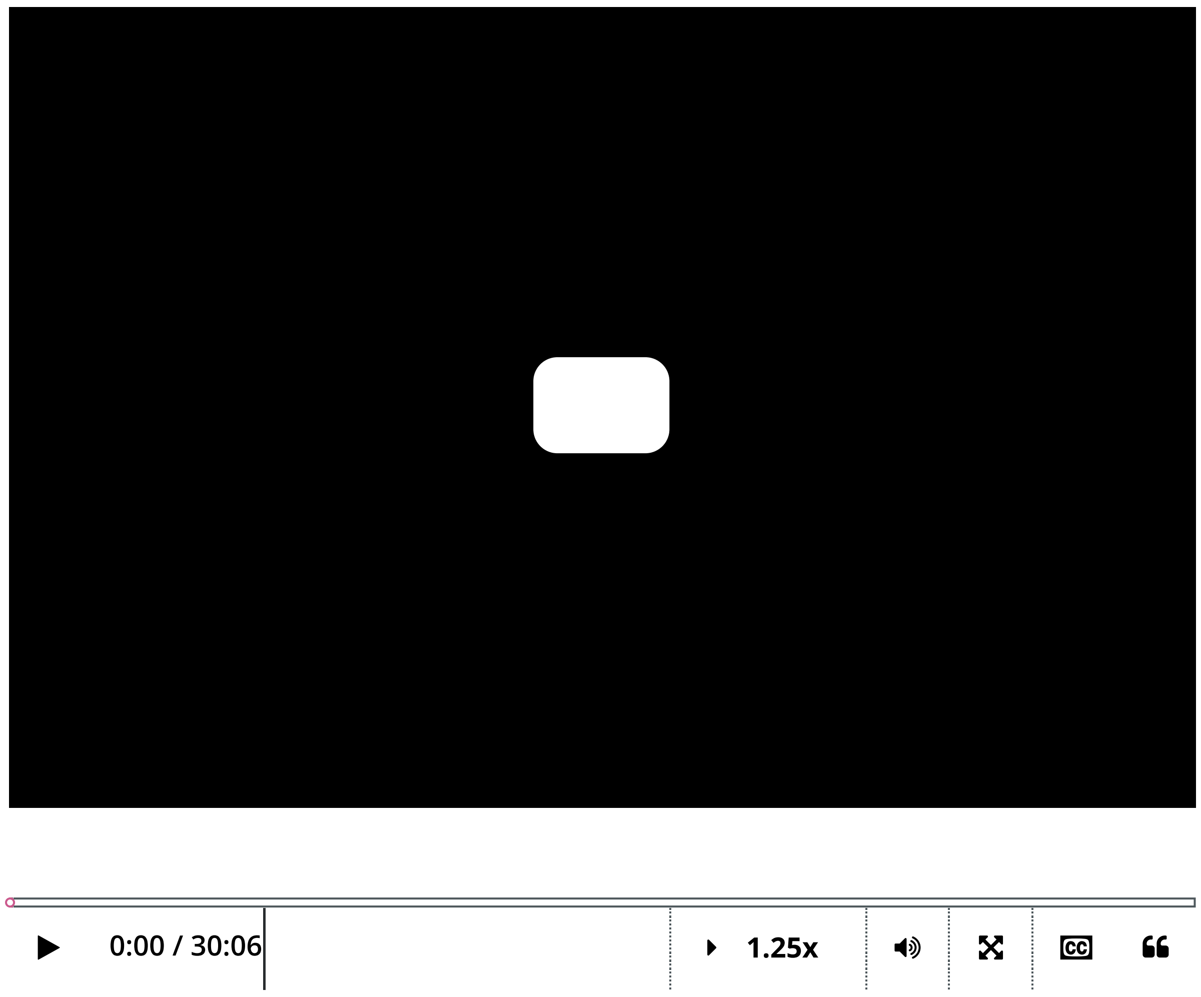
In addition to the command line time tool referenced above which is useful to time the run of your entire script, Python offers timeit which allows you to time expressions and calls within python modules.

https://docs.python.org/3.6/library/timeit.html

Timeit is used extensively in the videos associated with this lesson.

cProfile

Video 2 of 2: Profiling



Start of transcript. Skip to the end.

[MUSIC PLAYING]

OK.

We're going to use timeit to time some constructs with which you're already

familiar.

We're going to look at map and filter and then

we're going to look at doing pretty much the same thing with comprehensions.

And we're going to see how they perform vis a vis each other.

Video

Download video file

Transcripts

Download SubRip (.srt) file

Download Text (.txt) file

The same manner in which Python offers debuggers (pdb, ipdb) it also supplies a profiler. With little or no modification to your module’s source cProfile provides statistics on the number of times a function or method is called and the cumulative time spent within.

The profiler when invoked from the command line with default arguments provides information not only about your script, but also about how it exercises the Python interpreter.

```
re;">$ python -m cProfile great_circle.py
         12000539 function calls (12000538 primitive calls) in 3.179 seconds

   Ordered by: standard name

   ncalls  tottime  percall  cumtime  percall filename:lineno(function)
        2    0.000    0.000    0.000    0.000 <frozen importlib._bootstrap>:103(release)
        2    0.000    0.000    0.000    0.000 <frozen importlib._bootstrap>:143(__init__)
        2    0.000    0.000    0.000    0.000 <frozen importlib._bootstrap>:147(__enter__)
        2    0.000    0.000    0.000    0.000 <frozen importlib._bootstrap>:151(__exit__)
        2    0.000    0.000    0.000    0.000 <frozen importlib._bootstrap>:157(_get_module_lock)
        2    0.000    0.000    0.000    0.000 <frozen importlib._bootstrap>:176(cb)
        3    0.000    0.000    0.000    0.000 <frozen
importlib._bootstrap>:211(_call_with_frames_removed)
       46    0.000    0.000    0.000    0.000 <frozen importlib._bootstrap>:222(_verbose_message)
        2    0.000    0.000    0.000    0.000 <frozen importlib._bootstrap>:307(__init__)
        2    0.000    0.000    0.000    0.000 <frozen importlib._bootstrap>:311(__enter__)
        2    0.000    0.000    0.000    0.000 <frozen importlib._bootstrap>:318(__exit__)
        8    0.000    0.000    0.000    0.000 <frozen importlib._bootstrap>:321(<genexpr>)
        1    0.000    0.000    0.000    0.000 <frozen importlib._bootstrap>:35(_new_module)
        2    0.000    0.000    0.000    0.000 <frozen importlib._bootstrap>:369(__init__)
        3    0.000    0.000    0.000    0.000 <frozen importlib._bootstrap>:403(cached)
        2    0.000    0.000    0.000    0.000 <frozen importlib._bootstrap>:416(parent)
        2    0.000    0.000    0.000    0.000 <frozen importlib._bootstrap>:424(has_location)
        2    0.000    0.000    0.000    0.000 <frozen
importlib._bootstrap>:504(_init_module_attrs)
        2    0.000    0.000    0.000    0.000 <frozen importlib._bootstrap>:564(module_from_spec)
        2    0.000    0.000    0.000    0.000 <frozen importlib._bootstrap>:58(__init__)
        2    0.000    0.000    0.001    0.000 <frozen importlib._bootstrap>:651(_load_unlocked)
        2    0.000    0.000    0.000    0.000 <frozen importlib._bootstrap>:707(find_spec)
        2    0.000    0.000    0.000    0.000 <frozen importlib._bootstrap>:78(acquire)
        2    0.000    0.000    0.000    0.000 <frozen importlib._bootstrap>:780(find_spec)
        6    0.000    0.000    0.000    0.000 <frozen importlib._bootstrap>:843(__enter__)
        6    0.000    0.000    0.000    0.000 <frozen importlib._bootstrap>:847(__exit__)
        2    0.000    0.000    0.000    0.000 <frozen importlib._bootstrap>:870(_find_spec)
        2    0.000    0.000    0.001    0.000 <frozen
importlib._bootstrap>:936(_find_and_load_unlocked)
        2    0.000    0.000    0.001    0.000 <frozen importlib._bootstrap>:966(_find_and_load)
       11    0.000    0.000    0.000    0.000 <frozen
importlib._bootstrap_external>:1080(_path_importer_cache)
        2    0.000    0.000    0.000    0.000 <frozen
importlib._bootstrap_external>:1117(_get_spec)
        2    0.000    0.000    0.000    0.000 <frozen
importlib._bootstrap_external>:1149(find_spec)
        2    0.000    0.000    0.000    0.000 <frozen
importlib._bootstrap_external>:1228(_get_spec)
        9    0.000    0.000    0.000    0.000 <frozen
importlib._bootstrap_external>:1233(find_spec)
        2    0.000    0.000    0.000    0.000 <frozen
importlib._bootstrap_external>:263(cache_from_source)
        2    0.000    0.000    0.000    0.000 <frozen
importlib._bootstrap_external>:361(_get_cached)
        9    0.000    0.000    0.000    0.000 <frozen
importlib._bootstrap_external>:37(_relax_case)
        1    0.000    0.000    0.000    0.000 <frozen
importlib._bootstrap_external>:393(_check_name_wrapper)
        1    0.000    0.000    0.000    0.000 <frozen
importlib._bootstrap_external>:430(_validate_bytecode_header)
        1    0.000    0.000    0.000    0.000 <frozen
importlib._bootstrap_external>:485(_compile_bytecode)
        2    0.000    0.000    0.000    0.000 <frozen importlib._bootstrap_external>:52(_r_long)
        2    0.000    0.000    0.000    0.000 <frozen
importlib._bootstrap_external>:524(spec_from_file_location)
       42    0.000    0.000    0.000    0.000 <frozen
importlib._bootstrap_external>:57(_path_join)
       42    0.000    0.000    0.000    0.000 <frozen
importlib._bootstrap_external>:59(<listcomp>)
        2    0.000    0.000    0.000    0.000 <frozen
importlib._bootstrap_external>:63(_path_split)
        1    0.000    0.000    0.000    0.000 <frozen
importlib._bootstrap_external>:669(create_module)
        1    0.000    0.000    0.000    0.000 <frozen
importlib._bootstrap_external>:672(exec_module)
        1    0.000    0.000    0.000    0.000 <frozen importlib._bootstrap_external>:743(get_code)
       12    0.000    0.000    0.000    0.000 <frozen
importlib._bootstrap_external>:75(_path_stat)
        1    0.000    0.000    0.000    0.000 <frozen importlib._bootstrap_external>:800(__init__)
        1    0.000    0.000    0.000    0.000 <frozen
importlib._bootstrap_external>:825(get_filename)
        1    0.000    0.000    0.000    0.000 <frozen importlib._bootstrap_external>:830(get_data)
        1    0.000    0.000    0.000    0.000 <frozen
importlib._bootstrap_external>:840(path_stats)
        2    0.000    0.000    0.000    0.000 <frozen
importlib._bootstrap_external>:85(_path_is_mode_type)
        1    0.000    0.000    0.000    0.000 <frozen importlib._bootstrap_external>:908(__init__)
        1    0.000    0.000    0.000    0.000 <frozen
importlib._bootstrap_external>:919(create_module)
        1    0.000    0.000    0.000    0.000 <frozen
importlib._bootstrap_external>:927(exec_module)
        2    0.000    0.000    0.000    0.000 <frozen
importlib._bootstrap_external>:94(_path_isfile)
        1    0.000    0.000    0.000    0.000 cProfile.py:27(Profile)
        1    0.000    0.000    0.000    0.000 cProfile.py:5(<module>)
  1000000    0.150    0.000    0.150    0.000 great_circle.py:12(calculate_x)
  2000000    0.216    0.000    0.216    0.000 great_circle.py:16(calculate_coordinate)
  2000000    0.218    0.000    0.218    0.000 great_circle.py:16(calculate_coordinate)
  1000000    0.113    0.000    0.113    0.000 great_circle.py:20(calculate_theta)
  1000000    0.952    0.000    1.500    0.000 great_circle.py:24(calculate_acos)
  1000000    0.958    0.000    2.937    0.000 great_circle.py:28(great_circle_factored)
        1    0.240    0.240    3.178    3.178 great_circle.py:38(main)
        1    0.000    0.000    3.179    3.179 great_circle.py:8(<module>)
        1    0.000    0.000    0.000    0.000 {built-in method _imp._fix_co_filename}
       10    0.000    0.000    0.000    0.000 {built-in method _imp.acquire_lock}
        1    0.000    0.000    0.000    0.000 {built-in method _imp.create_dynamic}
        1    0.000    0.000    0.000    0.000 {built-in method _imp.exec_dynamic}
        2    0.000    0.000    0.000    0.000 {built-in method _imp.is_builtin}
        2    0.000    0.000    0.000    0.000 {built-in method _imp.is_frozen}
       10    0.000    0.000    0.000    0.000 {built-in method _imp.release_lock}
        4    0.000    0.000    0.000    0.000 {built-in method _thread.allocate_lock}
        4    0.000    0.000    0.000    0.000 {built-in method _thread.get_ident}
        1    0.000    0.000    0.000    0.000 {built-in method builtins.__build_class__}
        2    0.000    0.000    0.000    0.000 {built-in method builtins.any}
      2/1    0.000    0.000    3.179    3.179 {built-in method builtins.exec}
       12    0.000    0.000    0.000    0.000 {built-in method builtins.getattr}
       13    0.000    0.000    0.000    0.000 {built-in method builtins.hasattr}
       12    0.000    0.000    0.000    0.000 {built-in method builtins.isinstance}
        4    0.000    0.000    0.000    0.000 {built-in method builtins.len}
        2    0.000    0.000    0.000    0.000 {built-in method from_bytes}
        1    0.000    0.000    0.000    0.000 {built-in method marshal.loads}
  1000000    0.106    0.000    0.106    0.000 {built-in method math.acos}
  3000000    0.261    0.000    0.261    0.000 {built-in method math.cos}
  2000000    0.181    0.000    0.181    0.000 {built-in method math.sin}
        4    0.000    0.000    0.000    0.000 {built-in method posix.fspath}
        4    0.000    0.000    0.000    0.000 {built-in method posix.getcwd}
       12    0.000    0.000    0.000    0.000 {built-in method posix.stat}
        1    0.000    0.000    0.000    0.000 {method 'disable' of '_lsprof.Profiler' objects}
        3    0.000    0.000    0.000    0.000 {method 'endswith' of 'str' objects}
        4    0.000    0.000    0.000    0.000 {method 'get' of 'dict' objects}
       44    0.000    0.000    0.000    0.000 {method 'join' of 'str' objects}
        1    0.000    0.000    0.000    0.000 {method 'read' of '_io.FileIO' objects}
       17    0.000    0.000    0.000    0.000 {method 'rpartition' of 'str' objects}
       86    0.000    0.000    0.000    0.000 {method 'rstrip' of 'str' objects}
```

When invoked from within an interpreter, you can have the profiler be more selective with its reporting.

```
"go">$ ipython
Python 3.6.4 (default, Jan  6 2018, 11:51:59)
Type 'copyright', 'credits' or 'license' for more information
IPython 6.2.1 -- An enhanced Interactive Python. Type '?' for help.

In [1]: %run great_circle.py

In [2]: %prun main()
         12000004 function calls in 3.165 seconds
```