PIC 16, Winter 2018 – Quick Guide to Python Performance

Timing Code Segment Execution
Useful for comparing relative efficiency of multiple approaches to a given task.

A very simple but effective solution:
First, `import time`, then:

```
begin = time.clock() # record start time
# code goes here
end = time.clock()   # record end time
print end-begin      # calculate difference (elapsed time)
```

You might wish to enclose these commands within a `for` loop and save the elapsed times to a list, as execution times can vary considerably. Note: use `time.time()` instead of `time.clock()` on Linux.

Profiling Scripts (execution time)
Useful for finding bottleneck functions within a *script*. (Which *functions* take the most time?)

Open a command prompt (not a Python prompt, an OS command prompt), navigate to the directory of the desired script, and execute the following command:

```
python –m cProfile myprog.py
```

where `myprog.py` is to be replaced by the script name.

Profiling Functions (execution time)
Useful for finding bottleneck statements within *functions*. (Which *commands* take the most time?)

First, you may need to install `line_profiler`. As it is supported by the Anaconda distribution, you can install it at the OS command prompt by entering the command:

```
conda install line_profiler
```

Then, in your script, "decorate" the function you wish to profile by entering `@profile` on the line before the beginning of the function definition. (This will need to be removed if you want to run the script normally in Spyder or Jupyter, but it is needed for profiling.)
Finally, in the OS command prompt, navigate to the directory of the desired script and use the command:

```
kernprof –l –v myprog.py
```

Profiling Functions (memory usage)
Useful for determining memory usage of commands (must be within a function).

First, you may need to install `memory_profiler`. It is *not* part of the Anaconda distribution, but you can install it at the OS command prompt by entering the command:

```
pip install –U memory_profiler
```

Decorate the function you wish to profile as for execution time, then from the command line:

```
python –m memory_profiler myprog.py
```

See Also: https://docs.python.org/2/library/timeit.html, https://docs.python.org/2/library/profile.html, http://tinyurl.com/huyng-python-performance
http://stackoverflow.com/questions/1557571/how-to-get-time-of-a-python-program-execution