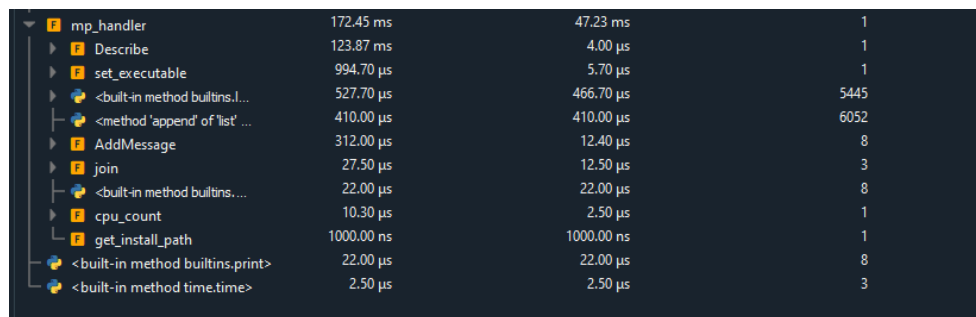


## GEOG 489: Lesson 1 Write Up

During this first lesson I learned about the general ability of Multiprocessing with Python. This lesson has shown me several ways to optimize the code I write where in the past my focus was that it worked, and I would not revisit it to make sure it runs in a timely and efficient manner. I was introduced to the profiler within Spyder, as a tool to look at the tasks within the code and their contributions to run time. My profile results for this assignment came back showing the Describe functions taking the longest at 123 ms within the main method, not accounting for the multiprocessing portion. With some simple math I figured each multiprocessor task too just under .3 seconds. I did this by subtracting the profile time (.172 seconds) from an actual execution (15.5 seconds) and dividing the remainder by the number of jobs  $((15.536 - .172) / 51 = .301)$ . This includes reading the list of results so there is assumed error.



mp_handler	172.45 ms	47.23 ms	1
Describe	123.87 ms	4.00 µs	1
set_executable	994.70 µs	5.70 µs	1
<built-in method builtins.l...	527.70 µs	466.70 µs	5445
<method 'append' of 'list' ...	410.00 µs	410.00 µs	6052
AddMessage	312.00 µs	12.40 µs	8
join	27.50 µs	12.50 µs	3
<built-in method builtins....	22.00 µs	22.00 µs	8
cpu_count	10.30 µs	2.50 µs	1
get_install_path	1000.00 ns	1000.00 ns	1
<built-in method builtins.print>	22.00 µs	22.00 µs	8
<built-in method time.time>	2.50 µs	2.50 µs	3

For the Above and Beyond, I implemented multiple inputs from the ArcPro Script and added a 'Field' selected from the Clipper layer to be added to the output name. For this assignment I used it to assign the state name to the clip, so the output name contained the Roads, Hydro, and Cities by each state.

I wrote a script that attempts to create feature classes in a GDB within the multicode.py, without the bandwidth to trouble shoot it in the next two days; it creates the files but a handful export as tables and a handful are empty feature classes. I attempt to bypass the GDB Lock by running a while loop and an except with `time.sleep(.1 + .01 * random number 1-9)`, I added a counter to break the loop. This way it attempts to perform the clip between other locks, after so many tries it returns false as an error. I also used `edit.start/stop` but it seemed to work without it as well. Without a way to ensure jobs are not double tasked, this would take some cleanup to keep only valid features. I tried several methods of validating the clip before the process ended and moved onto the next, such as running a describe and ensuring the count was not None or < 1.

If I needed this to work tomorrow, I would export each Clip to a temp file/folder on disk and run a script to .append them into a single feature class to be saved into the GDB, making a single lock event.

Hopefully, we cover this in the course.