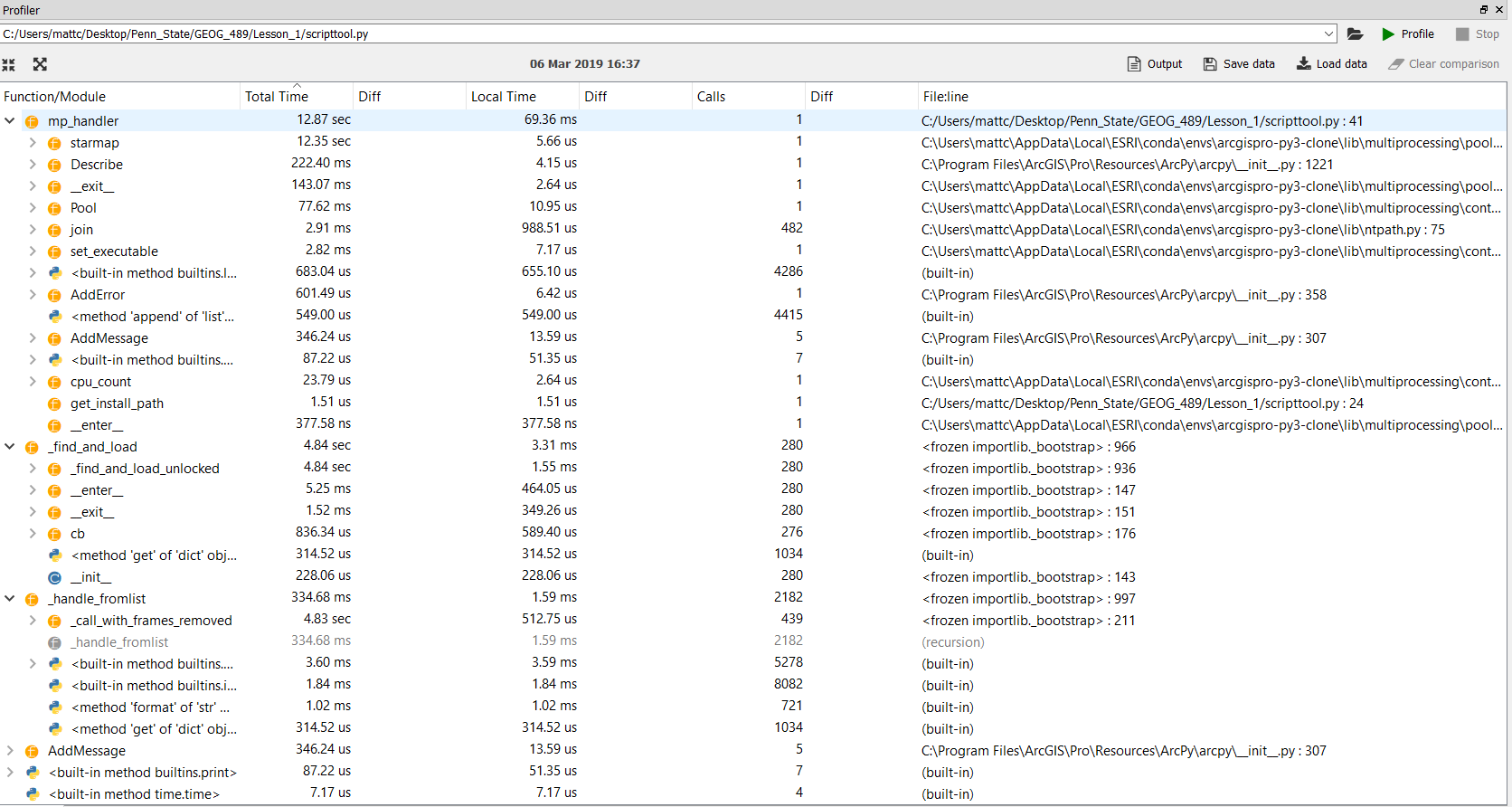
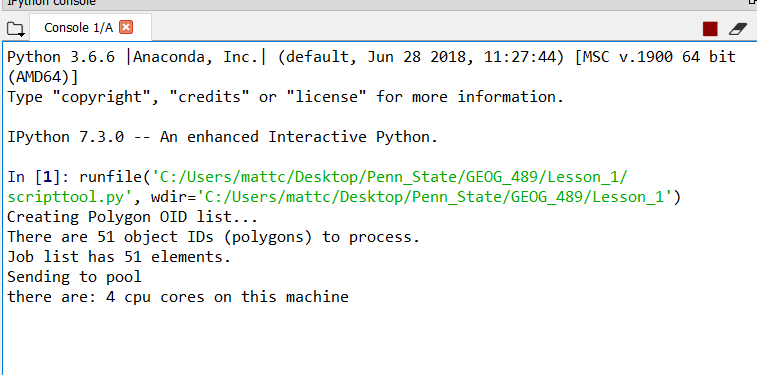
1. To fulfill the requirements of this section I modified the code of scripttool.py at the beginning to receive a third input parameter and assigned it to the variable name “outputPath”. I assigned this variable to my Lesson 1 folder where I have kept the USA.gdb and my Python scripts. I then added the “outputPath” as a fifth parameter value in the *worker* function that is defined in multicode.py so that the *worker* function could accept that value when it runs.
2. For Number 2 in task list I imported the time module and created a variable to store the total computation time of the script. I wrote code at the end of the script that would calculate this processing time and print it. I then used the Profiler window within Spyder and input my script to run a basic code profile. Below is a screenshot of my results, with the Profiler window expanded, and with some of the function/modules expanded to show a few of the subprocesses:



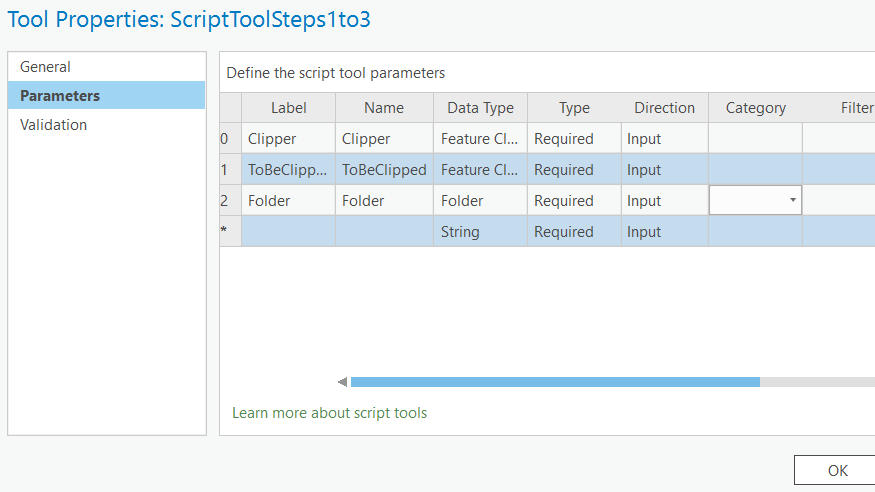
From the results of the Profiler, it appears the time varied a bit depending on Total Time versus Local Time. The most time spent on the entire process regarding Local Time was the “join” function within the mp\_handler module. Some built-in methods were behind the “join” functions time. However with respect to Total Time, the “starmap” function took the most time by far! The time spent executing any part of the rest of the script was significantly less than the time spent running starmap.

Out of curiosity I ran the script from within Spyder and it took a while for it to complete! In fact, I didn’t even wait for it to complete, it was still running after 15 minutes so I shut down the program. Here’s a screenshot of the output from the Run until I shut it down:



1. Number 3 in the task list asks that the code allow the user to enter the parameters, so I used the arcpy.GetParameterAsText() function to prompt the user for their desired value in each of the three input parameters (the clipper file, the to-be-clipped file, and the output path directory), changing it from their previously static assignments. This script tool is within my toolbox that I created for this lesson and is part of my zip file.

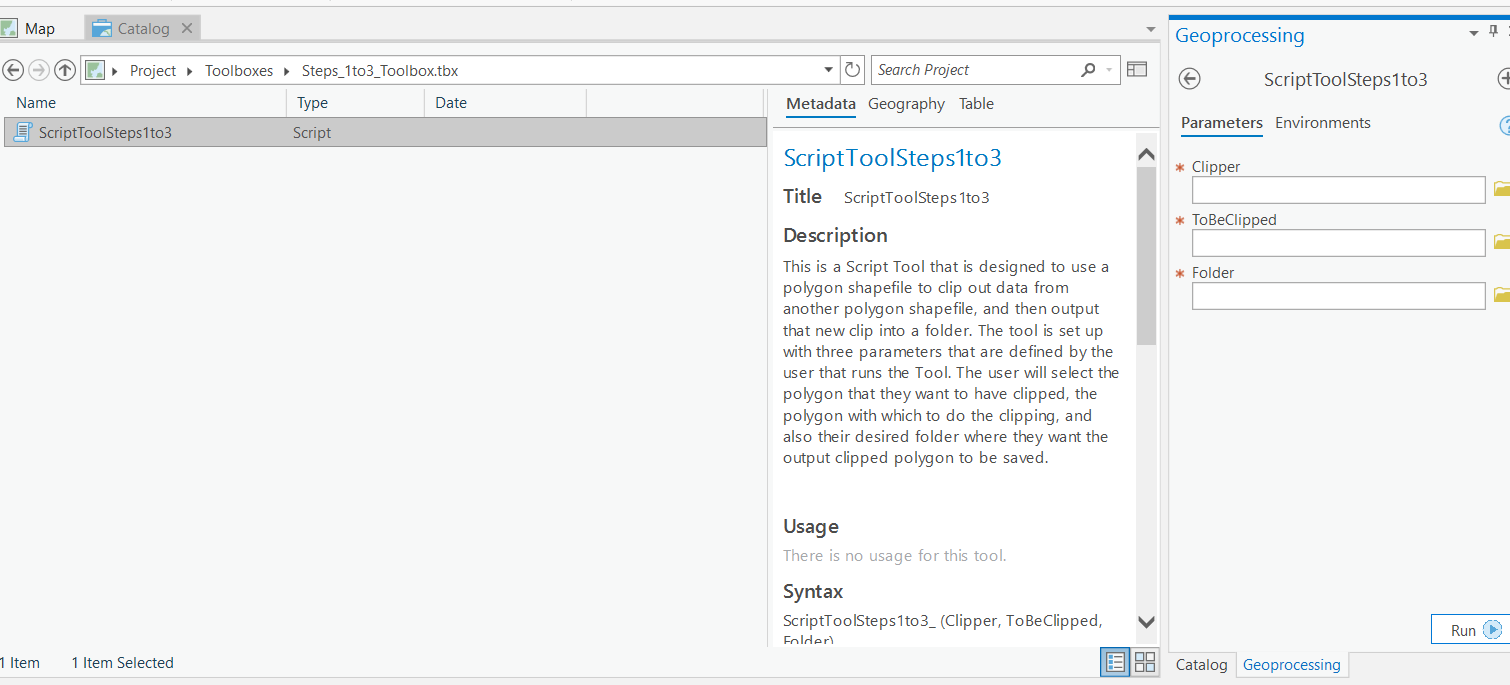
Below is a screenshot showing how I set up the Parameters within the Tool Properties of my Script Tool:



1. This step required updating the *worker* function code. Since we want the code to work with feature classes now, I updated the call to arcpy.MakeFeatureLayer\_management() to arcpy.MakeFeatureClass\_management(). To accommodate the new naming convention for the outFC variable, I updated that as well to include the clipper name provided by the user in their parameter to the output file path, and used the str() function to make sure it would be in a text format .

**Over and Above points:**

1. I put a copy of this summary on to GitHub. My username is “mpc23” and the repository I created is “GEOG489”.
2. I created a Script Tool for my Step 4 code as well.
3. I created in-tool documentation by writing a brief summary in the Metadata section of each of my two Script Tools, and also included Tags for each one.

****