Phillip Purvis

GEOG 489

Lesson 1 Writeup

I think this assignment was interesting getting back into using Arcpy and Python. To be honest I have not revisited Python since 485 and that was back in the Spring. One of the larger issues I had with coding this assignment was that I did a whole block of code at once instead of doing increments and it resulted in errors too complicated and time consuming to debug. I scrapped the code and worked on it incrementally like the whole concept of version control we went over in the lesson. The first step was to make the root folder path into a variable so the three example shapefiles would not be one ridiculously long line of code. Then I tested the input for the features to be clipped as a list but did not include multiple shapefiles just one. This led me to realizing for this instance that I had to use tobeclipped[0] so it could access the list. Then I made the code more complex and added a main for loop (above the scope of the id for loop) to iterate through still one item. I wanted to make sure that worked on its own before I added additional files. I changed the name tobeclipped to feature to not add confusion. Then I went to the worker function and made the output name change for the clipped features and then changed the temporary layer name. After I got it to work with an additional shapefile in the list I added more and then proceeded to clean up the code.

For the more advanced side such as adding it to the script tool, I had some issues with simple GetParameter. It produced an error regarding cannot pickle geoprocessing object, so I had to do the more complicated route. Using the split function and make the string into a list. After rectifying that it ran smoothly. To make the scripts more readable, I added metadata to both scripts with help prompts to the parameters. In addition, I added this write up and profiling results to the GitHub repository for analysis. I provided two screenshots of the profiling for Part 1. Please not the profile data is from PyCharm which was the main IDE I use for Python. It is interesting to note how the mp handler took 87% of the total time (15.9 seconds to run). With profiling 3 input feature classes, that resulted in 29.3 seconds. That surprisingly not triple the amount. Much of the time seemed dedicated to hidden processes running in the background.

Regarding writing to file geodatabases, I think it is difficult with the whole schema locking and no concurrent writing. A possible solution would be to have:

-the main tasks run in multiprocessing

-save each to an in-memory workspace

-have one task's job to save them to a geodatabase at a time from the in-memory workspace

This would not likely benefit from multiprocessing unless there was a way to have a worker actively one at a time saving what the other works produce as results. This would also be still bottlenecked by one worker doing this task repeatedly and the disk writing speeds.