NR 426 – Programming for GIS I
Lab 2 – Accessing Geoprocessing with arcpy

Warner College of Natural Resources | Colorado State University | Instructor: Elizabeth Tulanowski

Lab 1 learning objectives:

1. Understand the capabilities of arcpy
2. Implement Python code to run ArcGIS tools
3. Utilize non-tool arcpy methods to supplement data analysis

What to submit:

For each part, submit your code (using the template scripts provided) from the demos and self-directed activities.

Resources:

The Python Scripting for ArcGIS Book, Chapter 5 // Python for ArcGIS Pro Book, Chapter 2

ArcGIS arcpy help:

- https://pro.arcgis.com/en/pro-app/latest/arcpy/main/arcgis-pro-arcpy-reference.htm
- https://pro.arcgis.com/en/pro-app/latest/tool-reference/main/arcgis-pro-tool-reference.htm

Lab 2A: Accessing arcpy and running tools

Lab 2A In-class follow-along demos (done as a group):

1. Use the NR426 Lab 2A Template script to get started
    a. Located on the N:\ drive Lab 2 folder and the Lesson 2 Canvas module
2. We will walk through the template script writing code and explaining things as we go, to:
    a. Imports and variables
        i. Create variables for GDB and SHP paths, and feature services with a URL
    b. Finding syntax for a tool
    c. Running a few basic tools: Intersect, Select
        i. What SQL looks like in Python
    d. Running tools on raster data with the Spatial Analyst extension: Slope
3. Tricks for getting syntax and code samples
4. ArcGIS Pro Python window

Submit the above as part of your Lab 2A submission.

Lab 2A Self-directed activities (done on your own):

Continue using the Lab 2A template script from above to complete this section. Be sure to add your own additional comments, print statements, and empty lines for readability.

Imagine you are conducting a small study on the *bikeability* to parks in an area, working with layers for parks, natural areas, and bike routes.

You want to identify bike routes within 300 ft of parks or natural areas.

a. Use the Help and the code completion that pops up to guide your way.
b. Create any additional **variables** for the data you will use and the outputs to be created. Use these variables in the syntax for each tool you run.
    i. Look up and paste the syntax for any tools into a good spot in the script, to help you along.
    ii. Add print statements before each tool to inform the user what tool is about to run.

5. Write the code to perform the following tasks:
    a. Buffer the Parks layer by 300 feet.
    b. Buffer the Natural Areas layer by 300 feet.
    c. Union these two outputs together: Now you have a layer showing what's within 300 feet of all types of park areas
    d. Find bike infrastructure that is within your unioned buffer parks/natural areas layer.
        i. There are a few tools to do this: At this point, Intersect will be the easiest for you
6. **Report** the locations of each output dataset in a well formatted statement such as this. Use your variables, your arcpy.env.workspace, and maybe even os.path.join to print out the path. Don't hard-code (manually type) the full path into the print statement.

```
ParksBuffer can be found in <…path to your output…>
NaturalAreasBuffer can be found in <…path to your output…>
BikeNearPark can be found in <…path to your output…>
```

7. Practice with a raster tool: Unrelated to the bikeability analysis, run the Hillshade tool on the DEM
    a. Make sure the necessary imports are done and output is saved as we did in the demo (and on the Lesson 2 slide)
    b. Save the output to your arcpy.env.workspace

8. **Submit** the completed .py script file.

Lab 2B: Working with arcpy methods

Lab 2B in-class Follow-along demo (done as a group):

1. Use the Lab 2B template script provided on the N:\ drive and Canvas
    a. Set up imports, environments, variables
    b. Review finding help for tool syntax and running a basic tool: CLIP
        - Clip Trails to Natural Areas
    c. Working with other arcpy methods: Exists, Getting data properties with Describe, GetCount, overwriteOutput
    d. Modify the code to check that the data exists before running the Clip tool
    e. Add tool messaging for the user with GetMessages

- Submit the above as part of your Lab 2B submission.

Lab 2B Self-directed activities (done one your own):

Read up on the Describe and Spatial Reference documentation

2. Using your new skills, write the code to do the following tasks, as a Python script (use the Lab 2B template script).
Use clear commenting so your instructor can see what steps are being completed and grade it more easily!

a. Access another new layer in your script: the online feature class named Hydrology (lines), from the City of Fort Collins open data portal (https://data-fcgov.opendata.arcgis.com/).
    i. Create a variable for it like we did in the *Lab 2A Demo*
b. Run a tool to add the distance of the nearest bike route to each Park polygon attribute. (Which tool???? Ask the instructor for guidance or verification if you need it)
c. Intersect the Hydrology lines layer with the Trails layer
    i. This step will create a point layer of stream crossings along trails.
d. Add a well-formatted print statement telling the user:
    i. How many point features are in the output layer (Use GetCount like in the demo)
    ii. Try out the arcpy Describe method to return the geometry (shapetype)
e. Modify steps b and c above so the tools (both Near and Intersect) only run if certain conditions are met:
    i. The input dataset for each tool must exist
        1. Use an if statement
        2. Only let the script continue running if it exists (Hint: Look into sys.exit())
f. Modify the block of code running these two tools to add a try-except statement
    i. Make a print statement to return the arcpy messaging to the user (use GetMessages)

g. Modify your script as needed to make sure it has good comments, other useful print statements for the user, and good readability.

Optional challenges:

1. Modify the code to let the user pass in the path to the Lesson2LabData folder, and have the variables made in the code relative to that folder. (Hint: Use *input*)
2. Try running the code on a workspace or data you know won't work – then modify the code to make those errors more user-friendly.
3. Go into your ArcGIS Pro project: re-do some of these lines of code in the Python window to see how the code /setting up workspaces etc will differ, and how you can see result layers instantly in the map!
    - Make a screenshot of your code and submit to the Canvas item if you wish to share!

rev. 1/12/26