GEOG 432/832: Programming, Scripting, and Automation for GIS

Week 04.01: Even more geoprocessing in Python

Dr. Bitterman

Today's schedule

- Open discussion
- Discussion and exercises
- For next class

Open discussion

Functions

- A function is a piece of code that performs a specific task.
- Geogprocessing tools are functions, *but* not all functions are geoprocessing tools.
- What else can functions do?
    - list datasets
    - retrieve properties
    - check for the existence/presence
    - validate names

What else have we used?

note, not all ArcPy functions are available as standard tools in ArcGIS Pro

Calling a function

- a lot like calling a tool (b/c a tool is a function)
    - has parameters (which take arguments)
    - returns values (assuming it returns)

The syntax of a function in ArcPy is the same as for tools:

```
arcpy.<functionname>(<arguments>)
```

For example, this prints either True or False:

```
import arcpy
print(arcpy.Exists("C:/Data/streams.shp"))
```

what type of value do we think arcpy.Exists returns?

Many, many, many ArcPy functions

Some categories (not the full list - see text)

- ArcGIS Online/Portal
- Cursors
- Geodatabase admin
- Publishing
- Raster
- Spatial references and transformations

These categories won't be in the Python syntax

Miscellany

- ArcPy functions are divided into 2 categories:
    i. tool functions
    ii. nontool functions

Why do we care (a good question)?:

- Documentation is in different sections. Nontool functions are *only* documented under the Python tab of the ArcGIS Pro help pages
- Tools are licensed by:
    - license level (Basic, Standard, and Advanced)
    - extension (e.g., 3D Analyst, Network Analyst, Spatial Analyst)

(All arcpy nontool functions are available independent of the license level)

- Tools produce geoprocessing messages, nontool functions do not

Classes for tool parameters

So far, we have passed relatively simple parameters in our functions

For example:

```
inFc = "State_Park_Locations.shp"
clipFc = "lancaster_county.shp"
outputFc = "myFirstOutput.shp"

arcpy.Clip_analysis(inFc, clipFc, outputFc)
```

But parameters can be (and often are) more complex objects themselves

For example (any ideas...)???

Let's try this with the SpatialReference class

- Recall our discussion of object-oriented programming

What is a class?

Creating a new object from a class

Syntax:

```
arcpy.<classname>(parameters)
```

An actual example

```python
import arcpy
prjfile = "C:/Data/myprojection.prj"
spatialref = arcpy.SpatialReference(prjfile)
```

Let's break it down... what does the above code block do?

- What is the name of the class?
- What is the name of the object we created from the class?

What can you do with objects?

- Get work done!
- In our spatial reference example:
    - parameters of the CRS
    - domains
    - more

```python
import arcpy
prjfile = "C:/Data/streams.prj"
spatialref = arcpy.SpatialReference(prjfile)
print(spatialref.name)
```

what does the above code block do?

Many parameters are Strings

The previous code prnits the name of the spatial reference

`NAD_1983_StatePlane_Florida_East_FIPS_0901_Feet`

What is the CRS?

- Potential issues using a String representation of the CRS like this in your code?
- What does a .prj file look like? Open one in Notepad (or another text editor)

Alternative to working directly with Strings

- Much easier to refer to CRS by using the name of the coordinate system or referencing the .prj file that contains the string value
- Can do so using the SpatialReference class

For example, the synax to call Create Feature Class tool:

```
CreateFeatureclass(out_path, out_name, {geometry_type},
                   {template}, {has_m}, {has_z},
                   {spatial_reference}, {config_keyword},
                   {spatial_grid_1}, {spatial_grid_2},
                   {spatial_grid_3})
```

what are the parameters?

Substituting the spatial reference object for the string

```
import arcpy
out_path = "C:/Data"
out_name = "lines.shp"
prjfile = "C:/Data/streams.prj"
spatialref = arcpy.SpatialReference(prjfile)
arcpy.CreateFeatureclass_management(out_path, out_name, "POLYLINE",
                                    "", "", "", spatialref)
```

MUCH easier using the SpatialReference object than using actual string value contained in the .prj file

other stuff we're not going to cover in class

See your textbook for:

- tool messages
- dealing with licenses
- spatial references
- environments

Getting by with a little help (from our ESRI friends)

- Main support page for ArcGIS Pro is http://pro.arcgis.com/en/pro-app/help
- The Python tab brings up the ArcGIS Pro Python reference, which includes the official documentation of all the functionality of ArcPy.
- All ArcPy functions and classes are listed and described in detail, with sample code
- Separate sections on the various modules of ArcPy `(arcpy.da, arcpy.ia, arcpy.sa, and arcpy.mp)`

Today's in-class exercise

- All of you should completed Lab 1, part 2
- What did you do?

  i. Interpolate a precipitation surface from your points
  ii. Reclassify the interpolated surface into an ordinal classification of precipitation "zones" that delineate relatively dry, medium, and wet regions
  iii. Create vector polygons from the zones
  iv. Clip the zone polygons to the boundary of Nebraska

Today's task: replicate the workflow in a Python script

- Useful resources???
- How to structure your analysis???

For next class

- Readings
  - Wilson et al. 2020. Come with **at least one** (more is ok/great) discusison questions. We are going to discuss the paper as in a seminar
- Practice!
- Lab 01 is due on 2/19