An Introduction to Geoprocessing with ArcPy

August 24, 2016

1 An Informal Introduction to Geoprocessing

Most likely, a majority of the GIS work you do in your life (and certainly in this class) will not be cartographic, that is, creating map products. Instead, it will be work processing geographic data, working with *spatial data* to create *spatial information*. But this is essentially jargon on the level of "synergy" talks that I assume must go on corporate offices around the world. What's important is that you will be spending a lot of time in the magical (and often frustrating world of "geoprocessing".

1.1 But What Does Geoprocessing Mean for Me?

A good working definition for geoprocessing in the context of ArcGis is anything you do with one of the "tools". "Hey!", you may be thinking to yourself, "I've already geoprocessed all sorts of junk! I've hillshaded and projected and...", you're right that's geoprocessing, but don't close this document yet! As you do more and more GIS, you'll begin to do geoprocessing that's more complex, involving more files, steps and options. While you can (usually) get by searching for a tool in ArcMap, running it on one of your files, searching for the next tool, locating the output of the first tool, and running the second tool on that output, etc., there is a much more elegant and powerful way and elegant way to go about it.

2 Scripting

2.1 Making Lists

Imagine you had an unprojected elevation raster for Oberlin with the NAD83 datum that you wanted to project to UTM with the NAD83 datum, and then make into a hillshade. If you weren't currently in the GIS lab, you could write out a list of instructions to follow later:

1. Project raster
2. Make hillshade

Unfortunately, you are delayed 6 months and forget which raster you wanted to work with. This could've been solved with a more detailed list of instructions:

1. Project Oberlin elevation raster
2. Make hillshade

Problem solved? Alas, this time you were delayed a year and have forgotten where you stored all of your many elevation rasters on your hard drive. Luckily, this can be solved with an even more detailed list:

1. project `F:/gis/elevation_rasters/oberlin_dem.tif`
2. Make hillshade

However, this time you are delayed 25 years and have forgotten how all these tools work! So a more detailed list is in order:

1. Use the Project Raster tool with the following options:

   Input Raster: `F:/gis/elevation_rasters/oberlin_dem.tif`

   Output Raster: `F:/gis/elevation_rasters/oberlin_dem_utm.tif`

   Output Coordinate System: `NAD_1983_UTM_Zone_17N`

2. Use the Hillshade tool with the following options:

   Input Raster: `F:/gis/elevation_rasters/oberlin_dem_utm.tif`

   Output Raster: `F:/gis/hillshades/oberlin.tif`

What a list! Sure, it's detailed, but now you could give this list and your hard drive to anyone with a little ArcGIS experience and they could do it for you. However, I must admit it seems unlikely that you will be delayed for 25 years on your way to the GIS lab (and if you are, you have bigger problems than making a hillshade!), and making a list with this level of detail to give to someone else might not save a whole lot of your time.

2.2 Lists for Computers

Scripting is essentially writing lists of instructions, like the ones above, that a computer will follow, preforming each instruction one by one. This involves learning how to write lists that computers can understand, but becomes a real time saver and a powerful tool. With scripts (list of instructions written for computers) multi-step processes can be automated and reused. Scripts can be incredibly simple and incredible complex, but even the simplest ones can be massive time savers.

3 Scripting for ArcGIS

When writing scripts (instruction lists) for ArcGIS you will write in the style used by the Python programming language. This doesn't mean you will need to learn how to *program* with Python, simply how to write instruction lists in a way that Python understands. There are three main things you need to know about writing lists that Python can understand, first: that Python executes each step in order; second, how to tell Python to run tools, and third, how to name pieces of information so that you can refer to them later. We will illustrate these three things by looking at a pre-written script.

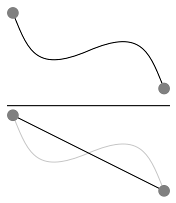
3.1 Calculating Sinuosity With a Script

3.1.1 Sinuosity

You will be using a script to calculate the sinuosity of the river features you digitized earlier in the lab. Sinuosity is defined as follows:

$$\frac{\text{the length of the path of the river}}{\text{the distance between each end of the river}}$$

or graphically:



Since the length of river's path can never be shorter than the distance between the ends, the minimum sinuosity value will be 1.

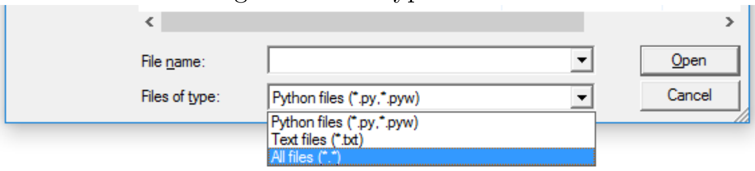
3.1.2 The sinuosity script

1. First, you need to get the sinuosity script.
   (a) Use the internet to search for a script to calculate sinuosity in ArcGIS. You should find a result on ArcGIS.com which has something for you to download.
   (b) Download and extract the zip file.
   (c) Find the file called `Sinousity.pyt`. A `.pyt` file is a special type of Python file that ArcGIS can use like a toolbox.
2. Now, run the script
   (a) Go into the ArcCatalog window and navigate to the `pyt` file. Expand the toolbox and open the Calculate Sinuosity tool.
   (b) Run the tool on your digitized river path shapefile.

3. It should have added a new field in the attribute table with the sinuosity of the river. However, the value is less than one! It turns out, the sinuosity tool is calculating inverse sinuosity. While it would be easy to simply fix the result, lets use this as learning opportunity and fix the script instead.

   (a) Open the program IDLE (Python GUI).

   (b) In IDLE, open up the Sinuosity toolbox.
   You will have to change "Files of Type" to "All files".

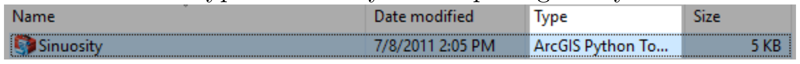


File name:

Files of type: Python files (*.py,*.pyw)

Python files (*.py,*.pyw)
Text files (*.txt)
All files (*.*)

Be sure that the type of the file you are opening is "Python Toolbox"



Name	Date modified	Type	Size
Sinuosity	7/8/2011 2:05 PM	ArcGIS Python To...	5 KB

Wow. There is a lot going on here. However, only a few lines are the script itself, the rest is code dedicated into making it a tool that you can run graphically in ArcMap. The important part is the four lines below `def getSinuosity(shape)`:

```
sinuosity_expression = '''
import math
def getSinuosity(shape):
    length = shape.length
    d = math.sqrt((shape.firstPoint.X - shape.lastPoint.X) ** 2 +
                  (shape.firstPoint.Y - shape.lastPoint.Y) ** 2)
    return d / length
'''
```

Take special note of the second and third lines:

```
sinuosity_expression = '''
import math
def getSinuosity(shape):
    length = shape.length
    d = math.sqrt((shape.firstPoint.X - shape.lastPoint.X) ** 2 +
                  (shape.firstPoint.Y - shape.lastPoint.Y) ** 2)
    return d / length
'''
```

Notice the indentation. This is really one line split into two lines simply to increase readability. So what we have is a list with three steps, the one starting with "`length`", the one starting with "`d`", and the one starting with "`return`". Lets take them one at a time.

1. `length = shape.length` This line tells Python to label a piece of information. Python uses the `=` character to label things. This line says "use '`length`' to refer to '`shape.length`', a label defined elsewhere that refers to length of the shape the tool is running on.

2. `d = math.sqrt(...)` This line is somewhat more complex. From `d = ...` we can see that there is an instruction to label something with “`d`”, however *what* is being labeled seems rather obtuse. For now lets focus on the sets of parentheses.

   ```
   d = math.sqrt( (shape.firstPoint.X - shape.lastPoint.X) ** 2 +
                  (shape.firstPoint.Y - shape.lastPoint.Y) ** 2)
   ```

   There are three sets of parentheses, the cyan parentheses, which encompasses both the yellow and orange parentheses. The text in between the cyan parentheses is a math expression, with labels like `shape.firstPoint.X` filling in for numbers. The yellow and orange parentheses are there to ensure correct order of operations in the math expression. So far, we can understand this instruction as “use ‘`d`’ to refer to ‘`math.sqrt([math expression])`’ ”

   But just what is up with `math.sqrt(...)` anyway? `math.sqrt` is a tool, like hillshade or project raster. The parentheses specify what should be used as input to the tool. In this case, the tool calculates the square root, and its input is math expression. So this instruction can be read as “run the square root tool on the given math expression and give the result the label ‘`d`’.”

3. `return d / length` This instruction has two components, `return` and `d / length`. `d / length` is a simple math expression, divide the number that the label `d` represents by the number that the label `length` represents. You don’t need to worry about the `return` component, it simply means take the result of the division operation and make it available for use elsewhere in the script.

Now that you’ve looked at each instruction, you should be able to figure out what to change to make the script calculate sinuosity as we’ve defined it. Make the change, save the file, and re-navigate to and run the tool. Congratulations! You’ve just worked with your first ArcPy script, and are well on your way to making your own.

`Now go back to the Lab3b for directions on what to turn in for the lab.`