Lab 3 Instructions - Manipulating (and Accessing) Spatial Data
Due by midnight the day of your lab, Week 5

This lab continues with an exploration of the data tables that underlay the vector data structures used to represent geospatial information. Using a cursor, the locations of the points and vertices that make up vector points, lines, and polygons is explored.

Instructions

Copy and paste the “Lab3” folder from the class userspace to your userspace on the R: drive. Run through Exercise 9 (The PDF is located in the “Lab3” folder - it has been edited to match the ‘virtual lab’ data locations better).

**NOTE - paths to data have only been partially corrected in the PDFs! you should use something like the following syntax to correct this in your scripts…

```
workspace = "R:/Geog491_591_6/Student_Data/yourusername/Lab3/Ex09"
arcpy.env.workspace = workspace
```

NOTE - **comments are a key part of your grade. Make sure your name is in the comments at the head of the script and be sure your code is fully commented. Here is a sample header comment:

```
# Name: PassengerCalm.py
# Purpose: Mimics a scene from the movie Airplane in which a panicked jet
# passenger is 'calmed down' by a series of violent slapstick interactions.
# Allows one round of user input
# Author:      Nicholas
# Created: 04/11/2014
```

Here is a link to suggested best practices for commenting in Python:

http://legacy.python.org/dev/peps/pep-0008/#comments

Using IDLE in the virtual lab

*Note - when using the virtual 'High Performance Lab' it is a bit tricky to get a version of the IDLE program started that likes to work with ArcPro. Steps to open the correct version of IDLE are:

1. Steer to the *.py file you want to open using the file browser in Windows (if you want to make a new *.py file, steer to an old one first to open IDLE, then create a new script once the correct IDLE is open).
2. Right-click on the *.py file and choose to "Edit with IDLE (ArcGIS Pro)"

3. The version of IDLE running Python 3.6.9 should start up with the script open to edit (it will have a '(3.6.9)' at the end of the file name at the top).

Exercise 09 - Instructions

1) Save the short python scripts you create in Exercise 9 in the "Lab3" folder in your userspace. ***Add comments to the finished scripts "Multipart_vertices.py" and "Create_polyline.py" to show you understand what they do! Your grading will primarily be on the comments!***

2) Do the following challenge exercises, and comment the scripts. Here are some hints:

- ***Challenge 1:*** Write a script that determines the perimeter (in meters) and area (in square meters) of each of the individual islands of the Hawaii.shp feature class. Recall that this is a multipart feature class.
    - The trick here is dealing with the multipart feature. One way to get around this is to turn it into a single part feature (one part for each island) using the "Multipart to Singlepart" function in arcpy. Then proceed like the example in the beginning of the exercise that reads road lengths.

- ***Challenge 2:*** Write a script that creates an envelope polygon feature class for the Hawaii.shp feature class. There is actually a tool that accomplishes this called Minimum Bounding Geometry. You can look at the tool to get some ideas, but your script needs to work directly with the geometry properties
    - This one is easier when working with a multipart feature with only 1 part (as compared to a single part feature with many parts) - the 'extent' of this feature is a class in ArcPy which can be queried for the envelope, and extent is also part of the Arcpy Describe object. See as an example: Extent—ArcGIS Pro | Documentation

Turn in:

1. Two scripts from the exercise 8 ***with comments*** - 'multipart_vertices.py', 'create_polyline.py'
2. The scripts for the 2 challenge exercises with comments.
3. Written answers to the following questions...

   Questions:

   1. What does an 'array' do within the context of geometry objects?

   2. When processing through the vertices of a part of a 'multipart' data layer (such as the *`Hawaii.shp`* file), how many nested 'for' loops are necessary to use to access the information about the actual x, y locations of the vertices? What does each one of the 'for' loops process through?

   3. In the 'multipart.py' script, why is it necessary to use the following statement?

```
if row[1].isMultipart:
```

Turn in your work on Canvas in the assignments section.

- the 2 commented python scripts in the exercise.
- 2 commented scripts to answer the challenge exercises.
- a document with answers to the lab questions.

20 points (2 Lab scripts with comments - 4 points; Challenge Exercises 1 - 5 points; Challenge 2 - 5 points; Lab Questions - 6 points).