GEOG503A Lab 7

Use PyCharm or Jupyter Notebook to perform the following tasks:

Tasks:

Input data folder: \lab7

Results folder: \lab7\Results\

1. Write a script that creates a new polygon feature class containing a single (square) polygon with the following coordinates: (0, 0), (0, 1,000), (1,000, 0), and (1,000, 1,000). (5 pt.)
2. 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. (5 pt.)
3. 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. (10 pt.)
4. Write a script that creates a Polygon shapefile based on the coordinates from the *coordinates.txt*, which contains coordinate (X, Y) pairs for 11 features. You should use relevant Geometry classes to perform this task. Do NOT use InsertCursor function. Use the spatial reference of *Hawaii.shp* for the resulting Polygon shapefile. (20 pt.)

What to submit:

1. The Python scripts (or a single Jupyter Notebook) for the tasks above. Submit the zip file that only contains the scripts without the data.

File name convention for assignment submissions:
lastname_firstname_lab7.zip

Exercise 8
Working with geometries

Work with geometry objects

Working with full geometry objects can be costly in terms of time. Geometry tokens provide shortcuts to specific geometry properties of individual features in a feature class. In the next example, you will see how to work with geometry tokens in scripting.

1 **Start PythonWin. Create a new Python script and save as** geometry.py **to your C:\EsriPress\Python\Data\Exercise08\Results folder.**

2 **Enter the following code:**

```
import arcpy
from arcpy import env
env.workspace = "C:/EsriPress/Python/Data/Exercise08"
fc = "rivers.shp"
cursor = arcpy.da.SearchCursor(fc, ["SHAPE@LENGTH"])
length = 0
for row in cursor:
    length = length + row[0]
print length
```

3 **Save and run the script.** This prints the total length of all the river segments to the Interactive Window. The spatial reference object can be used to determine the units.

4 **Replace the** `print length` **statement with the following lines of code:**

```
units = arcpy.Describe(fc).spatialReference.linearUnitName
print str(length) + " " + units
```

5 **Save and run the script.**

This prints the total length of all the features, followed by the units, to the Interactive Window, as follows:

```
256937.409437 Meter
```

In the `print` statement, the length is converted to a string because if it were a number followed by a plus sign (+), it would suggest a calculation.

Read geometries

In addition to the geometry properties of a feature, a feature's individual vertices can also be accessed. You'll do that next.

1 In PythonWin, create a new Python script and save as points.py **to the Results folder for exercise 8.**

2 Enter the following code:

```
import arcpy
from arcpy import env
env.workspace = "C:/EsriPress/Python/Data/Exercise08"
fc = "dams.shp"
cursor = arcpy.da.SearchCursor(fc, ["SHAPE@XY"])
for row in cursor:
    x, y = row[0]
    print("{0}, {1}".format(x, y))
```

3 Save and run the script.

This prints a pair of x,y coordinates for each point, as follows:

```
852911.336075 2220578.93538
792026.89767 2310863.76822
784830.427658 2315171.53882
741687.458878 2321601.76549
702480.327773 2340545.89511
623387.057118 2361903.92116
...
```

Working with other geometry types, such as polylines and polygons, requires some extra code because an array of point objects is returned for each feature. As a result, an extra iteration is required to interact with the array first before you can get to the points that make up the array. Next, you will work with a polyline feature class.

4 In PythonWin, create a new Python script and save as vertices.py **to the Results folder for exercise 8.**

5 Enter the following code:

```
import arcpy
from arcpy import env
env.workspace = "C:/EsriPress/Python/Data/Exercise08"
fc = "rivers.shp"
cursor = arcpy.da.SearchCursor(fc, (["OID@", "SHAPE@"]))
for row in cursor:
    print("Feature {0}: ".format(row[0]))
    for point in row[1].getPart(0):
        print("{0}, {1}".format(point.X, point.Y))
```

6 Save and run the script.

This prints a pair of x,y coordinates for each vertex in each feature, as follows:

```
Feature 0:
745054.499005 2324903.18355
745122.149446 2324835.9415
...
Feature 1:
747821.018772 2317111.87693
746909.057058 2317254.89224
...
Feature 2:
753597.862134 2315541.91647
753757.716791 2319122.62937
...
Feature 3:
751089.22614 2314030.50506
749458.383228 2315289.96843
...
```

This script works for polyline and polygon feature classes but does not work for multipart features, which you will work with next.

Work with multipart features

For multipart features, cursors return an array containing multiple arrays of point objects. Working with multipart features therefore requires an extra iteration over the parts of each feature.

The `isMultipart` property of the geometry object can be used to determine whether a particular feature is multipart.

1 In PythonWin, create a new Python script and save as multipart.py **to the Results folder for exercise 8.**

2 Enter the following code:

```
import arcpy
from arcpy import env
env.workspace = "C:/EsriPress/Python/Data/Exercise08"
fc = "dams.shp"
cursor = arcpy.da.SearchCursor(fc, ["OID@", "SHAPE@"])
for row in cursor:
    if row[1].isMultipart:
        print("Feature {0} is multipart.".format(row[0]))
    else:
        print("Feature {0} is single part.".format(row[0]))
```

3 Save and run the script.

This prints whether each feature in the features class is multipart or single part, as follows:

```
Feature 0 is single part.
Feature 1 is single part.
Feature 2 is single part.
Feature 3 is single part.
Feature 4 is single part.
...
```

The results are single part for each feature, because each feature consists of a single point—that is, a dam.

4 Modify the script as follows:

```
fc = "Hawaii.shp"
```

5 Save and run the script.

This confirms that the feature class representing Hawaii is a multipart polygon, as follows:

```
Feature 0 is multipart.
```

The `partCount` property can be used to determine the number of parts, which you'll do next.

6 Modify the first `print` statement following the `if` statement as follows:

```
    print("Feature {0} is multipart and has {1} parts.".format(row[0], ➔
  ➔ str(row[1].partCount)))
```

7 Save and run the script.

This prints the number of parts for every multipart feature, as follows:

```
Feature 0 is multipart and has 11 parts.
```

Because the geometry object for multipart features consists of an array containing multiple arrays of point objects, it is necessary to iterate over all parts of a feature to obtain its geometry. You need a loop to iterate over the arrays in the array, and then you need a loop to iterate over each point in the point array.

8 Modify the script as follows:

```
import arcpy
from arcpy import env
env.workspace = "C:/EsriPress/Python/Data/Exercise08"
fc = "Hawaii.shp"
cursor = arcpy.da.SearchCursor(fc, ["OID@", "SHAPE@"])
for row in cursor:
    print("Feature {0}: ".format(row[0]))
    partnum = 0
    for part in row[1]:
        print("Part {0}: ".format(partnum))
        for point in part:
            print("{0}, {1}".format(point.X, point.Y))
        partnum += 1
```

9 Save and run the script.

This prints the part number of each feature, followed by pairs of x,y coordinates of the vertices. In the case of the feature class Hawaii.shp, there is one feature with 11 parts, as follows:

```
Feature 0:
Part 0:
829161.23775 2245088.48637
829562.014007 2244825.55006
...
Part 1:
757434.846245 2276341.38313
757032.135863 2276024.28579
...
Part 2:
710001.620203 2315999.27091
712130.145201 2315404.37626
...
```

Write geometries

Insert and update cursors can be used to create new features and update existing features, respectively. Point objects can be created to set the geometry of these new or updated features.

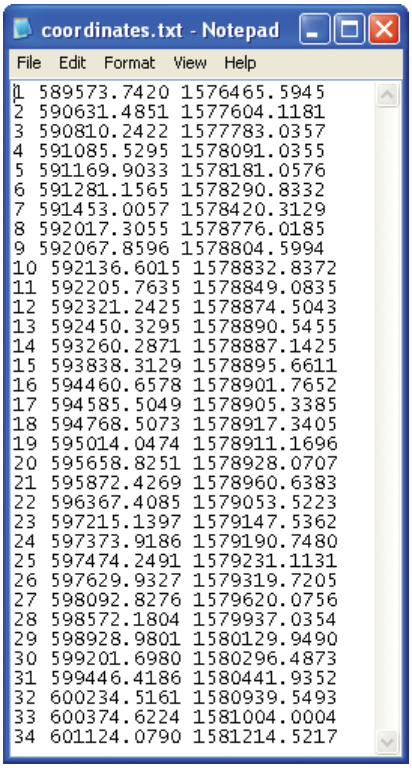
In the next example, you will use an insert cursor to create a new polyline feature from a list of coordinates.

1 Start Notepad. Typically, you can get there by clicking the Start button and then, on the Start menu, click All Programs > Accessories > Notepad.

2 On the Notepad menu bar, click File > Open and browse to the Exercise08 folder. Select the coordinates.txt file and click Open. ➔

The file consists of 34 pairs of x,y coordinates, with each pair preceded by an ID number and separated by a space. First, you will create a new empty polyline feature class, and then use this list of coordinates to create a new polyline.

3 Close Notepad.



4 **In PythonWin, create a new Python script and save as** create.py **to the Results folder for exercise 8.**

5 **Enter the following code.**

```
import arcpy
import fileinput
import string
import os
from arcpy import env
env.workspace = "C:/EsriPress/Python/Data/Exercise08"
env.overwriteOutput = True
outpath = "C:/EsriPress/Python/Data/Exercise08"
newfc = "Results/newpolyline.shp"
arcpy.CreateFeatureclass_management(outpath, newfc, "Polyline")
```

6 **Save and run the script.** This creates a new empty polyline feature class.

7 **Start ArcMap. Open the Catalog window. Navigate to the Results folder for exercise 8 and confirm that the file** newpolyline.shp **has been created.**

8 **Close ArcMap. There is no need to save your map document.**

Next, you will read the text file.

9 **In the create.py script, add the following line of code:**

```
infile = "C:/EsriPress/Python/Data/Exercise08/coordinates.txt"
```

Note: *The full path is needed here because the workspace applies to only ArcPy functions and classes.*

The coordinates in this text file will be used to create the vertices for a polyline. This requires the use of an insert cursor to create new rows and an array object to contain the point objects.

10 **Add the following lines of code:**

```
cursor = arcpy.da.InsertCursor(newfc, ["SHAPE@"])
array = arcpy.Array()
```

Next, the script needs to read through the text file, parse the text into separate strings, and iterate over the text file to create a point object for every line in the text file.

11 Add the following lines of code:

```
for line in fileinput.input(infile):
    ID, X, Y = string.split(line," ")
    array.add(arcpy.Point(X, Y))
cursor.insertRow([arcpy.Polyline(array)])
fileinput.close()
del cursor
```

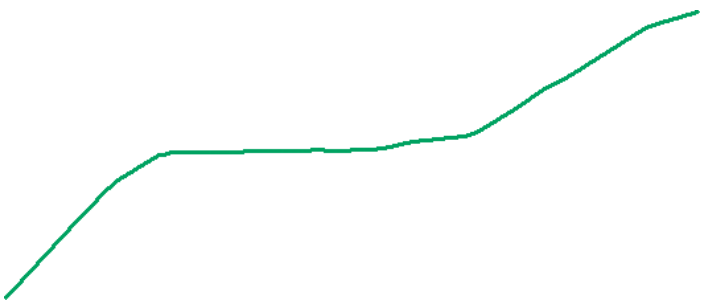
The completed script looks like the example in the figure.

newpoly.py

```
import arcpy
import fileinput
import string
import os
from arcpy import env
env.workspace = "C:/EsriPress/Python/Data/Exercise08"
env.overwriteOutput = True
outpath = "C:/EsriPress/Python/Data/Exercise08"
newfc = "Results/newpolyline.shp"
arcpy.CreateFeatureclass_management(outpath, newfc, "Polyline")
infile = "C:/EsriPress/Python/Data/Exercise08/coordinates.txt"
cursor = arcpy.da.InsertCursor(newfc, ["SHAPE@"])
array = arcpy.Array()
for line in fileinput.input(infile):
    ID, X, Y = string.split(line," ")
    array.add(arcpy.Point(X, Y))
cursor.insertRow([arcpy.Polyline(array)])
fileinput.close()
del cursor
```

12 Save and run the script.

13 Start ArcMap. Open the Catalog window and browse to the Results folder for exercise 8. Drag newpolyline.shp into the map document.



When adding the shapefile to the map document, you may get an error related to the spatial reference because this has not been set. This could be added to the code using the Create Feature Classes tool or as a separate line of code—for example, using the Define Projection tool.

Challenge exercises

Challenge 1

Write a script that creates a new polygon feature class containing a single (square) polygon with the following coordinates: (0, 0), (0, 1,000), (1,000, 0), and (1,000, 1,000).

Challenge 2

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.

Challenge 3

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.