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

Unit 07.02: Geometries

Dr. Bitterman

Today's schedule

- Open discussion
- Slides, discussion and exercises
- For next class

Open discussion

How's lab 3 going?

Geometries

It's useful to remember our spatial topologies

- Points: 0-dimensional
- Lines: 1-dimensional, made of two points and an arc
- Polygons: 2-dimensional, made up of lines

The structure of geometries

- So far, we've looked at feature classes mostly with respect to their attributes
- But there's a geometry attribute as well... accessed with the `SHAPE@` syntax

Let's try - open ArcGIS Pro and the week 7 data

```
workspace_string = "C:\\Users\\pjbitterman\\Dropbox\\GEOG432\\week07\\week07data"
arcpy.env.workspace = workspace_string
streams = "Streams_303_d_.shp"

with arcpy.da.SearchCursor(streams, "SHAPE@") as cursor:
    for row in cursor:
        print(row[0])
```

What happened?

`SHAPE@` gives you access to many geometry-related properties

Let's try:

```
streams = "Streams_303_d_.shp"

with arcpy.da.SearchCursor(streams, "SHAPE@Length") as cursor:
    for row in cursor:
        print(row[0])
```

See for more properties: https://pro.arcgis.com/en/pro-app/latest/arcpy/get-started/reading-geometries.htm

And we can do more

```
streams = "Streams_303_d_.shp"

length_total = 0
with arcpy.da.SearchCursor(streams, "SHAPE@Length") as cursor:

    for row in cursor:
        length_total += row[0]
    print(length_total)
```

Straightforward?

Paired programming exercise

Creating geometry

Let's create some geometry

General syntax

```
arcpy.Geometry(geometry, inputs, {spatial_reference}, {has_z}, {has_m})
```

Specifying a point is not the same as creating a Geometry object

```
point = arcpy.Point(100, 500)
arcpy.PointGeometry(point, 26914)
```

`26914` is the "factory code" for the CRS. Can use other representations too

Let's make a line

Break it down

```
point_a = arcpy.Point(0,0)
point_b = arcpy.Point(100, 500)
listOfPoints = arcpy.Array([point_a, point_b])
myline = arcpy.Polyline(listOfPoints, 26914)

print(myline.length)
myline
```

What happened?

and we can do simple checks:

What does this code do?

```
point_1a = arcpy.Point(0,0)
point_1b = arcpy.Point(100, 500)
listOfPoints_1 = arcpy.Array([point_1a, point_1b])
myline_1 = arcpy.Polyline(listOfPoints_1, 26914)

point_2a = arcpy.Point(100,0)
point_2b = arcpy.Point(0, 500)
listOfPoints_2 = arcpy.Array([point_2a, point_2b])
myline_2 = arcpy.Polyline(listOfPoints_2, 26914)

print(myline_1.crosses(myline_2))
#myline_1
#myline_2
```

and some automation:

```
mylistofxy = [[0, 1], [4, 6], [2, 2.33], [1, 1]]

listOfPoints = arcpy.Array()

for (x, y) in mylistofxy:
    mypoint = arcpy.Point(x,y)
    listOfPoints.append(mypoint)

myline = arcpy.Polyline(listOfPoints, 26914)
print(myline.length)

myline
```

something (just) a bit different

Let's make a multipoint geometry

```
mylistofxy = [[0, 1], [4, 6], [2, 2.33], [1, 1]]

listOfPoints = arcpy.Array()

for (x, y) in mylistofxy:
    mypoint = arcpy.Point(x,y)
    listOfPoints.append(mypoint)

arcpy.Multipoint(listOfPoints)
```

What happened?

you can do interesting work with a Geometry object:

How is this different?

```
mylistofxy = [[0, 1], [4, 6], [2, 2.33], [1, 1]]

listOfPoints = arcpy.Array()

for (x, y) in mylistofxy:
    mypoint = arcpy.Point(x,y)
    listOfPoints.append(mypoint)

myMp = arcpy.Multipoint(listOfPoints)
myMp.convexHull()
```

Your result?

Back to the task(s) from last class:

For all state parks within 2 miles of a municipal boundry, figure out the size (in acres) of the parks and what county they are in. Print the output like:

```
Arbor Lodge SHP is 56.788745 acres and in Otoe County
```

Optional bonus points (only try once you've completed above):

- (easier) Format the park size such that it only prints 2 decimal places
- (harder) Include what muncipality they are nearest to
- (even harder) Print whether the state parks are within 10 miles of a 303d stream

For next class

- Read Chapter 10 for next week