

## Lesson 2: Fundamentals of Python

Download the Lesson2.zip file from the course webpage. Unzip the folder to begin.

### CLASS ACTIVITY: Learning Python Syntax

- 1) We'll begin by entering Python code in the Interactive Window.
- 2) The >>> is the PythonWin prompt. Do not type >>>. After typing, press Enter and see what is returned.
- 3) This is what we did in class (text after the prompt is what we typed, text without a prompt is what was returned).

```
>>> # Working with variables
>>> boy = "Jim"
>>> print boy
Jim
>>> a = 2 + 4
>>> print a
6
>>> boyList = ["Jim", "John", "Andy", "Frank"]
>>> print boyList[0]
Jim
>>> print boyList[2]
Andy
>>> print boyList[-1]
Frank
>>> shpName = "Streams.shp"
>>> print shpName[:-4]
Streams
>>> # Working with built-in functions
>>> print round(9.2333)
9.0
>>> # Importing a new module
>>> import math
>>> # Find out what functions come with the math module
>>> dir(math)
['__doc__', '__name__', 'acos', 'asin', 'atan', 'atan2', 'ceil',
 'cos', 'cosh', 'degrees', 'e', 'exp', 'fabs', 'floor',
 'fmod', 'frexp', 'hypot', 'ldexp', 'log', 'log10', 'modf',
 'pi', 'pow', 'radians', 'sin', 'sinh', 'sqrt', 'tan',
 'tanh']
```

```

>>> # Look at the documentation for the log function in the math
      module
>>> print math.log.__doc__
log(x[, base]) -> the logarithm of x to the given base.
If the base not specified, returns the natural logarithm (base e)
of x.
>>> # Import the string module
>>> import string
>>> string.capitalize("HELLO")
'Hello'
>>> # Example of Concatenating strings
>>> a = "streams"
>>> p = ".shp"
>>> print a + p
streams.shp
>>> filename = string.capitalize(a[0]) + a[1:] + p
>>> print filename
Streams.shp

```

Do the following on your own:

- 4) Open a new Script Window. Click on File > New, then choose Python Script, then click OK.
- 5) Now copy all the text from the Interactive Window and paste it in the Script Window. Highlight all the text from the Interactive Window and right-click and choose Copy. Put your cursor in the top of the Script Window, then right-click and choose Paste.
- 6) Click the Edit menu and choose Replace. For Find What put >>> and leave Replace with: blank. Then click the Replace All button. This removes all >>> prompts.
- 7) The output from what you pasted is flagged with a – to the left of it. Delete each of these by backspacing. Then take out the blank character between each remaining line by selecting all, then right-click>source code>Dedent region. All the code should now be flush with the left margin of the Script Window.
- 8) Go to the Interactive Window, right-click and choose Select All. Then press your delete key. The Interactive Window should be blank.
- 9) Save the script by first making the Script Window active by clicking on it (anywhere). Then go the File menu, and choose Save As.... Save the file as lesson2a.py in the Lesson2 folder.
- 10) Now, run the lesson2a.py file by clicking on the Run button, or choosing Run from the File menu.
- 11) We're read for more, but first let's comment out all the code we've got in the script already. Highlight all the text in the Script Window, right-click>Source Code>Comment out region.

- 12) Now, type the following code:

```
# working with decision making
x = 25
if x < 25:
    print "the number is less than 25"
elif x > 25:
    print "the number is greater than 25"
else:
    print "the number is 25"
```

- 13) Press the Run button. In Interactive Window you should see the number is 25.

- 14) Things to remember: if, elif and else must be lowercase, colons must be at the end of each condition, and Python interprets the indentation as the end of the condition, thus executing the line of code.

- 15) Highlight the decision making code and comment it out as before, and save the file.

- 16) Now, type the following code:

```
# working with loops
# while loop example
z = 5
while z < 30:
    print z
    z = z + 5
```

- 17) Press Enter twice to end the construct. Press the Run button (note, when you press the Run button, PythonWin automatically saves the file before running it). You should see the following values in the Interactive Window: 5 10 15 20 25

- 18) Highlight the while loop example code, and comment it out as before, and save the script file.

- 19) Now, type the following code:

```
# Counted loop example
grdList = ["elev30m", "elev30m10", "elev30m100"]
for eachGrd in grdList:
    print eachGrd
```

- 20) Press Enter twice to end the construct. Press the Run button. You should see the following in the Interactive Window:

```
elev30m
elev30m10
elev30m100
```

- 21) Highlight the count loop example code, and comment it out as before, and save the script file.

- 22) Now, type the following code:

```
# for loop example
for num in range (2,6):
    print num
```

- 23) Press Enter twice to end the construct. Press the Run button. You should see the following values in the Interactive Window: 2 3 4 5
- 24) Things to remember: while, for, in, and range have to be lowercase. Colons are required at the end of each while and for statement. Python recognizes the construct as a loop only when it ends with an indentation—colons and indentations are VERY important in Python! Also, Python is friendly to “white space” that is, whether you have one space or multiple spaces, it is all the same (e.g. for num in range (2,6) is the same as for num in range (2,6)).
- 25) Close the python script lesson2a. File > Close.

## Assignment 2b: Writing a Simple Python Script

- 1) Open the Lesson2b.mxd and look at the flowline layer. Minimize ArcMap or move it to the side.
- 2) In PythonWin open a new script window and save it as lesson2b.py in the Lesson2 folder.
- 3) Type the following comments to start the script.

```
# Author: John Lowry
# Date: August 17, 2007
# Purpose: Lesson 2b example of a simple python script
```

- 4) Import the arcgisscripting module and create the gp object.

```
# Import the arcgisscripting module
import arcgisscripting
# Create the Geoprocessor object
gp = arcgisscripting.create()
```

- 5) Set the current workspace as the Lesson2 folder (note your path to the Lesson2 folder may be different than what is typed below). [Note you can also think of this as “assigning the workspace *property* to the gp object.”]

```
# Set the current workspace
gp.workspace = "C:/<your path here>/Lesson2"
```

- 6) In ArcToolbox, find the Select tool. If you don't know where it is, use the Search Tab. Choose the Select tool from the Analysis Tools toolbox. Click on the Locate button. This will take you to the tool within ArcToolbox.

- 7) Right-click on the Select tool and choose Help.

Scroll down to the **Scripting Syntax** part, take a look at the table for the Parameters, Explanation, and Datatype. Take a look at syntax for the Select\_analysis tool and copy and paste this to your script. Your code in the script window should look something like this.

```
#Select Red Butte Creek
#Usage: Select_analysis (in_features, out_feature_class,
#where_clause)
```

Now take a look at the **Script Example**. (You may also want to put this comment in your script --Note we're copying all of this because we're new at this and I recommend that when you're leaning its better to have too much comment than too little.)

```
# For shapefile expression, fields are double quoted (")
and text values are single quoted (')
```

We note from the table that DataType for the where\_clause is an SQL Expression. From the Explanation in the table, we are told that the field name for shapefiles has double quotes. The attribute we are selecting on has single quotes (if you didn't already know this, you could click on the SQL Reference link in the table. In other words, the SQL expression under normal conditions (not in a script) would be:

```
"GNIS_NAME" = 'Red Butte Creek'
```

In Python the entire expression must be enclosed in single quotes, something like

```
'"GNIS_NAME" = 'Red Butte Creek''
```

However, Python will be confused by this, so we use what is called an “escape character.” By putting the \ before the single quote, we are saying ignore the single quote as python syntax.

- 8) Type in the code for the Select\_analysis tool as follows. [Note you can also think of this as “using a *method* that is available on the gp object.”]

```
gp.select_analysis ("flowline.shp", "RBCreek.shp", ' "GNIS_Name" = \'Red Butte Creek \' ')
```

- 9) Save the code and run the script.

- 10) Now, do the same for the Buffer tool from the analysis toolbox—find the tool in ArcToolbox, go to Help, and scroll down to they scripting syntax, then copy and past the syntax as a comment in the script window. Then write the line of code to execute the tool (note if you’re not using the optional Parameters, you don’t have to put them in (e.g. line\_side,line\_end\_type, dissolve\_option,dissolve\_field). Your code should look something like this:

```
# Buffer the selected creek
# Buffer_analysis (in_features, out_feature_class,
buffer_distance_or_field, line_side,line_end_type,
dissolve_option, dissolve_field)
gp.buffer_analysis ("RBCreek.shp", "RBCreekBuff.shp", "100
feet")
```

- 11) When the script is executed, it tells you the RBCreek.shp already exists. Type the following just below the gp.workspace line.

```
gp.overwriteoutput = 1
```

- 12) Do the same for the Polygon to Raster tool—find the tool in ArcToolbox, go to Help, and scroll down to they scripting syntax, then copy and past the syntax as a comment in the script window. Then write the line of code to execute the tool. Your code should look something like this:

```
#Convert bufferarea to raster
#PolygonToRaster_conversion (in_features, value_field,
out_raster_dataset, cell_assignment, priority_field,
cellsize)
gp.PolygonToRaster_conversion("RBCreekBuff.shp", "FID",
"BuffGrd", "CELL_CENTER", "NONE", "30")
```

- 13) Finally, it’s good practice to delete the gp object at the end of the script, so at the very end of the script enter the following code: del gp.

- 14) Run the code. Remember you’ll need to pass the arguments in the Run Script dialog. Each argument is separated by a space, so you enter something like: 500 feet.

- 15) Your code should look something like this, and you should have a pretty good idea what it means...

```

# Author: John Lowry
# Date: Jan. 15, 2008
# Purpose: Lesson 2b example of a simple python script
#####

# Import the arcgisscripting module
import arcgisscripting

# Create the Geoprocessor object
gp = arcgisscripting.create()

# Set the current workspace & set overwrite on
gp.workspace = "C:\john\WILD6900_ArcGISPython\Lesson2_results"
gp.overwriteoutput = 1

# Select Red Butte Creek
gp.select_analysis ("flowline.shp", "RBCreek.shp", ' "GNIS_NAME" = \'Red
Butte Creek \' ')

# Buffer the selected creek
gp.buffer_analysis ("RBCreek.shp", "RBCreekBuff.shp", "100 feet")

#Convert bufferarea to raster
gp.PolygonToRaster_conversion("RBCreekBuff.shp", "FID", "BuffGrd",
"CELL_CENTER", "NONE", "30")

#Free memory
del gp

```

## Assignment 2c: Write a script with a List Loop and learn about commenting code

- Start with the code provided in lesson2c.py. Some comments are provided to you. Fill in the header comments and add any additional comments you think are important.
- Use a List Loop with the Slice tool to create 3 elevation zone grids for 5, 10 and 15 zones.
- Hint 1: You'll need to check out the spatial analyst license to use Slice.  

```
# check out spatial analyst license  
gp.CheckoutExtension("spatial")
```
- Hint 2: the items in the list loop will look like this [5, 10, 15]. However this presents a challenge when you create the output, because each of the output rasters should have a new name (e.g. slice5, slice10, slice15). The items in the loop are integers and you can't concatenate an integer to a string, so you'll have to convert the integers to strings. Here is some partial code to get you started.

```
# Set the list variable  
zoneList = _____  
# Use List Loop to create a grid for each slice zone in the zoneList variable.  
for ____ in zoneList:  
    _____ = str(zone)  
    gp.slice_sa("elev30m", "Slice"+_____, zone)  
    print "_____"
```