GIS Services Workshops
Alkek Library - Texas State University

# Introduction to Python Scripting II

**INTRODUCTION:** This session is the continuation of the previous session and the goal is to implement a GIS Analysis by applying your previous knowledge in Python Scripting. *Some background knowledge in GIS is necessary to fully understand this lesson*.

## Understanding Class and Modules

Last time, we touched on class and modules briefly. Here is a quick recap:

- A **class** is like the "blueprint" that defines how to create an object, the properties and methods available to that object, how the properties are set and used, and what each method does.
- **Modules** are simply a collection of various classes. You import modules into your code to tell your program what objects you'll be working with. You can write modules yourself, but most likely you'll import from other sources. Although, Python recognizes various kinds of modules, they aren't imported by default thus, the need for the programmer to import them. For best results, import only the modules needed for a particular project. Some popular modules include: **os** (permits access to functions in the operating system); **random** (allows for generation of random numbers); **csv** (allows the reading and writing of spreadsheet files in comma-separated value format); **math** (permits access to advanced math operations) and **arcpy** (for undertaking GIS analyses – the focus for today).

## Writing and Editing Python Scripts for GIS Projects

GIS programs such as ArcGIS and QGIS have python fully integrated into them. In this task, we will focus only on ArcGIS. The python module for ArcGIS is called **arcpy**, and it is arguably the most widely recognized GIS module currently. Search for a tool or function at http://resources.arcgis.com/en/home/ to learn about their associated syntax.

Python codes in ArcGIS, can be written, edited and executed in different ways including:

I. ArcGIS Python Window within ArcMap or ArcCatolog
II. Field Calculator
III. A stand-alone script using an IDE such as IDLE, PyScripter, PyCharm and PythonWin

**Dataset**: To continue, first download the zipped file on the workshop sign-up webpage onto your storage device and then unzip it.

## Running Python in ArcGIS Python window

You can type in simple Python commands in this window without writing full permanent scripts. Let's explore it:

- Go to Start Menu → ArcGIS → Click ArcMap 10.7 to open it.
- Use the **Add Data** tool  to add the *Campus_Roads_2013.shp* dataset from your unzipped folder.
- On the **Standard toolbar**, click the **Python window** button  . Drag the opened window to the top or bottom of the screen to dock it.
- Type the following in the Python window (ignore the **>>>.** These are included to show you where a new line begins in the Python window.)

```
import arcpy
arcpy.Buffer_analysis("Campus_Roads_2013", "Campus_Roads_buffer", "100
Feet", "", "", "ALL")
```

  Note: After you type **…. _analysis(**, an expected syntax along with it parameters appear in the window to the left. This provides useful hints about what to include in your script. Note that, in our script, some parameters are simply represented as `""`. This signifies, no input will be provided here. In such cases, the tool will stick to the default option. With the cursor at the end of the script, hit **Enter** on your keyboard to execute the script.
- When completed, the output will be added to your map. If not, find and add the output from the default directory for your map; it will likely look like C:\\Users\\<your netID>\\Documents\\ArcGIS\\Default.gdb\\

You can also get the code for a tool after running it through either of these methods:

- Go to the Geoprocessing tab → Results. In the Results window, find and right click the completed run tool you want →"Copy Python Snippet" → Paste the code into Python Window → Hit Enter on keyboard to run it.
- Go to the Geoprocessing → Results window → find the completed tool run you want → left-click and hold and then drag into the Python Window → Hit Enter on keyboard to run it.

YOUR TURN: Use any of the above methods to execute a buffer on the campus roads using the following parameters:

- Output name: TXST_rd_buff
- Buffer distance: 75 feet.
- Line side: LEFT
- Dissolve option: NONE

Now, let's try another commonly used function in ArcPy: the list function. There are actually several of them. They are generally used to obtain a list of items such as files, feature classes, tables etc. in a workspace. See the links below for a complete list of them:

- http://pro.arcgis.com/en/pro-app/arcpy/get-started/listing-data.htm
- http://desktop.arcgis.com/en/arcmap/latest/analyze/arcpy-functions/alphabetical-list-of-arcpy-functions.htm

In this workshop, we will try only the **ListFeatureClasses** function: it returns the feature classes in the set workspace. In this function (like its relatives), one needs to set the workspace environment first before utilizing it. Here's how:

```
import arcpy
arcpy.env.workspace = "<type workspace pathname here>" #See HINT below.
```

HINT: There are three ways of specifying the pathname for your workspace or directory. Choose one.

- path = "C:\\folderName\\"
- path = "C:/folderName/"
- path = r"C:\folderName\"

After the above code, type this:

```
fcList = arcpy.ListFeatureClasses()
print fcList
```

Note the parameters in the expected syntax; you can delimit your search using wildcards (*) and feature type.
YOUR TURN: List only the line feature class(es) in your workspace. HINT: Change the second parameter in the function to "Line".

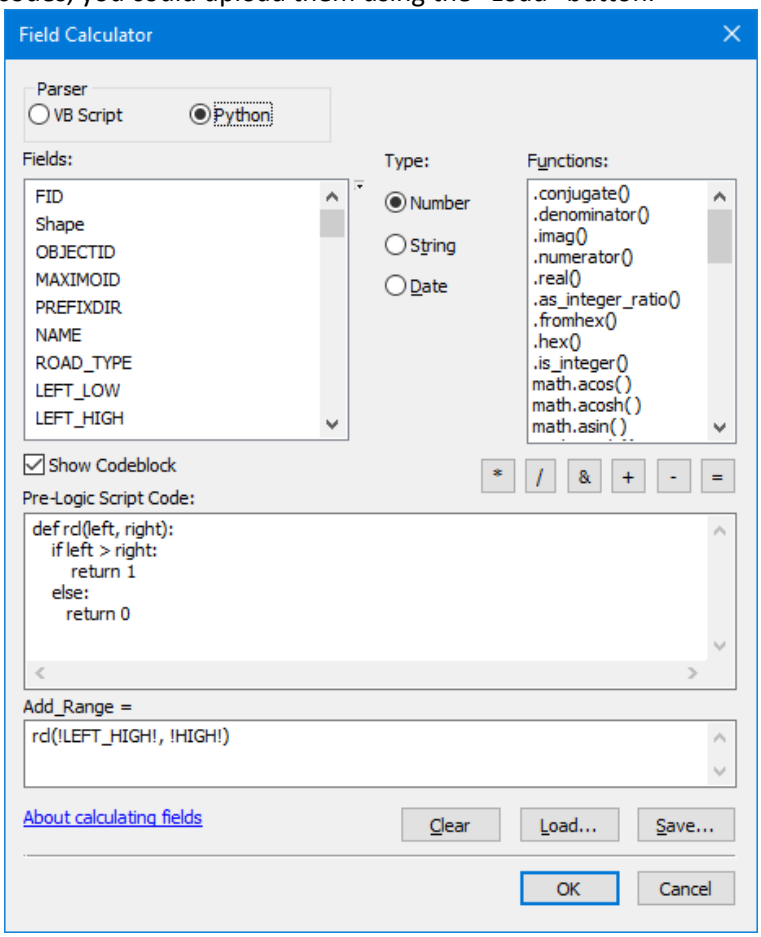
Running Python in Field Calculator

Python can also be executed in a field calculator, specifically, the parser box. In this example, we want to find which roads have left address range (LEFT_HIGH) exceeding the right side (HIGH). Where it exceeds, we will assign 1, else 0.

- Open the attribute table of the campus roads → Click on Table Options → Add Field. Name the field Add_Range and set the type as Short Integer.
- Right-click the new field and choose Field Calculator → Select the radio button for Python in the Parser box and Check the box "Show Codeblock".
- Type in the code as shown in the figure below. Notice that, we are employing an if/else statement which we learned about in Part I here. In view of that, be careful with your **indentation**. You can't use tab key here; only the space bar. Consistency is required here.
- When you are done, click OK to run it.

# Introduction to Python Scripting II

Sometimes you may get error messages. Don't panic. Check your "Results" window (Geoprocessing → Results ) for the kind of error message to allow you to trouble shoot. You may also want to save your code; use the "Save" button to do that. If you have existing codes, you could upload them using the "Load" button.




YOUR TURN: Run a similar analysis but in a new field, with this condition (HINT: use elif and "=="):

- If right address range (HIGH) exceed the left side (LEFT_HIGH), assign **1**
- If right address range (HIGH) equals the left side (LEFT_HIGH), assign **0**
- Anything else, assign **-1**

Running Python in a stand-alone script

Outside ArcMap, one can also use IDEs (Integrated Development Environments) like PyCharm, PyScripter and PythonWin to write, edit and run python scripts. But writing a script for a complex project from scratch can be quite challenging especially when you are a beginner in programming. A way to ease the challenge is by using model builder to implement the project and then exporting the model as a python code. The code could then be edited and run in an IDE.

Challenge for the day

1. Use model builder to create a model that:
   - Selects individual roads,
   - Creates a buffer around each of them and saves them individually.
   - Export the model as a python script.

2. Use PyScripter to edit the exported code so the process will be fully automated. See code below for help. You may also use the **.txt** file called "Challlenge. Open it in PyScripter and save it as a **.py** file.

```
# ---------------------------------------------------------------------------
# Name: Automate_buffer.py
# Created on: 2017-03-07 02:56:08.00000; Originally generated with ArcGIS/ModelBuilder
# Author: Nathaniel Dede-Bamfo
# Description: These lines of code automate a buffering process of all roads on campus based on specific measurements defined by a field.
# ---------------------------------------------------------------------------
# Import arcpy module
import arcpy
from arcpy import env

# Environment setting
path = "E:\\Training_n_Workshops\\Labs\\Completed\\Python_Scripting\\TXST_Data" # Work directory. Replace with yours. Warning! Do not use a network drive
                                                                                # as your directory. Due to network fluctuations, your program may crash.
arcpy.env.workspace = path
arcpy.env.overwriteOutput = True # To allow overwrite of previously created outputs

# Define Local variables:
myFC = "Campus_Roads_2013.shp"
myLayer = "lyr1"
myLayer2 = "lyr2"

# Process: Make Feature Layer
arcpy.MakeFeatureLayer_management(myFC, myLayer)

# Introduce a Cursor
cur = arcpy.SearchCursor(myLayer) # It allows a sequential search of the attribute table record by record

y = 1
for row in cur: # A for loop
   fidField = row.getValue("FID") # The fid field to be used in the query
   name = row.getValue ("NAME") # The name field to be used in naming each road buffer

   # Define local variables for Attribute Query tool
   myInput = myLayer # The layer to be used by the attribute query tool
   sQL = "FID = %s" % (fidField) # SQL for selecting attributes

   # Process: Select Layer By Attribute
   arcpy.SelectLayerByAttribute_management(myLayer, "NEW_SELECTION", sQL)

   # Process: Make Feature Layer (2)
   arcpy.MakeFeatureLayer_management(myLayer, myLayer2)

   # Process: Buffer
   output = name[:2] + str(fidField)+ "_"+ "rd_buff"   # Slicing; use first two letters of road name, the fid
                                    # and suffix 'rd_buff' to name the output
   print "Now processing.... " + output # String; just a statement
   arcpy.Buffer_analysis(myLayer2, output, "Buff_Dist", "FULL", "ROUND", "ALL")

   y +=1 # Counter indicating increments of 1

# Delete cursors to release locked schemas
del cur

# A Greeting to indicate all is done.
print "Mission accomplished, Boss!" # String; just a statement
```

References:

1. Allen, David W. 2014. GIS Tutorial for Python Scripting. Redlands, US: Esri Press.
2. An Informal Introduction to Python: https://docs.python.org/2/tutorial/introduction.html
3. ArcGIS Resources: http://resources.arcgis.com/en/home/
4. Python – Tutorial: https://www.tutorialspoint.com/python/index.htm
5. Python – Tutorial: http://www.w3resource.com/python/python-tutorial.php
6. Sheehan, Daniel 2013. Python Programming for Arcgis 1. https://libraries.mit.edu/files/gis/PythonProgrammingforArcgis.pdf

That's all for today. Thanks for coming and hope to see you in our furture workshops.