Python Programming for Arcgis

Daniel Sheehan
dsheehan@mit.edu, gishelp@mit.edu
http://libguides.mit.edu/gis

10:00 AM - 4:00 PM
January 11, 2018
Arcgis 10.6.1 and Python

• You need a normal install of Arcgis 10.6.1 or greater. This install includes Python 2.7.2 and includes the IDLE interface, which we will use.

• Other installs of Pythons will not have the ‘arcpy’ module and will not work.
  – Check that now if you are working on a personal laptop ...
Goals for the workshop

• Learning enough Python to
  – Access Arcgis commands through Python
  – Access individual records from attribute table
  – Access individual geometries for use in geoprocessing; reading attribute tables
  – Automate some mapping functions

• Develop the ability to record and document your geoprocessing tasks
Outline

• Introduction to Python and Arcgis
• Programming Principles and Modules
• Model Builder
  – Making a model
  – Modifying Model Builder output (your first Arcmap/Python program)
• Working with attribute tables
• Automating mapping
Outline

• Introduction to Python and Arcgis
• Programming Principles and Modules
• Model Builder
  – Making a model
  – Modifying Model Builder output (your first Arcmap/Python program)
• Working with attribute tables
• Automating mapping
Python + Arcgis

- Python can interact with Arcgis and be used to repeat many types of analyses and change maps.

- Why Python?
  - It is an integral part of Arcgis
    - Geoprocessing functions
    - Mapping function (select, applying symbology)
  - Easy to read syntax
  - Large user community
Arcgis and Python Help

• Use the Arcmap 10.6.1 Help!

• Getting started with mapping tools in Arcgis and Python:

Python documentation is found at

http://docs.python.org and look for Python 2.7
(used in Arcgis 10.6.1)

Also look Python and Arcgis help at
http://gis.stackexchange.com

And learn to use Arcmap Help (open it now from Arcmap)
Logistics

- We will be using the IDLE programming environment

- Windows: START -> Programs -> Arcgis -> IDLE (Python GUI)

- We are using Arcgis 10.6.1 on lab computers and assume that you are using 10.6.1 if you are using your own laptop but other versions may work
Learn by doing

• Try every line of code in the slides
• Create a new file for each different slide to create a record for yourself *(this is not optional)*
• Use the Python Help and Arcgis Help and ask questions

To create a Python file from IDLE: open new file and save as *.*py *(extension is necessary!)*
Warning: beware when copying from slides

• Double quotes aren’t the same in PP as in IDLE

• Maintain indents so logic of the program remains as intended

• And don’t open shapefiles that are open in Arcmap with a cursor (this will come later today)
Outline

• Introduction to Python and Arcgis
• **Programming Principles and Modules**
• Model Builder
  – Making a model
  – Modifying Model Builder output (your first Arcmap/Python program)
• Working with attribute tables
• Automating mapping
Programming concepts

• Variables

• Control Structures (IF statements and FOR loops)

Python is case sensitive and reads whitespace for defining programming blocks – use space bar, not tabs.
The Print Function and Strings

# this is a comment
print "hello world"

# this also works
print ‘hello world’
Or
Print ‘hello’ + ‘ ‘ + ‘world’

""" Alternative
Commenting
Style """"
# this is a variable that contains a string
name = "Daniel"
print (name)
Integers and Floats

# declare variables
int_sample = 10
float_sample = 10.0

# printing variables
# cast non-string variable as a string using str()
print “The value of this integer is: “ + str(int_sample)
print “The value of this float is: “ + str(float_sample)
Lists = []

An ordered set of elements enclosed in square brackets. Zero based (first element is accessed by typing list[0]).

# This is a list []
animals = [ 'dog', 'cat', 'horse', 'seal' ]

print animals[3]

animals.append('owl')

Lists are mutable
Tuples = ()

Zero base (first element of a non-empty tuple is always tuple(0))

# This is a tuple
Countries = (‘Costa Rica’, ‘India’, ‘Abu Dhabi’)

# You cannot change a tuple (immutable)
String processing – slicing a string

# slicing a string
Name = ‘Massachusetts’

#Fenceposts – starting at index 1 and ending
# before index 2
Name[1:2]

# find the first 4 characters of Name
Name[:4]
Removing the first and last characters of a string

stateName = "Massachusetts"
print stateName[1:-1]
Concatenating strings with numbers

- \( I = 500 \)
- \( \text{shapefileName} = \text{"buffer"} + \text{str}(I) + \text{".shp"} \)
- print shapefileName
if statement

x = 2

# Condition checks if statement is true

if x == 1:
    print ‘x is 1!’
if / elif / else statement

x = 2

# Condition checks if statement is true
if x == 1:
    print 'x is 1!'
elif x == 2:
    print 'x is 2!'
else:
    print 'I\'m sorry Dave, I don\'t know what to do'
for loop

```python
for i in range(3):
    # convention is to use 4 spaces to indent
    # python reads whitespace at the beginning of a line
    print i
```

Python, like most programming languages, uses arrays that are zero based.
For loop with a list

You can run a for loop based on the contents of a list:

```python
myList = [100, 200, 300]
for i in myList:
    print(i)

myList2 = [“Vermont”, “Maine”, “New Hampshire”]
for i in myList2:
    print(i)
```
while loop

# define j
j = 1

# ‘while’ less than some condition
while j < 3:
    print j
    # increment j
    j += 1

How many times is “j” printed?
Three ways to access a folder

# Accessing a folder and dealing with “\” as a control character in Python

path = “C:\folderName\”

path = “C:/folderName/”

path = r”C:\folderName\”

** use the first option for this workshop
Importing Modules

Use the *import* command:

```python
# count the number of files in a directory
import os
path = "c:\\..."
files = os.listdir(path)
print len(files)
```

A module is a list of Python programs that can be accessed. Commonly used modules are *os*, *sys*, *glob*. 
In computer programming, in particular in a Unix-like environment, glob patterns specify sets of filenames with wildcard characters. For example, the Unix command mv *.txt textfiles/ moves (mv) all files with names ending in .txt from the current directory to the directory textfiles. Here, * is a wildcard standing for "any string of characters" and *.txt is a glob pattern. The other common wildcard is the question mark (?), which stands for one character.

From Wikipedia.

glob is shorthand for global
Glob in Python

Python has a glob module in the standard library which performs wildcard pattern matching on filenames,[21] and an fnmatch module with functions for matching strings or filtering lists based on these same wildcard patterns [22] Guido van Rossum, author of the Python programming language, wrote and contributed a glob routine to BSD Unix in 1986.[23] There were previous implementations of glob, e.g., in the ex and ftp programs in previous releases of BSD.

From wikipedia.
Exercise 1: Try using the glob module

import glob # use the glob module
path = (the folder you copied earlier)
# loop through all files
ListofFiles = glob.glob(path + "*"

for i in ListofFiles:
    print i

Try replacing ‘*’ with ‘*.shp’
Reading from a file

• Open file
  
  import os
  
  inFile = open("path", "r")

• Read the file
  
  for line in inFile:
    print line

• Close file
  
  inFile.close()
Lines are read as strings

• Just as lines are written as strings, they are read as strings
• You will need to split strings to read
  – ListFromString = line.split("","")

• Print elements of a list:
  – print ListFromString[0]
  – print ListFromString[1][:-1]
Exercise 2

• Use the code from the previous slides to read JerseyCityStations.csv. This is a comma delimited file with a header line. Look at this file in Excel but don’t save it. Read the file and print out the Rail Line and the Station with a comma between them.
Writing files

• Import os module:
  – import os

• Create file with open function:
  – outFile = open("path", "w") (also can use "a" for appending to a file)

• Write lines with write method:
  – outFile.write("string" + "\n")

• Close file with close function:
  – outFile.close()
Writing to a text file

“w” means create a new file, “a” appends to an existing file, will create it if it doesn’t already exist

outFile.flush() # flushes buffer
Exercise 3

• Add code to Exercise 2 that writes the Rail Line and Stations to a new file.
Outline

• Introduction to Python and Arcgis
• Programming Principles and Modules
• Model Builder
  – Making a model
  – Modifying Model Builder output (your first Arcmap/Python program)
• Working with attribute tables
• Automating mapping
Importing the Arcgis module

At 10.6.1:

```python
import arcpy
```

And start coding. Or you can build a model and export the model to a Python script.
Building models

- Built in Arcmap
- Add a Model to the Toolbox in your folder
- Drag and drop tools into the canvas
- Double click to open interface for tools
- Be sure to data from the file location, not the version in the map
- Right click on output files and select Add To Display if you need to add these to your map
Exercise 4: ModelBuilder

Using ModelBuilder and referring to previous slides:

• Buffer PATHstation.shp (5280 feet)
  – Units of data is feet
• Clip censusTracts.shp with buffer
• Run Model (under Model menu item, click Run)
• Export model as ‘PythonCode.py’
• Check variable names
Exercise 4

• The follow slides show you how to fill in the forms.
• You can export models to Python from Model -> Export – Python Script – give it a .py extension
Buffer settings

Input Features
PATHstation.shp

Output Feature Class
C:\Users\dsheehan\Desktop\IAP2019\PythonData\testData\buffer5280.shp

Distance [value or field]
- Linear unit: 5280 Feet
- Field:

Side Type (optional)
- FULL

End Type (optional)
- ROUND

Method (optional)
- PLANAR

Dissolve Type (optional)
- NONE

Dissolve Field(s) (optional)
- FID
- OBJECTID
- Id
- StatName
First make the model

- Add the workshop toolbox from your folder to the Arcmap toolbox (right click on the main toolbox in the toolbox window and select Add Toolbox)

- Create a model by right click on your toolbox and then click on New then Model
Model Builder
Clip settings

Input Features
C:\Users\dsheehan\Desktop\IAP2019\PythonData\PythonGISdata\censusTracts.shp

Clip Features
buffer5280.shp

Output Feature Class
C:\Users\dsheehan\Desktop\IAP2019\PythonData\testData\clip5280.shp

XY Tolerance (optional)
Catching exceptions

try:

<your code>

except:
    print arcpy.GetMessages()
    raise
Overwriting files

• from arcpy import env

• env.overwriteOutput = True
  – But files can’t be open in Arcgis when you try to overwrite

Why would you want to overwrite a file?
Exercise 5: Convert ModelBuilder Code into a loop

• Using the code from ModelBuilder
• Iterate through a loop 3 times, buffering 5280, 10560, and 15820 feet
• Clip the censusTract layer with buffer around the PATH station. You should have 3 new pairs of shapefiles
Outline

• Introduction to Python and Arcgis
• Programming Principles and Modules
• Model Builder
  – Making a model
  – Modifying Model Builder output (your first Arcmap/Python program)
• Working with attribute tables
• Automating mapping
Warning: beware when copying from slides

- Double quotes aren’t the same in Power Point as in IDLE
- Maintain indents so logic of the program remains as intended
- And don’t open shapefiles that are open in Arcmap with a cursor
How do you access an attribute table?

• With Cursors
  – Think of your cursor in Excel
  – You decide which columns (or fields) you want to access
  – The content of the fields are held in a list, by row (row[] in the exercise)

In the next exercise, you will access the attribute table and pull out one column (SE_T037_00, the size of the labor force in each census tract) and sum up the column
How to sum a column

• Set a variable to hold the running sum, initialized to zero
  
  \texttt{sumLaborForce} = 0

• Add the current row to the running sum (notice the shorthand)
  
  \texttt{sumLaborForce += row[0]}

• Print out the running sum after you read all records
  
  \texttt{print sumLaborForce}
Helpful information

- Find the metadata.txt file in your folder – what are the SE_T037_00 and Urate columns?
Accessing the Attribute Table

Shapefile = ""
# prints the length of all line segments in an attribute table
import arcpy, os
sumLaborForce = 0.0
try:
    with arcpy.da.SearchCursor(shapefile, ("SE_T037_00:") as cursor:
        for row in cursor:
            sumLaborForce += row[0]
    del cursor
except:
    print arcpy.GetMessages()
    raise
print sumLaborForce
Exercise 6

Use the code from the previous slide in a new .py file. Set the Shapefile Variable to the complete path to the shapefile (censusTracts.shp) Be sure to print the result.

Once this works, add a where clause to the searchCursor function to find only records where Urate > ‘ 5.0’. Search help for SearchCursor (arcpy.da version) and look for where_clause.

Note that fields need to look like this: “FieldName”
Last steps for Exercise 6

• Using the metadata, find the column that represents the total workforce and the unemployment rate. Sum up both and calculate the unemployment rate for all tracts.
Outline

• Introduction to Python and Arcgis
• Programming Principles and Modules
• Model Builder
  – Making a model
  – Modifying Model Builder output (your first Arcmap/Python program)
• Working with attribute tables
• Automating mapping
Make a project file for the mapping exercise

- Create a new project file with the streets layer and the PATH station
- Save it as Python.mxd in your folder
And now to mapping

• Works with a map project file (*.mxd)
• Requires access to Data Frame
• You can add and remove layers (and change the transparency of the layers)
• You can export the map to JPG files (and other formats)
• You can save mxd files after you have added and removed layers
Find the map document and data frame

```python
import os
import arcpy
from arcpy import env
env.overwriteOutput = True

mxd = arcpy.mapping.MapDocument("python.mxd")

df = arcpy.mapping.ListDataFrames(mxd, "Layers")[0]
```
Adding shapefiles

• Shapefiles can’t be added directly, instead you create a layer (as in an Arcmap document)

newLayer = “censusTracts.shp”

arcpy.MakeFeatureLayer_management(newLayer, “myLayer”)
Adding shapefiles, continued

• Next you create a mapping layer from the feature layer you just created. You can make it transparent here

```python
addLayer = arcpy.mapping.Layer("myLayer")
addLayer.transparency = 50
```

• Then you can add that layer to the map by adding it to the data frame

```python
arcpy.mapping.AddLayer(df, addLayer)
```
You can change a query definition

To find a layer, get the list of layers:

```python
for lyr in arcpy.mapping.ListLayers(mxd):
    print lyr.name
```

And if you get the one you want, you can apply a layer query (don’t use this code, it’s just an example):

```python
if lyr.name == "BikeUserCitiBike12AM":
    lyr.definitionQuery = "\"BikelID\" = 16334"
```
Now you can export the map

Export:

```python
arcpy.mapping.ExportToJPEG (mxd, "map5280.jpg", df)
```

You can remove layer:

```python
arcpy.mapping.RemoveLayer(df, "myLayer")
```

Or save map:

```python
mxd.save()
```
And delete your variables!

• Variables exist in Python until you close your window, using memory and causing problems when trying to save the map document

del mxd
del df
Exercise 7

• Create maps for all new clipped census tracts you created earlier. Use a loop to walk through each of the new layers. Some lines of code should be before the loop (opening the MXD file and getting the data frame) while creating the JPEG files should be inside the for loop.
Questions, project issues?