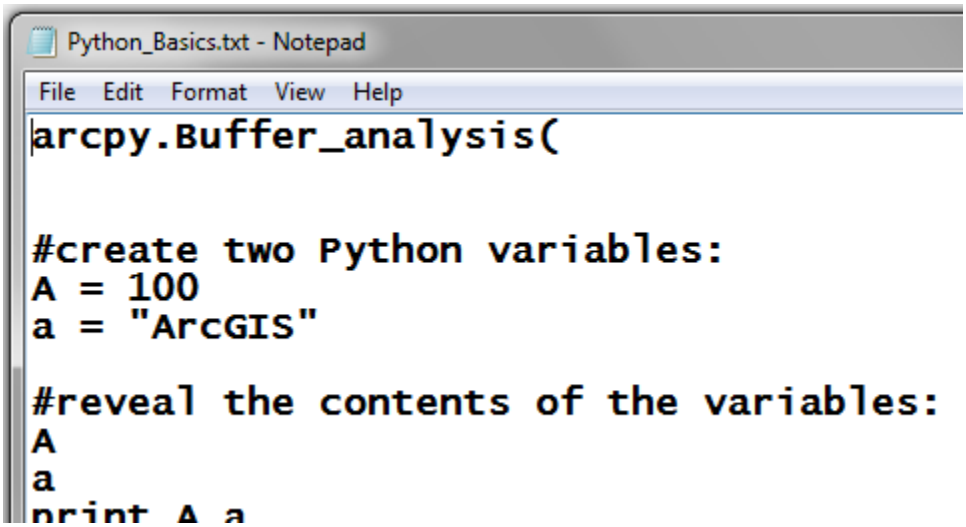
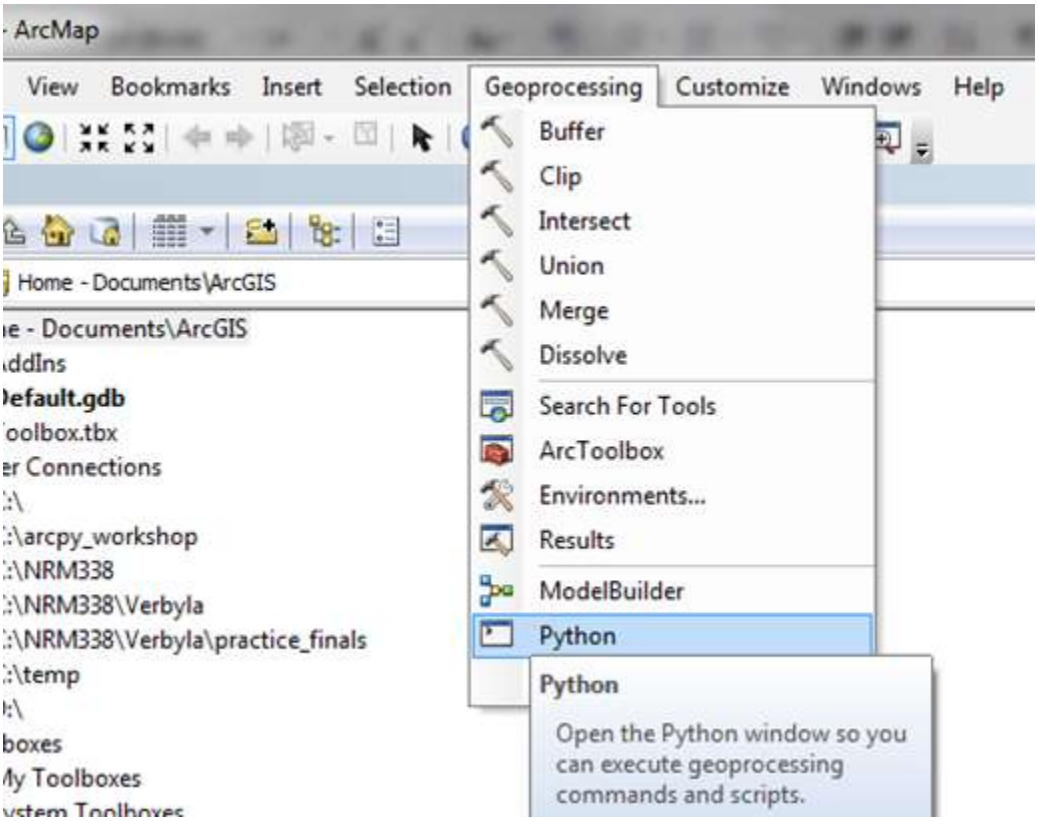
Session 1: Python Basics

In this session we will cover some Python scripting basics. For this session, copy ***(Ctrl-C)*** and paste ***(Ctrl-V)*** Python script lines from the text file ***Python_Basics.txt.***

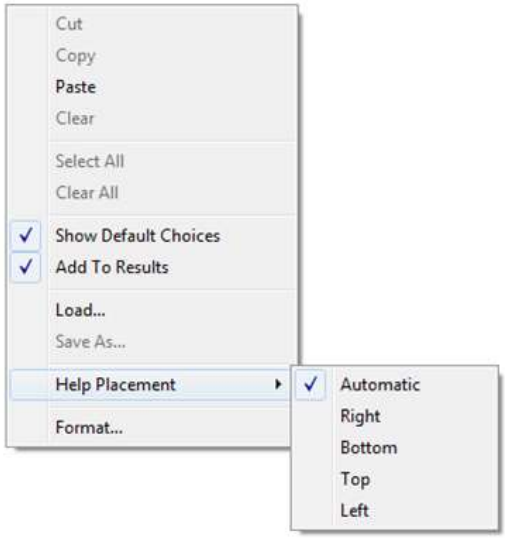


Start by opening the Python window under the Geoprocessing menu…

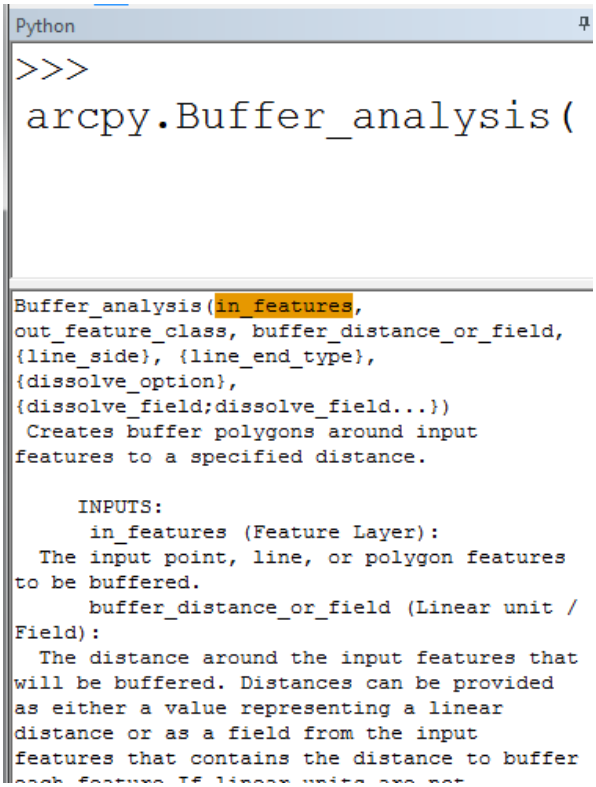


ArcGIS Python Window

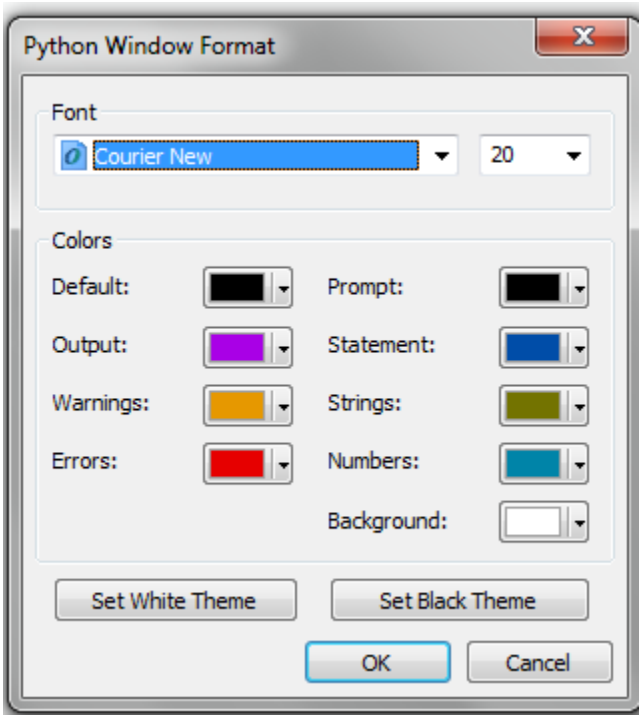
Right mouse click in your ArcGIS Python window.



The Help pane is sometimes hidden…it is useful because when we do arcpy geoprocessing, the Help pane shows the correct syntax for commands as you type them…for example try arcpy.Buffer



You can also specify the viewing properties of you Python window, by right mouse-clicking and selecting Format…



Python Comments

Python is scripting language where the ***# symbol is a comment that is ignored*** by the Python interpretor and is not executed. For example, only A=100 and a=”ArcGIS” are executed:

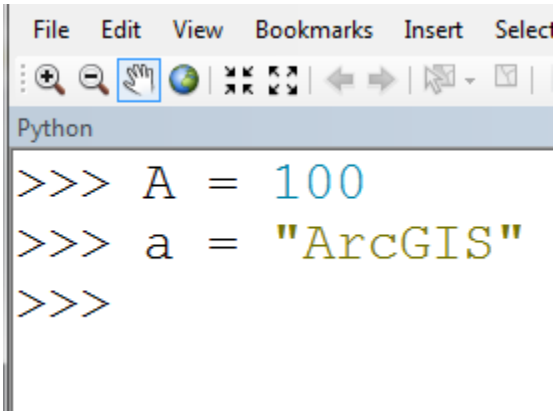
```
#create two Python variables:

A = 100 #integer value of 100 is now inside variable named A

a = "ArcGIS" #text string of ArcGIS characters is now inside variable named a
```

Variables as Data Containers

The most basic data container is called a variable. A variable can contain only 1 value. Variable names are case-sensitive. For example, create a variable A and a variable a, and put the value 100 in container A and "ArcGIS" in container a.



You can see the contents of a variable by typing its name or by using the Python print statement:

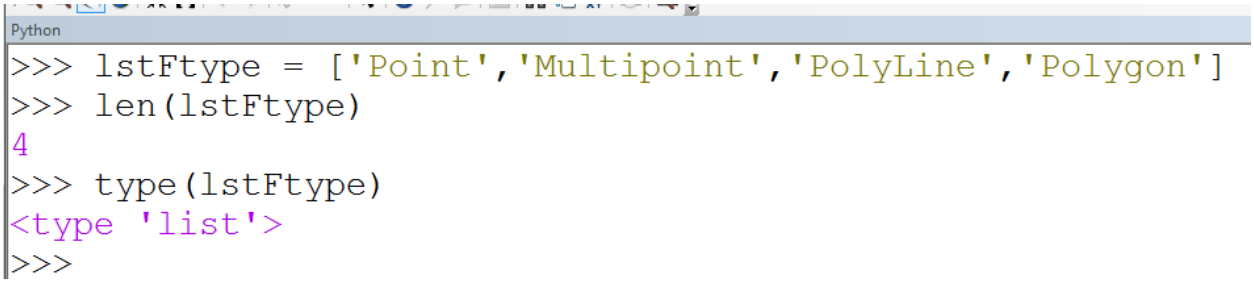
```
>>> A
100
>>> a
'ArcGIS'
>>> print A,a
100 ArcGIS
>>>
```

The type of variable is similar to field types in ArcGIS (integer, text, floating point, date, etc.). You can use the Python type() function to determine the type of any variable:

```
>>> type(A)
<type 'int'>
>>> type(a)
<type 'str'>
>>>
```

Lists as Data Containers

A Python list is another type of data container that can hold more than one value. I often use the prefix lst when I name a list. Try the following:



The Python len() function returns the length of the list or the count of the number of items in the list. Lists can contain values of different types for example integer, floating point, string, boolean values all could be in a list container.

```
>>> lstMixedTypes = [100,100.333,'Text',True]
>>>
```

Each item in a list is indexed with an integer starting with 0 for the first item in the list. Try the following (a # is a comment in Python and is ignored by the Python interpreter):

```
>>> lstFtype[0] #first item in the list
'Point'
>>> lstFtype[1] #second item in the list
'Multipoint'
>>> lstFtype[2] #third  item in the list
'PolyLine'
>>> lstFtype #all items in list
['Point', 'Multipoint', 'PolyLine', 'Polygon']
>>>
```

What does lstFtype[-1] mean?

What error message results from the following Python statement:

```
>>> lstFtype[10]
```

You can also determine the type of each list item or element:

```
>>> lstMixedTypes
[100, 100.333, 'Text', True]
```

Python

```
>>> type( lstMixedTypes[-1] )
<type 'bool'>
>>> type( lstMixedTypes[0] )
<type 'int'>
>>> type( lstMixedTypes[1] )
<type 'float'>
>>> type( lstMixedTypes[2] )
<type 'str'>
```

You can use the .append() function to append values to a list. Try the following:

```
>>> lstMixedTypes
[100, 100.333, 'Text', True]
>>> lstMixedTypes.append(False)
>>> lstMixedTypes.append(2014)
>>> lstMixedTypes
[100, 100.333, 'Text', True, False,
 2014]
>>> len(lstMixedTypes)
6
```

```
>>> lstFtype
['Point', 'Multipoint', 'PolyLine', 'Polygon', 'Network']
```

And you can assign new values to any item in your list. Try the following:

```
>>> lstFtype.append('Network')
>>> lstFtype
['Point', 'Multipoint', 'PolyLine',
 'Polygon', 'Network']
```

```
>>> lstFtype[4] #fifth item in list
'Network'
>>> lstFtype[-1] #last item in list
'Network'
>>> lstFtype[1:-1] #second to last item in list
['Multipoint', 'PolyLine', 'Polygon']
```

```
>>> lstFtype[3] = 'Network'
>>> lstFtype[4] = 'Polygon'
>>> lstFtype
['Point', 'Multipoint', 'PolyLine',
 'Network', 'Polygon']
>>>
```

Dictionaries as Lookup Tables

One handy Python container is called a dictionary…it contains pairs of values where key values are linked to specific information. A dictionary is created using curly braces {keyvalue:companion value,keyvalue:companionvalue} Try the following example:

```
>>> dctLookup = {1:'Fairbanks North
 Star Borough',2:'Denali
 Borough',3:'Mat-Su Borough'}
>>> txtLabel = dctLookup[3]
>>> txtLabel
'Mat-Su Borough'
```

We will use a Python dictionary in the next session when working with Python in the field calculator….

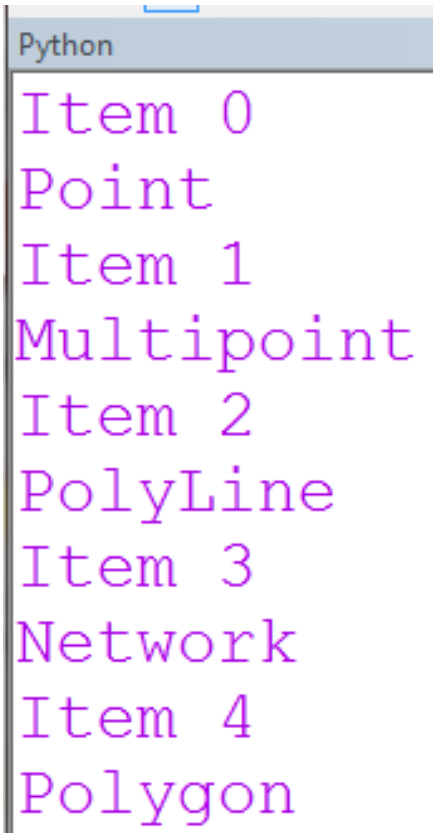
Basic Looping

In this workshop, we will often want to loop through each item in a list. We can do this using a Python ***for item in list:*** loop

Two key points: 1) all ***statements inside your loop must be indented*** the same number of spaces. 2) You must end your for statement with a colon **:**

Try the following:

```
#loop through every item in list:
count = 0
for item in lstFtype:
    print "Item " + str(count) #convert count to string value
    print item #print list item value at count position
    count = count + 1
```



Basic Decisions

Python allows you to ask questions using if(): elif(): else: statements. Each statement within an if/elif/else statement ***must be indented the same number of spaces.*** Try the following:

```
Python
>>> lstFtype
['Point', 'Multipoint', 'PolyLine', 'Network', 'Polygon']
>>> testVar = lstFtype[4]
>>> if (testVar == 'Point'):
...     print "No length or area"
... elif (testVar == 'PolyLine'):
...     print "Length, but no area"
... elif (testVar == 'Polygon'):
...     print "Length and area properties"
... else:
...     print ("testVar is not Point, PolyLine or Polygon")
...
```

See if you can guess what will be printed before you execute this decision.

___*No length or area*

___*Length,but no area*

___*Length and area properties*

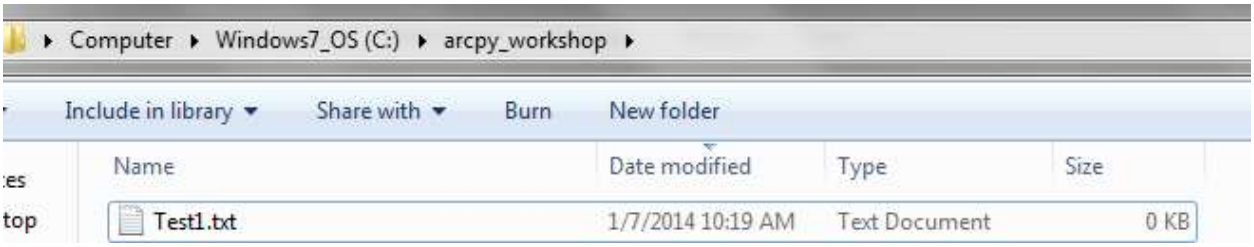
___*testVar is not Point, PolyLine or Polygon*

Writing To Text Files

In order to read or write text files, you need to access a module named os (os stands for operating system, and this module accesses many operating system capabilities such as reading and writing files). To write to a text file, you import the os module, open a new file with write access, write to the open text file, and close your text file when you are done. Create a text file…try the following:

```
Python
>>> import os
>>> os.chdir(r"c:\arcpy_workshop")
>>> txtFile = open("Test1.txt","w")
>>> txtFile.close()
>>>
```

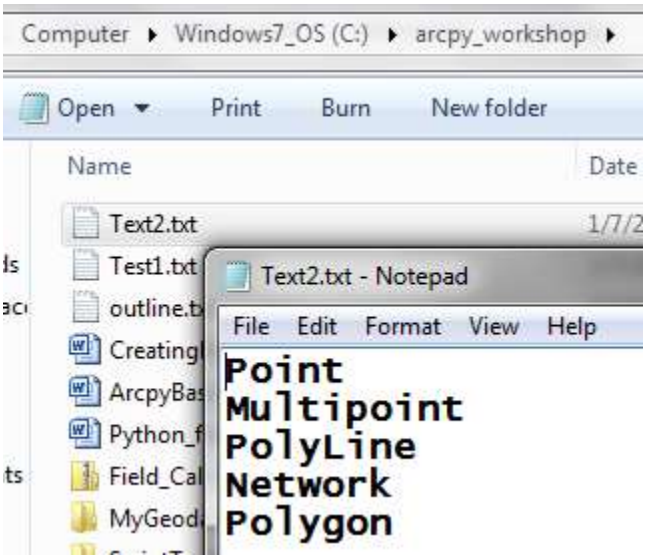
Use windows explorer to see if you created your text file successfully…



Create a text file and write records to the file. One key point, you need to add a newline character and the end of each line. The Python interpreter interprets the string "\n" as new line.

```
>>>
>>> txtFile = open("Text2.txt","w")
>>> for item in lstFtype:
...     strRecord = item + "\n" #add newline character
...     txtFile.write(strRecord)
...
>>> txtFile.close()
```

Use notepad to see the contents of your text file…



When we get to arcpy geoprocessing, we will write the results of geoprocessing operations to a text file…more to come.

Next session…**Session2: Using Python in ArcGIS Field Calculations.**