Introduction to Python for ArcGIS

Description

Programming tools are now a standard feature within GIS software packages and allow GIS users to automate, speed up, and become more precise in their data management and analytic work. This workshop is designed for GIS users who have little to no experience with computer programming and will cover core programming concepts related to GIS using the Python programming language. The workshop will focus on guiding attendees through hands-on exercises designed to provide the essential skills to programmatically manipulate data as part of a GIS workflow. This workshop is designed to be preparation for the afternoon workshop on **Advanced Python for ArcGIS, but may be taken independently**.

Specific Topics Include:

- Core Python programming concepts
- Introduction to ArcPy site package for ArcGIS
- Working with geospatial data using Python and ArcPy
- Simple data management and geoprocessing tasks

Computing and Software Needs

- Your own laptop; computers will not be provided.
- ArcGIS Desktop 10.x (Standard or Advanced preferred)
- Python IDE
    - PyScripter (http://sourceforge.net/projects/pyscripter/files/) (v. 2.6.0 **32-bit** version)
        - *NOTE:* The zip files contain portable versions of PyScripter. No installation is needed. Just unzip the archive and start using PyScripter.
    - IDLE (https://docs.python.org/2/library/idle.html) is also shipped with ArcGIS for Desktop, so it can also be used

Instructor

James Whitacre, GIS Specialist, University Library, University of Illinois at Urbana-Champaign

As the GIS Specialist in the Main Library at the University of Illinois at Urbana-Champaign, Mr. Whitacre's primary role is to provide GIS consultation and research assistance for faculty, staff, and students. Additionally, he teaches a myriad of GIS workshops for beginner to advanced users and helps manage the Library's GIS data and software assets. He is also a central resource for the GIS community on campus to promote the use of GIS in research. Mr. Whitacre holds a Master of Science in Geography and was previously the GIS Manager for the Carnegie Museum of Natural History.

I. Introductions and Outline

Outline

Data and Software Setup

What is Python?

- Print Statement
- Variables
- Basic Data types
  - Strings
  - Numbers
  - Lists
  - Tuples
  - Dictionaries
  - Booleans
- Data Type Conversions
- Simple Math with Python
- Python Basic Syntax
- Conditional Statements
- Functions
- Loops

Calculate Fields Using Python

Break

Introduction to ArcPy

- What is ArcPy?
- Modules vs. Site Packages
- `import` Statements
- Utilizing the ArcPy ArcGIS Desktop Help Documentation

Using ArcPy in the ArcMap Python Window

- The Python Window
- Describing Data with ArcPy
  - System Paths vs. Catalog paths
- Listing Data with ArcPy
  - Environmental Settings
  - List comprehensions
- Geoprocessing Tools with ArcPy

Conclusion

II. Data and Software Setup

Download Exercise Data

[https://uofi.box.com/ILGISA-Python (https://uofi.box.com/ILGISA-Python)](https://uofi.box.com/ILGISA-Python)

- Extract the data to a folder on the Desktop or other well-known folder
- For folks who can't download the files, we have it on a flash drive

Python IDE: PyScripter

- Download [PyScripter (http://sourceforge.net/projects/pyscripter/files/)](http://sourceforge.net/projects/pyscripter/files/) (v. 2.6.0 **32-bit** version)
- Open PyScripter: Two Ways...
  - Install it like a normal program
    - Download **PyScripter-v2.6.0-Setup.exe**, run the install file, open program
    - Admin privileges needed
  - Run the portable version
    - Download **PyScripter-v2.6.0.zip**, extract, and run **PyScripter.exe**
    - Admin privileges *NOT* needed
- Already using a Python IDE? If you feel comfortable in your own envirionment, please use it, but there is no guarantee that issues can be supported

- If all esle fails, you may need to use IDLE, which ships with ArcGIS Desktop and is installed automatically

- Let's take a quick tour!
  - Editor
  - Code explorer
  - Interpreter
  - Messages
  - Add line numbers: Tools > Options > Editor Options
  - Change the theme: View > Themes > Choose your favorite!!

Type the code below and run it.

I will give an explanation of concepts, then have code examples that can be run in the IDE in the grey boxes.

Please *type* the code as we go and try to avoid copy and pasting unless instructed otherwise. Typing the code will help you learn it better!

When typing in the IDE, you will likely notice the auto complete functionality. Mastering this will help type code faster and with less mistakes. I will point out tips as we go.

```
In [ ]:  print("hello world")
```

IV. Python Basics

Print Statement

```
In [ ]:  print "Python is so cool!"
```

```
In [ ]:  print("Python is so cool!")
```

Variables

```
In [ ]:  food = "cheese"

         food_count = 6

         print(food)
         print(food_count)
```

```
In [ ]:  # cool tips...

         food, food_count = "bread", 100

         print(food)
         print(food_count)

         food1 = food2 = food3 = "banana"

         print(food1)
         print(food2)
         print(food3)
```

Basic Data Types

Data Type	Examples
String	"cheese" or 'Food Time'
Integer Number	68 or 23456 or 0
Float Number	345.67 or 28.1 or 98.0
Boolean	True or False
List	["apple", "orange"]
Tuple	("apple", "orange")
Dictionary	{"lat":39.799, "lon":-89.64}

Strings

```
In [ ]: print("cheese " + "whiz") # Note the space...
        print("cheese" * 3) # Note no space...
        print("cheese"[1:4])
        print("cheese"[:2])
        print("cheese"[:-2])
        print("cheese"[2:])
        print("cheese"[-2:])
```

Numbers

```
In [ ]: print( 5 + 7 - 3)

        three = 3

        print(5 + 7 - three)
```

```
In [ ]: y = 0 # Run it once to evaluate, then change to 1 and run again; change to []
         and run again

        if y:
            print(True)
        else:
            print(False)
```

Lists

- See https://www.tutorialspoint.com/python/python_lists.htm*
  (https://www.tutorialspoint.com/python/python_lists.htm*)

```
In [ ]: fruitlist = ["apples", "oranges", "bananas"]

        print(fruitlist)

        print(fruitlist[1])

        print(fruitlist[1:3])

        print(len(fruitlist))
```

```
In [ ]: # Iterate over the list...more on this later
        for fruit in fruitlist:
            print(fruit)
```

```
In [ ]: # Change the list...
        fruitlist = ["apples", "oranges", "bananas"]

        fruitlist[1] = "peaches"
        fruitlist.append("cherries")
        fruitlist.remove("apples")
        del fruitlist[1]

        print(fruitlist)
```

Tuples

```
In [ ]: fruittuple = ("apples", "oranges", "bananas")

        print(fruittuple)
        print(fruittuple[1])
        print(fruittuple[1:3])
```

```
In [ ]: fruittuple = ("apples", "oranges", "bananas")

        for fruit in fruittuple:
            print(fruit)
```

```
In [ ]: fruittuple[1] = "peaches" # Invalid syntax with tuple
```

Dictionaries

Note: Dictionaries are good to know about, but we will not work with them in much detail for this workshop.

```
In [ ]: dicttest = {"key":"value", "lat":39.98, "long": -89.65}

        print(dicttest["key"])
        print(dicttest["lat"])
        print(dicttest["long"])

        print dicttest.keys()

        print dicttest.values()
```

Booleans

- The following will be evaluated as `False`:
  - `None`
  - `False`
  - Zero of any numeric type (i.e. `0`)
  - Any empty sequence (e.g. `''`,`()`,`[]`)
  - Any empty dictionary (e.g. `{}`)

```
In [ ]:  print(True)
         print(False)

         a = True
         b = False

         print(a)
         print(b)
```

Data Type Conversion

```
In [ ]:  # Run the following code...What happens?

         x = 99

         print("There are " + x + " files.")

         # Change x to be str(x)
```

```
In [ ]:  x = 99
         s = "99"
         l = [s, x]
         t = (s, x)

         print(type(x))
         print(type(s))
         print(type(l))
         print(tuple(l))
         print(type(t))
         print(list(t))
```

Simple Math with Python

Operator	Explanation	Example	Result
x + y	x plus y	1.5 + 2.5	4.0
x - y	x minus y	3.3 - 2.2	1.1
x * y	x times y	2.0 * 2.2	4.4
x / y	x divided by y	4.0 / 1.25	3.2
x // y	x divided by y (floor division)	4.0 / 1.25	3.0
x % y	x modulo y	8 % 3	2
-x	negative expression of x	x = 5 /n -x	-5
+x	x is unchanged	x = 5 /n +x	5
x ** y	x raised to the power of y	2 ** 3	8

Source: http://desktop.arcgis.com/en/arcmap/latest/tools/data-management-toolbox/calculate-field-examples.htm (http://desktop.arcgis.com/en/arcmap/latest/tools/data-management-toolbox/calculate-field-examples.htm)

When performing field calculations with a Python expression, Python math rules are in effect. For example, dividing two integer values will always produce an integer output (3 / 2 = 1). To get decimal output:

- One of the numbers in the operation must be a decimal value: `3.0/2 = 1.5`

- Use the float function to explicitly convert the value to a float:

`float(3)/2 = 1.5`

`float(3)/2 = 1.5`

Python Basic Syntax

- Variables cannot start with a number or have a space in it

```
1line = 5 # This will not work...

a line = 5 # Neither will this...
```

- Here is a list of common **reserved words**; do **NOT** use these as variable names:

```
and, del, from, not, while, as, elif, global, or, with, assert, else, if, pa
ss, yield, break, except, import,
   print, class, exec, in, raise, continue, finally, is, return, def, for, la
mbda, try

   # Basically, if it turns a color when you are done typing it, don't use it
 as your variable's name!
   # There are likely many more reserved words!
```

- Variable name capitalization matters!!

```
Cat != cat
```

- Indentation matters!! AND...
- Colons matter!!

```
if x
  print(x)

   # Will not work, but this will:
   if x:
       print(x)

   # typically 2 or 4 spaces...4 spaces are prefered (using Tab in the IDE sh
ould do this automatically)
```

- Quotations are a bit tricky, but very cool
  - Single ('), double (") and triple-single and triple-double (''' or """) quotes can be used to denote strings
  - Make sure to end the strip denotation with the same type of quote structure

```
# Examples

word1 = 'Dog'
word2 = "'Dog'"
word3 = '"Dog"'
print(word1, word2, word3)
# The three words above will print: Dog, 'Dog', and "Dog"

words1 = 'That's the dog's toy' # Is a syntax error
words2 = "That's the dog's toy" # Prints: That's the dog's toy

more_words = """She said, "Good dog!" And the dog's tail wagged."""
# Prints: She said, "Good dog!" And the dog's tail wagged.
```

- Backslashes can be confusing...

```
# These are all the same thing...
  "C:\\data\\things"
  "C:/data/things"
  r"C:\data\things" # I prefer this one when working in ArcGIS...I will tell
 you why later
```

- Commenting is great!!
  - Use it to help docuement and explain your code...we will do this throughout the exercises!
  - Comments are not run in code; they are ignored
  - Blank lines are also ignored

```
# This is a block comment
## So is this
### The blank line below this one will be ignored

""" This is good for multi-line block comments
Notice that this line is still a comment
Use block commnets as mush as you need, but not too much
Don't forget to close the multi-line comment"""

s = "something" # This is an in-line comment...use these sparingly i
n your code
```

Conditional Statements and Decision Making

`if` Statements

Operator	Description	Example

Comparison Operators

Operator	Description	Example
==	If the values of two operands are equal, then the condition becomes true.	(1 == 2) is NOT true.
!=	If values of two operands are not equal, then condition becomes true.	(1 != 2) is true.
<>	If values of two operands are not equal, then condition becomes true.	(1 <> 2) is true. This is similar to != operator.
>	If the value of left operand is greater than the value of right operand, then condition becomes true.	(1 > 2) is not true.
<	If the value of left operand is less than the value of right operand, then condition becomes true.	(1 < 2) is true.
>=	If the value of left operand is greater than or equal to the value of right operand, then condition becomes true.	(1 >= 2) is not true.
<=	If the value of left operand is less than or equal to the value of right operand, then condition becomes true.	(1 <= 2) is true.
and	Called Logical AND operator. If both the operands are true then then condition becomes true.	(a and b) is true
or	Called Logical OR Operator. If any of the two operands are non zero then then condition becomes true.	(a or b) is true
not	Called Logical NOT Operator. Use to reverses the logical state of its operand. If a condition is true then Logical NOT operator will make false.	not(a && b) is false
in	Evaluates to true if it finds a variable in the specified sequence and false otherwise.	x in y, here in results in a 1 if x is a member of sequence y
not in	Evaluates to true if it does not find a variable in the specified sequence and false otherwise.	x not in y, here not in results in a 1 if x is not a member of sequence y

Adapted from: https://www.tutorialspoint.com/python/python_basic_operators.htm
(https://www.tutorialspoint.com/python/python_basic_operators.htm)

```
In [ ]:  a = 21
         b = 10
         c = 0

         if ( a == b ):
             print "Line 1 - a is equal to b"
         else:
             print "Line 1 - a is not equal to b"

         if ( a != b ):
             print "Line 2 - a is not equal to b"
         else:
             print "Line 2 - a is equal to b"

         if ( a <> b ):
             print "Line 3 - a is not equal to b"
         else:
             print "Line 3 - a is equal to b"

         if ( a < b ):
             print "Line 4 - a is less than b"
         else:
             print "Line 4 - a is not less than b"

         if ( a > b ):
             print "Line 5 - a is greater than b"
         else:
             print "Line 5 - a is not greater than b"
```

```
In [ ]:  a = 5;
         b = 20;
         if ( a <= b ):
             print "Line 6 - a is either less than or equal to  b"
         else:
             print "Line 6 - a is neither less than nor equal to  b"

         if ( b >= a ):
             print "Line 7 - b is either greater than  or equal to b"
         else:
             print "Line 7 - b is neither greater than  nor equal to b"
```

Functions

```
# Basic funtions syntax

def functionname( parameters ):
    """function_docstring""" # This is optional
    function_suite
    return [expression]
```

Source: https://www.tutorialspoint.com/python/python_functions.htm (https://www.tutorialspoint.com/python/python_functions.htm)

Script Source: https://www.codecademy.com/learn/python (https://www.codecademy.com/learn/python)

```
In [ ]:  def tax(bill):
             """Adds 8% tax to a restaurant bill."""
             bill *= 1.08
             print "With tax: %f" % bill
             return bill

         def tip(bill):
             """Adds 15% tip to a restaurant bill."""
             bill *= 1.15
             print "With tip: %f" % bill
             return bill

         # Change the values a few times to see how the results change
         meal_cost = 100
         meal_with_tax = tax(meal_cost)
         meal_with_tip = tip(meal_with_tax)
```

Loops

for Loops

```
In [ ]:  a_list = ['a', 'b', 'c', 'd']

         for item in a_list:
             print item
```

`while` Loops

```
i = 0 # This is called a sentry variable
    while i <= 10:
        print i
        i += 1 # Increment the sentry variable to ensure the exit condition
```

`i =+ 1` *is another way to increment a numeric value by a constance value (in this case 1).*

See https://www.tutorialspoint.com/python/assignment_operators_example.htm for other similar operations. Python is full of these types of tricks!

```
In [ ]:  i = 0

         while i <= 10:
             print i
             i += 1
```

Calculate Fields Using Python

- **A great place to practice Python in ArcGIS is by calculating fields**
- Python and VB Script can be used to calculate fields
- VB Script is the default, so Python needs to be set as the parser
- Simple expressions use the Expression box
- More complex calculations can use the Pre-Logic Script Code (window) or Code Block (tool) boxes
- **Note: Python can be difficult to debug and check for errors when calculating fields**

Using Python in the Field Calculator Window

1. Open the *Illinois.mxd* ArcMap document

2. Open the Attribute Table for the 'Illinois Counties' Layer

```
We see that there is a field named 'AREA', but we don't know the units.
Note the other fields
```

3. Add a field with the following parameters:

```
Name: AreaTest
Type: Double
```

4. Right-click on the new field header and select Field Calculator

5. Double click any field to add it to the expression box

```
Make sure the VB Script radio button is selected
```

6. Select the Python radio button and add the same field

```
What is different?
Clear the expression box when done
```

7. Click on the 'About calculating fields' link at the bottom left

```
Notice the different examples
Find the 'Code samples-geometry' section
These expressions can be used in lieu of the 'Calculate Geometry' window accessed t
hrough the attribute table
These expressions can also be used in the Calculate Field tool
```

8. Enter the following code in the expression box

```
In [ ]:   """ Note: The following code is intended to be used to calculate an attribute
          table field in
          ArcGIS using the Field Calculator window or Calculate Field tool."""

          # First try square meters

          !Shape.area@SQUAREMETERS!

          # How close are the vales compared to the AREA field?


          # Second, try square miles

          !Shape.area@SQUAREMILES!

          # Are these vales closer to the AREA field? Do you think we can get better?


          # Now try

          !Shape.geodesicArea@SQUAREMILES!

          # Which method is more acurate? Why do you think that is?


          # One last trick, type:

          None

          # This will make all the values NULL in the field...it is good to know about!
```

Field Calculator Window (continued)

Now we know the units, but we want to calculate the population density. We also want to know the difference between the median and average household income.

1. Delete the 'AreaTest' field

```
Click 'Yes' on the warning
```

2. Add two fields with the following parameters:

```
Name: DensitySqmi
Type: Double
Alias: Density Per SqMi

Name: DiffIncome
Type: Double
Alias: Avergae vs. Median Income
```

3. Calculate the field using Python

```
Clear the expression box before each calculation
Make sure the Python radio button is selected
Enter the code below
```

In [ ]:
```python
""" Note: The following code is intended to be used to calculate an attribute table field in
ArcGIS using the Field Calculator window or Calculate Field tool."""

# DensityPerSqmi calculation

!ACSTOTPOP! / !AREA!

###########################################################

# DiffIncome calculation

!ACSAVGHINC! - !ACSMEDHINC!
```

Field Calculator Window (continued)

Now we would like to have three more fields for a short name, a long name, and a name with the area.

1. Add two fields with the following parameters:

```
Field 1:
    Name: NameLong
    Type: Text
    Alias: Long Name
    Length: 100

Field 2:
    Name: NameShort
    Type: Text
    Alias: Short Name
    Length: 50

Field 1:
    Name: NameArea
    Type: Text
    Alias: Name (with Area)
    Length: 100
```

2. Calculate the fields using Python

```
Clear the expression box before each calculation
Make sure the Python radio button is selected
Enter the code below
```

```
In [ ]:  """ Note: The following code is intended to be used to calculate an attribute
         table field in
         ArcGIS using the Field Calculator window or Calculate Field tool."""

         # NameLong calculation

         !NAME! + ", " + !STATE_NAME!

         ############################################################

         # NameShort calculation

         !NAME!.replace(" County", "")

         # Note the first parameter...what do you notice?

         ############################################################

         # NameArea calculation

         !NAME! + " (Area: " + !AREA! + " SqMi)"

         # Did it work? Why or why not? Hint: Check the ArcMap Results window if you ca
         n't figure it out
         # Change the expression to be correct...


         # NameArea can also be calculated this way

         " ".join([!NAME!, "(Area:", str(!AREA!), "SqMi)"])

         # What do you notice that is different with the elements being joined? I'm loo
         king for 2 specific things...
         # Hint: Don't space out on me...and...0, 1, 2, 3
```

Field Calculator Window (continued)

Now we would like to classify the median income levels.

1. Add a field with the following parameters:

```
Name: IncomeLevel
Type: Text
Alias: Income Level
Length: 50
```

2. Calculate the fields using Python

```
Clear the expression box
Make sure the Python radio button is selected
Check the 'Show Codeblock' box
Enter the code below
```

3. When you have completed the calculation in the Field Calculator window, open the result from the Results window

```
How does this look different?
```

```python
In [ ]:  """ Note: The following code is intended to be used to calculate an attribute
          table field in
         ArcGIS using the Field Calculator window or Calculate Field tool."""

         # When calulating a field, a function usually is used in the
         # Pre-Logic Script Code:

         def income_level(median_income, low_income, mid_income):
             """Classifies median income levels given low income  and middle income val
         ues"""
             if median_income < low_income:
                 return "Low Income"
             elif low_income <= median_income < mid_income:
                 return "Middle Income"
             else:
                 return "High Income"


         # Expression:

         income_level(!ACSMEDHINC!, 35000, 60000)
```

Field Calculator Window (continued)

Now we would like to know how many vertices are in each feature...just for fun!

We will use an example from the ArcGIS Desktop help for Calculate Field examples (http://desktop.arcgis.com/en/arcmap/latest/tools/data-management-toolbox/calculate-field-examples.htm#ESRI_SECTION1_C9E46547D3B246C1B89A36B0C84F7BA8). This page is a great resource.

1. Add a field with the following parameters:

```
Name: VertexCount
Type: Long Integer
Alias: Number of Vertices
```

2. Calculate the fields using Python

```
Clear the expression box
Make sure the Python radio button is selected
Make sure to check the 'Show Codeblock' box
```

Copy and Paste the code below (it is for demonstrating that while loops can be used in the Field Calculator)

```
In [ ]:   """ Note: The following code is intended to be used to calculate an attribute
          table field in
          ArcGIS using the Field Calculator window or Calculate Field tool."""

          # Pre-Logic Script Code:
          def VertexCount(feat):
              partnum = 0

              # Count the number of points in the current multipart feature
              partcount = feat.partCount
              pntcount = 0

              # Enter while loop for each part in the feature (if a singlepart
              # feature this will occur only once)
              #
              while partnum < partcount:
                  part = feat.getPart(partnum)
                  pnt = part.next()

                  # Enter while loop for each vertex
                  #
                  while pnt:
                      pntcount += 1
                      pnt = part.next()

                      # If pnt is null, either the part is finished or there
                      # is an interior ring
                      #
                      if not pnt:
                          pnt = part.next()
                  partnum += 1
              return pntcount


          # Expression:

          VertexCount(!Shape!)
```
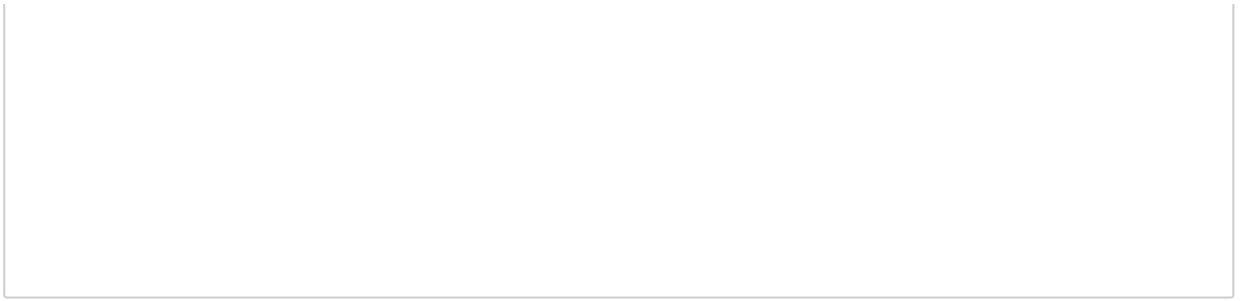
Future Challenge: Try using Python with map labels in ArcMap

Break

V. Introduction to ArcPy

What is ArcPy?

ArcPy is a site package that builds on (and is a successor to) the successful arcgisscripting module. Its goal is to create the cornerstone for a useful and productive way to perform geographic data analysis, data conversion, data management, and map automation with Python.

This package provides a rich and native Python experience offering code completion (type a keyword and a dot to get a pop-up list of properties and methods supported by that keyword; select one to insert it) and reference documentation for each function, module, and class.

The additional power of using ArcPy within Python is the fact that Python is a general-purpose programming language. It is interpreted and dynamically typed and is suited for interactive work and quick prototyping of one-off programs known as scripts while being powerful enough to write large applications in. ArcGIS applications written with ArcPy benefit from the development of additional modules in numerous niches of Python by GIS professionals and programmers from many different disciplines.

From http://desktop.arcgis.com/en/desktop/latest/analyze/arcpy/what-is-arcpy-.htm (http://desktop.arcgis.com/en/desktop/latest/analyze/arcpy/what-is-arcpy-.htm)

What is a Python Module?

- Extensions that can be imported and used in Python scripts to expand the buil-in capabilities
- Contain pre-written code to help out with specialized tasks and defines functions, classes and variables for those tasks
- Also allows you to logically organize your Python code
- Single file consisting of Python code and also include runnable code
- Modules are imported into a script using the `import` statement
- Many times you need to download and install new modules

What is a Python Site Package?

- Like a module, but contains a collections of modules, functions, and classes
- Site packages and modules both add functionality to Python

`import` Statements

```
import arcpy
    import os
    import sys

    # Best practice is to stack each import statement as above, but this is not wro
ng:
    import arcpy, os, sys

    import arcpy as a # This can sometimes make things simpler; you can use a.whate
ver instead of arcpy.whatever
    from arcpy import da # Do this if you only want a specific part of the site pac
kage/module
    from arcpy.sa import * # This also works, but should be used sparingly
```

Importing ArcPy

See _http://desktop.arcgis.com/en/arcmap/10.3/analyze/python/importing-arcpy.htm_
(http://desktop.arcgis.com/en/arcmap/10.3/analyze/python/importing-arcpy.htm) for more information

```
In [ ]:  # Always import the ArcPy site package and other modules first in your script
         # This may take a while if you are working outside of ArcGIS

         import arcpy
```

Utilizing the ArcPy ArcGIS Desktop Help Documentation

- ArcGIS Desktop Online Documentation: http://desktop.arcgis.com/en/documentation/ (http://desktop.arcgis.com/en/documentation/)
- ArcPy Help: http://desktop.arcgis.com/en/desktop/latest/analyze/arcpy/what-is-arcpy-.htm (http://desktop.arcgis.com/en/desktop/latest/analyze/arcpy/what-is-arcpy-.htm)

Using ArcPy in the ArcMap Python Window
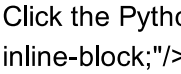
Another great place to practice Python in the ArcMap Python Window

1. Download exercise data from: [https://uofi.box.com/ILGISA-Python (https://uofi.box.com/ILGISA-Python)](https://uofi.box.com/ILGISA-Python)

- You should have already done this...but just is case you didn't

2. Open the 'Illinois.mxd' ArcMap Document

3. Open the Python Window in ArcMap

- Click the Python widow button >
- Allows for running geoprocessing tools while also taking advantage of other Python modules and librarires
- Can be used for single-line code (e.g. Used instead of ArcToolbox to access tools)
- Can be used for testing syntax and longer, more complex scripts that might be used for automating workflows
- Replaces the Command Line for earlier releases of ArcGIS

The Python Window

- Left side is the interactive Python interpreter panel
    - This is where code is entered
    - Note the three greater-than symbols (>>>)
- Right side is the help and syntax panel
- Code is generally executed one line at a time and displayed immediately. Exceptions include:
    - Multi-line constucts (such as `if` statements)
    - Pressing SHIFT or CTRL + ENTER at the end of a line of code (you may need to hit ENTER a few times when you are ready to run the code)
- Other Advanatages of the Python window
    - Autocompletion
    - Conditional and Iteration executuion
    - Scripts can be saved and reused or opened with another Python IDE

See [http://desktop.arcgis.com/en/arcmap/latest/analyze/executing-tools/what-is-the-python-window-.htm (http://desktop.arcgis.com/en/arcmap/latest/analyze/executing-tools/what-is-the-python-window-.htm)](http://desktop.arcgis.com/en/arcmap/latest/analyze/executing-tools/what-is-the-python-window-.htm) for more info

Describing Data with ArcPy

See ArcGIS Documentation: http://desktop.arcgis.com/en/desktop/latest/analyze/arcpy-functions/describe.htm (http://desktop.arcgis.com/en/desktop/latest/analyze/arcpy-functions/describe.htm)

System Paths vs. Catalog paths

- System paths are recognized by the Windows operating system
  - Files in folders
  - Ex. Shapefiles and rasters
- Catalog paths are only recognized by ArcGIS
  - Used for feature classes and other data in geodatabases
  - Geodatabases could be file (.gdb), personal (.mdb) or enterprise (.sde)
  - Contain 2 parts:
    - Workspace - could be the geodatabase root or a feture dataset
    - Base name - the feature class, raster, or other file types that can be saved in geodatabases
  - Requires the programmer to be aware of the context inwhich the path is being used

Type, do not copy, the following code into the Python window

Note: The following code is intended to only be run in the Python window of ArcMap

```
In [ ]:  # Step 1:
         import arcpy

         file_name = r'C:\Data\CSV\JeopardyContestants_LatLon.csv'
         print(arcpy.Exists(file_name))

         # What kind of path is this? System or Catalog?
         # What is the result?


         # Try again, this time, but type the up arrow to reload the last code
         # Replace the '...' with the location where you placed your downloaded data
         # I also have a cool trick...drag, drop, and roll!

         file_name = r'C:\...\Data\CSV\JeopardyContestants_LatLon.csv'
         print(arcpy.Exists(file_name))

         # Does it exist? Keep trying until you get the path correct and the result ret
         urns Treu


         print(arcpy.Describe(file_name).dataType)

         # What type of data is this?
```

```
In [ ]:  # Step 2:

         # Replace the '...' with the location where you placed your downloaded data

         feature_class = r'C:\...\Data\Illinois.gdb\Illinois_Counties'
         # What kind of path is this?

         print(arcpy.Exists(feature_class))

         print(arcpy.Describe(feature_class).dataType)
         # What type of data is this?
```

```
In [ ]:  # Step 3:

         layer_name = 'Illinois Counties'

         # What kind of path is this? System or Catalog?

         print arcpy.Exists(layer_name)

         lyr_desc = arcpy.Describe(layer_name)
         # Note how this Describe function and variable look different than before...
         # We can have describe objects become variables to be used later

         print lyr_desc.dataType
         print lyr_desc.dataElement.dataType
```

What is the difference between the last two print statements?

Lets look at the help:

- http://desktop.arcgis.com/en/desktop/latest/analyze/arcpy-functions/describe-object-properties.htm
  (http://desktop.arcgis.com/en/desktop/latest/analyze/arcpy-functions/describe-object-properties.htm)
- http://desktop.arcgis.com/en/desktop/latest/analyze/arcpy-functions/layer-properties.htm
  (http://desktop.arcgis.com/en/desktop/latest/analyze/arcpy-functions/layer-properties.htm)

```
In [ ]:   # Step 4:

          lyr_datatype = lyr_desc.dataElement.dataType

          lyr_path = lyr_desc.path

          lyr_basename = lyr_desc.baseName

          lyr_spatialref = lyr_desc.spatialReference.name

          lyr_count = arcpy.GetCount_management(layer_name)

          # Here is a new way to format strings with inline variable substitution

          print("{} is a {} stored at {} with a file name of {} with the {} coordinate s
          ystem and contains {} features."\
              .format(layer_name, lyr_datatype, lyr_path, lyr_basename, lyr_spatialref,
          lyr_count))

              # The slash does not need to be typed, it is there for formatting purposes
```

Formatting Strings

*See https://pyformat.info/ (https://pyformat.info/) and https://docs.python.org/2/library/string.html#custom-
string-formatting (https://docs.python.org/2/library/string.html#custom-string-formatting)

Listing Data with ArcPy

Environmental Settings

Type, do not copy, the following code into the Python window

Note: The following code is intended to only be run in the Python window of ArcMap

```
In [ ]:   # Step 1:

          # First, we must set the environmental settings
          # Replace the '...' with the location where you placed your downloaded data
          arcpy.env.workspace = r'C:\...\Data\CSV'

          print(arcpy.ListFiles())

          arcpy.env.workspace = r'C:\...\Data\Illinois.gdb'

          print(arcpy.ListFiles())

          print(arcpy.ListFeatureClasses())

          # What is the difference between the two list functions?
```

```
In [ ]:   # Step 2:

          print(arcpy.ListFields("Illinois Counties"))

          # What is the result? Can you understand it?
          # What is actually being output in this case? (Hint: What type of object?)

          fields = arcpy.ListFields("Illinois Counties")
          print(type(fields[0]))

          # We can print each field name...
          for field in fields:
              print(field.name)

          # Or the field aliases...
          for field in fields:
              print(field.aliasName)
```

```
In [ ]:    # Step 3:
           # We can also create a list of the fields; this is called a 'list comprehensio
           n' and they are very powerful

           field_list = [field.name for field in arcpy.ListFields("Illinois Counties")]

           print(field_list)

           # How does this result look differently than above?

           test_field = "OBJECTID"

           if test_field in field_list:
               print("{} exists".format(test_field))
           else:
               print("{} does not exist".format(test_field))

           # Repeat the above code using a test field that you know doesn't exist to test
            the 'else:' statement code
           # Does it work?
```

List Comprehensions

*See:

- http://desktop.arcgis.com/en/arcmap/latest/analyze/arcpy-functions/listfields.htm
  (http://desktop.arcgis.com/en/arcmap/latest/analyze/arcpy-functions/listfields.htm)
- http://www.learnpython.org/en/List_Comprehensions
  (http://www.learnpython.org/en/List_Comprehensions)
- http://www.pythonforbeginners.com/basics/list-comprehensions-in-python
  (http://www.pythonforbeginners.com/basics/list-comprehensions-in-python)

Geoprocessing Tools with ArcPy

Type, do not copy, the following code into the Python window

Clear the ArcMap Python Window

Note: The following code is intended to only be run in the Python window of ArcMap

```
In [ ]:  # Step 1:
         import arcpy

         layer_name = "Illinois Counties"

         expression = "ACSMEDHINC <= 40000"

         arcpy.SelectLayerByAttribute_management(layer_name, "NEW_SELECTION", expressio
         n)

         # Now clear the selection
         arcpy.SelectLayerByAttribute_management(layer_name, "CLEAR_SELECTION")
```

```
In [ ]:  # Step 2:

         # Ensure that a default geodatabase is set
         arcpy.env.workspace = r'C:\...\Data\Illinois.gdb'

         # Create a variable for dissolve fields
         disolve_felds = ["STATE_NAME", "ST_ABBREV"]

         # New drag, drop, and roll trick...

         arcpy.Dissolve_management(layer_name, "Illinois", disolve_felds, "", "SINGLE_P
         ART", "DISSOLVE_LINES")

         arcpy.PolygonToLine_management("Illinois", "Illinois_Boundary")

         # You may want to rearrage your layers in the TOC
```

```
In [ ]:  # Step 3:

         # Replace the '...' with the location where you placed your downloaded data

         csv_file = r"C:\...\Data\CSV\JeopardyContestants_LatLon.csv"

         arcpy.CopyRows_management(csv_file, "JeopardyContestants_Table")

         arcpy.MakeXYEventLayer_management("JeopardyContestants_Table", "lon", "lat",
         "Jeopardy Contestants")

         arcpy.Select_analysis("Jeopardy Contestants", "JeopardyContestants", '"lat" IS
          NOT NULL OR "lon" IS NOT NULL')

         arcpy.Buffer_analysis("JeopardyContestants", "JeopardyContestants_Buffer", "5
          Miles", "FULL", "ROUND", "ALL", "", "GEODESIC")
```

Your turn:

1: Write code to clip the buffer to the Illinois state layer

```
Be careful with which tool you choose...there are few tools named 'Clip_...'
```

2: Write code to intersect the new buffer to the Illinois Counties layer

```
You might need to check the tool's help documentation...
```

3: Save your work as a text file (.txt) in the 'Scripts' folder

```
Open the text file and view it in a text editor (like Notepad)
```

4: Save your work as a Python file (.py) in the 'Scripts' folder

```
After saving, open the .py file in PyScipter or other IDE
You may need to clean up the script for any mistakes or extraneous code
Clear the Python window in ArcMap
```

5. Load the Python file back into the Python window

```
Try running the whole script at once
This is one way you can create, save, and reuse a script for use again!
```

Complete!

VI. Conclusion

Python is simple to use and easy to learn!

The only way to get better is to practice and challenge yourself

Utilize Python and ArcGIS help documentation and forums

There is so much more to learn!!

```
In [ ]:   print("Have a great Lunch. Ta ta for now!")
```