PYTHON FOR ARCGIS

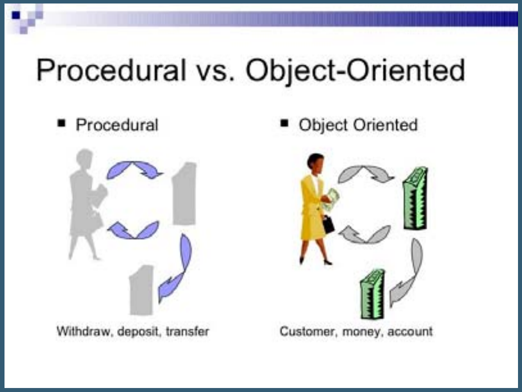
Su Zhang, Ph.D., GISP, CMS-RS

UNM CONTINUING EDUCATION

WHAT IS PYTHON

- It is a programming language that is both simple and powerful, and more importantly, it is simple and easy to learn
- It is free and open source
- It is cross platform
- Interpreted language, no special compilers required
- Object Oriented Programming language

gisgeography.com



alphansotech.com

SCRIPTING VS PROGRAMMING

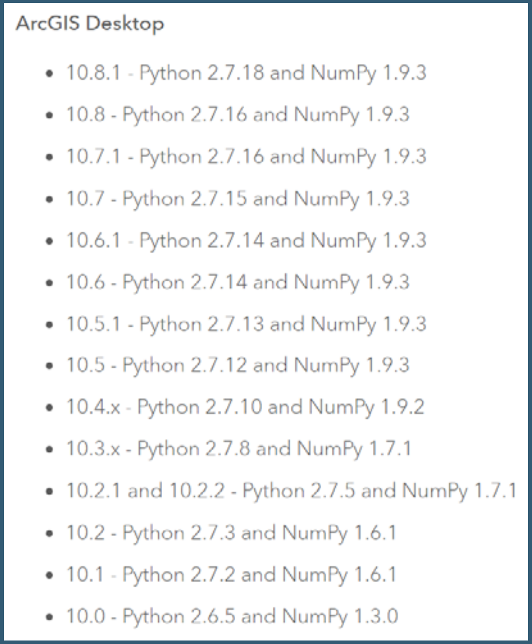
- Scripting – automating certain functionality within another program
- Programming – developing more sophisticated multifunctional applications
- Scripting is a programming task allows you to connect various existing components to accomplish a new, related task
- Scripting is the glue that allows you to put various existing elements together
- Programming allows you to build components from scratch, as well the applications that incorporate these components (system language)

geeksforgeeks.org

Python is both a scripting and a programming language!

SCRIPTING IN ARCGIS

- ArcGIS 9 introduced scripting support for many popular scripting languages, including, but not limited to, Python, VBScript, JavaScript, JScript, PERL, C#, Ruby, Scala, and Arcade
- Starting with ArcGIS 10, the Visual Basic for Applications (VBA) development environment is no longer installed by default, and ESRI is discouragingly the continued use of VBA
- Python is included with ArcGIS for Desktop installation
- ArcToolBox contains tools created from python scripts
- Check your ArcGIS Installation files for python version

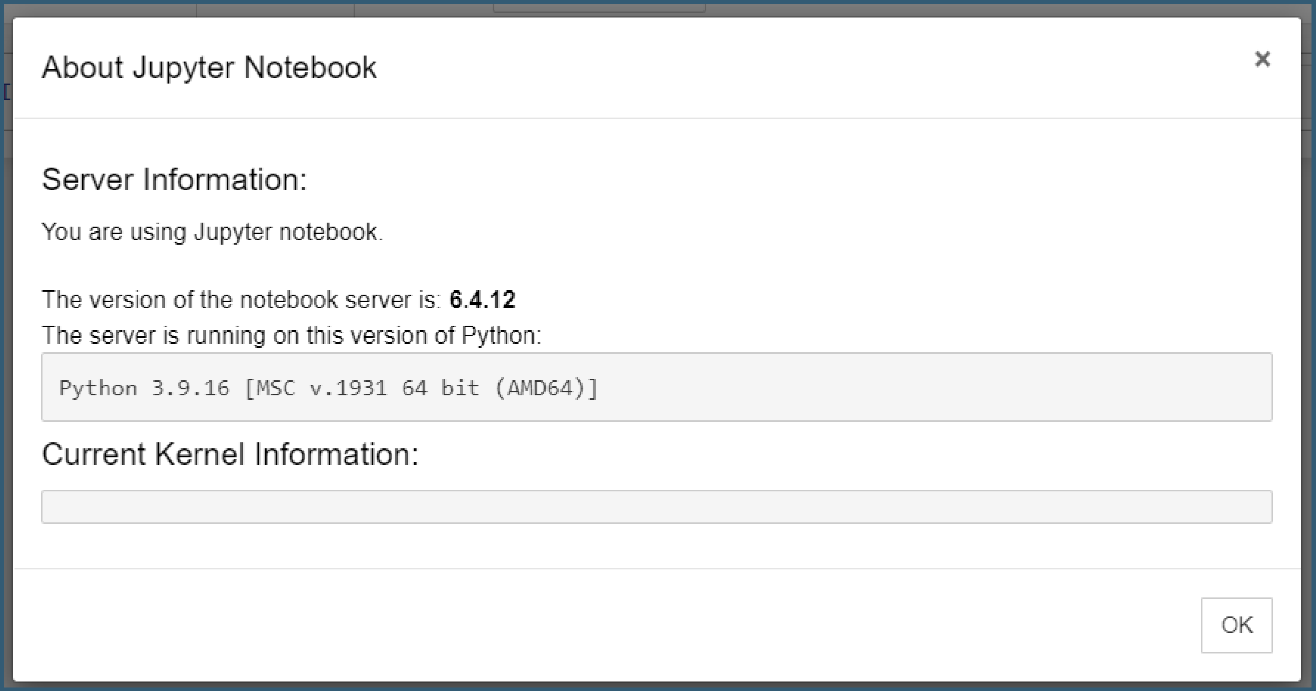


ArcGIS Desktop

- 10.8.1 - Python 2.7.18 and NumPy 1.9.3
- 10.8 - Python 2.7.16 and NumPy 1.9.3
- 10.7.1 - Python 2.7.16 and NumPy 1.9.3
- 10.7 - Python 2.7.15 and NumPy 1.9.3
- 10.6.1 - Python 2.7.14 and NumPy 1.9.3
- 10.6 - Python 2.7.14 and NumPy 1.9.3
- 10.5.1 - Python 2.7.13 and NumPy 1.9.3
- 10.5 - Python 2.7.12 and NumPy 1.9.3
- 10.4.x - Python 2.7.10 and NumPy 1.9.2
- 10.3.x - Python 2.7.8 and NumPy 1.7.1
- 10.2.1 and 10.2.2 - Python 2.7.5 and NumPy 1.7.1
- 10.2 - Python 2.7.3 and NumPy 1.6.1
- 10.1 - Python 2.7.2 and NumPy 1.6.1
- 10.0 - Python 2.6.5 and NumPy 1.3.0

esri.com

SCRIPTING IN ARCGIS PRO



Server Information:

You are using Jupyter notebook.

The version of the notebook server is: **6.4.12**
The server is running on this version of Python:

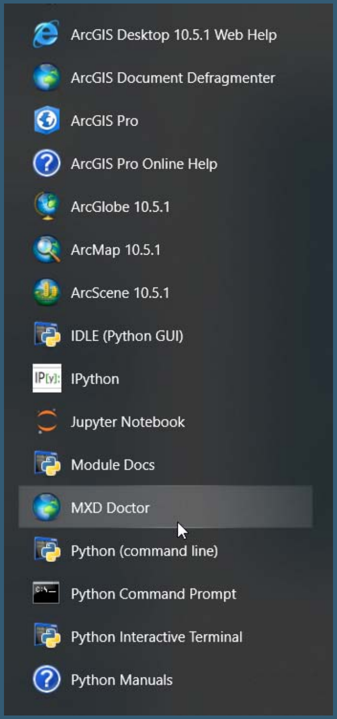
```
Python 3.9.16 [MSC v.1931 64 bit (AMD64)]
```

Current Kernel Information:

HOW TO WRITE PYTHON SCRIPTS?

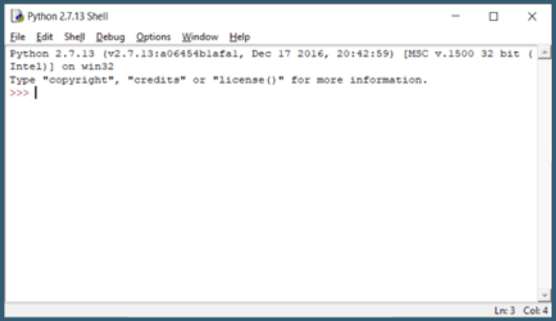
- Python Command line
  - All Programs > ArcGIS > Python x.x > Python (commnad line) or Python Command Prompt or IDLE (Python GUI) or Python Interactive Terminal or IPython
- Python Script Editor
  - Integrated development environments (IDEs)
  - Syntax formating and highlighting
  - Jupyter Notebook
- Python window in ArcMap or ArcGIS Pro
- Other code editors – Notepad++, PythonWin, Visual Studio Code, ...

https://wiki.python.org/moin/PythonEditors



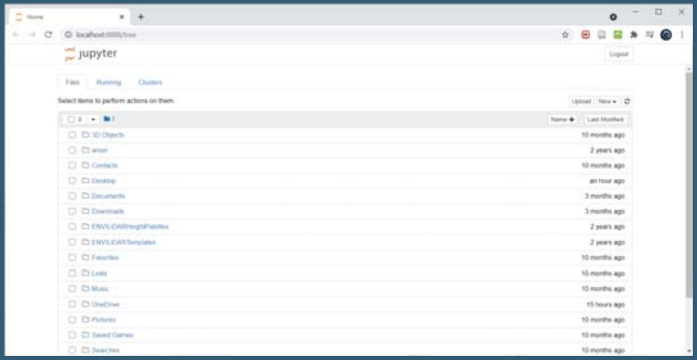
SCRIPTING IN ARCGIS

IDLE (Python GUI)



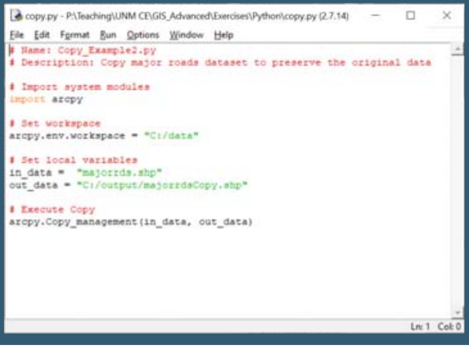
```
Python 2.7.13 (v2.7.13:a06454b1a1a1, Dec 17 2016, 20:42:59) [MSC v.1500 32 bit (
Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> |
```

Jupyter Notebook



```
3D Objects                10 months ago
arcgis                    2 years ago
Contacts                  10 months ago
Desktop                   an hour ago
Documents                 3 months ago
Downloads                 3 months ago
ENVILiDARHeightPalettes   2 years ago
ENVILiDARTemplates        2 years ago
Favorites                 10 months ago
Links                     10 months ago
Music                     10 months ago
OneDrive                  15 hours ago
Pictures                  10 months ago
Saved Games               10 months ago
Searches                  10 months ago
```

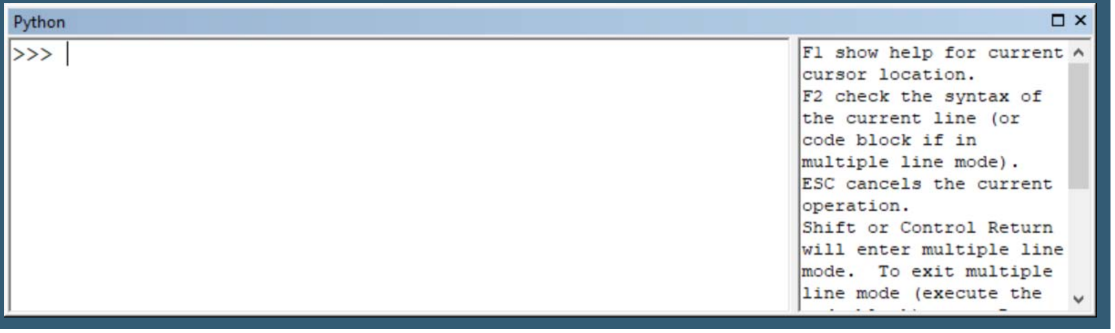
Python File



copy.py - P:\Teaching\UNM CE\GIS_Advanced\Exercises\Python\copy.py (2.7.14)

```python
# Name: Copy_Example2.py
# Description: Copy major roads dataset to preserve the original data

# Import system modules
import arcpy

# Set workspace
arcpy.env.workspace = "C:/data"

# Set local variables
in_data =  "majorrds.shp"
out_data = "C:/output/majorrdsCopy.shp"

# Execute Copy
arcpy.Copy_management(in_data, out_data)
```

Python window in ArcMap

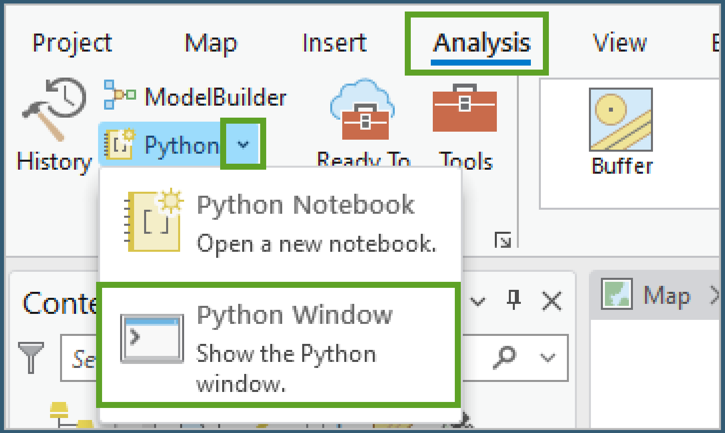


```
>>> |
```

```
F1 show help for current
cursor location.
F2 check the syntax of
the current line (or
code block if in
multiple line mode).
ESC cancels the current
operation.
Shift or Control Return
will enter multiple line
mode.  To exit multiple
line mode (execute the
```

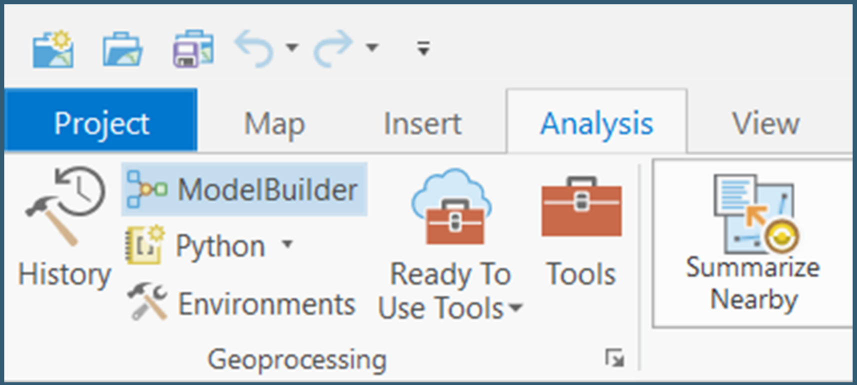
PYTHON AND GEOPROCESSING

- Allows to perform spatial analysis, modeling, and automating GIS tasks
- Supports the automation of workflows by creating a sequence that combines series of tools
- Geoprocessing framework comprises of
  - Tools organized in toolboxes and toolsets
  - Methods to find and execute tools (standard tools, model builder, & python)
  - Parameters and environment settings
  - Results window that logs the tool execution



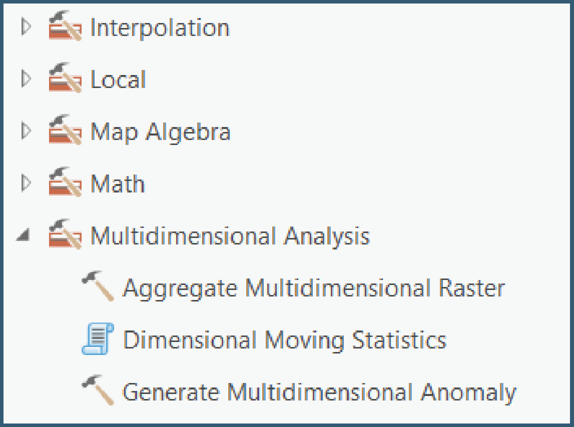
TYPES AND CATEGORIES OF TOOLS

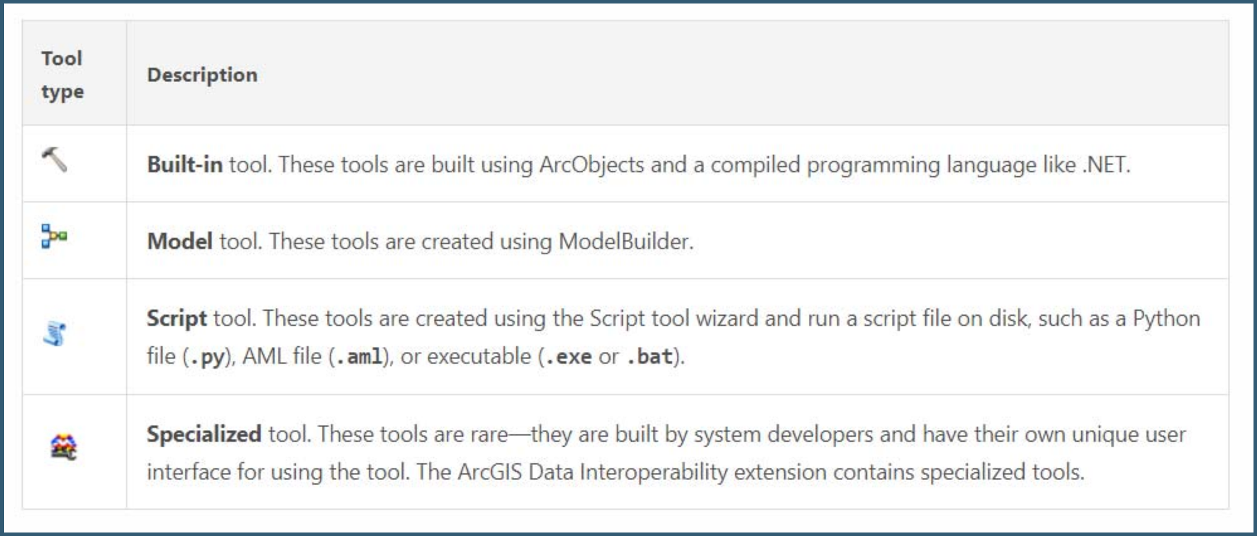
- Types
  - Built-in tools
  - Model tools
  - Script tools
  - Specialized tools
- Categories
  - System tools - installed as part of ArcGIS software
  - Custom tools - script and model tools or other third party Add-ins



desktop.arcgis.com

TOOL TYPES

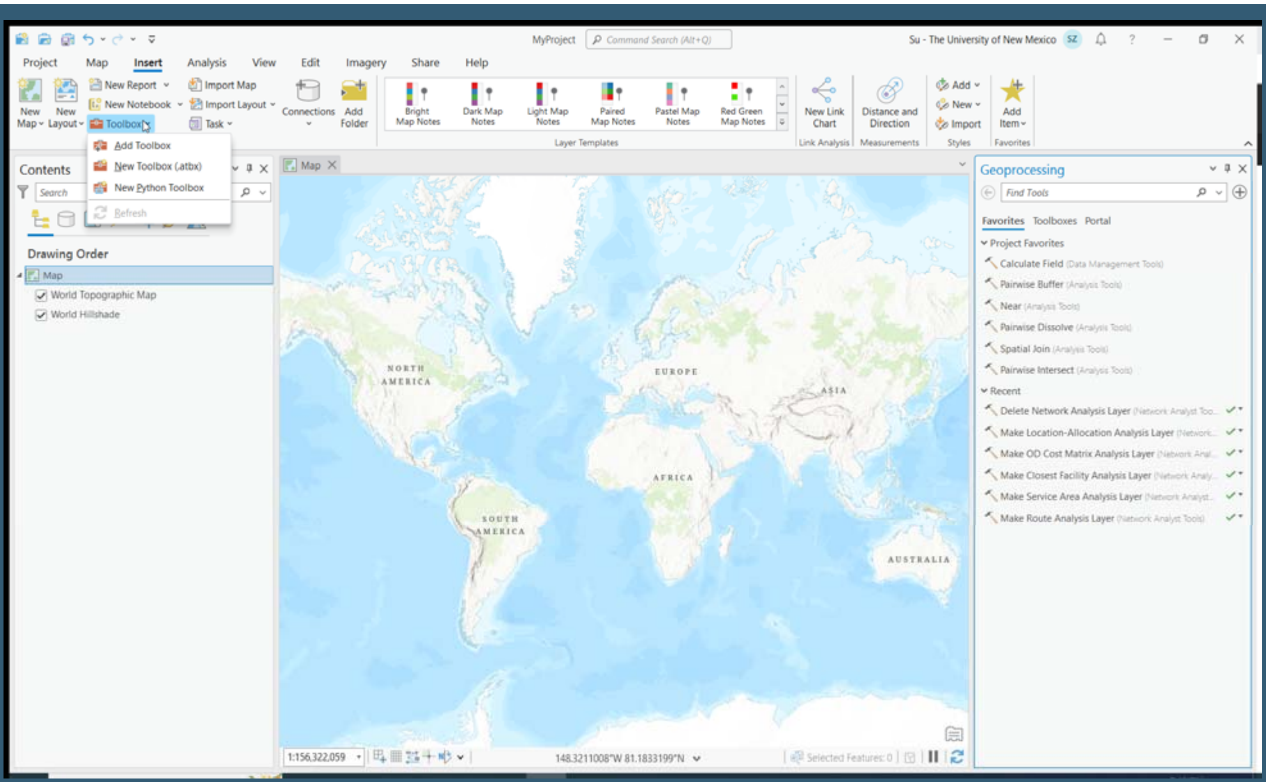

Interpolation
Local
Map Algebra
Math
Multidimensional Analysis
Aggregate Multidimensional Raster
Dimensional Moving Statistics
Generate Multidimensional Anomaly



Tool type	Description
🔨	**Built-in** tool. These tools are built using ArcObjects and a compiled programming language like .NET.
	Model tool. These tools are created using ModelBuilder.
	Script tool. These tools are created using the Script tool wizard and run a script file on disk, such as a Python file (`.py`), AML file (`.aml`), or executable (`.exe` or `.bat`).
	Specialized tool. These tools are rare—they are built by system developers and have their own unique user interface for using the tool. The ArcGIS Data Interoperability extension contains specialized tools.

desktop.arcgis.com

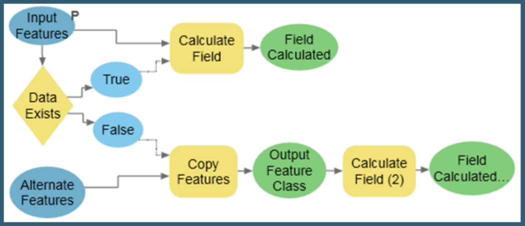
TOOL CATEGORIES



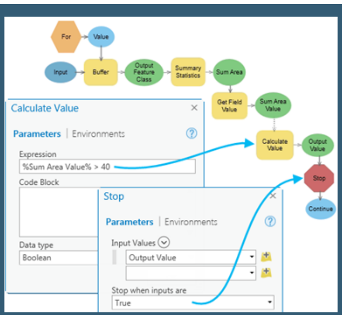
MODELBUILDER VS PYTHON

- ModelBuilder is a visual programming language
  - Intuitive way to create tools/workflows
  - String together sequences of geoprocessing tools
  - Requires no programming experience or to learn syntax
  - Could be slow
- Python is a text-based programming language
  - Allows advanced programming logic
  - Can be used with other software packages
  - Can be run as a standalone script outside of ArcGIS
  - Can be scheduled to run at a specific time
  - Much faster

Build your model first! And then build your code!



pro.arcgis.com



Expression: %Sum Area Value% > 40

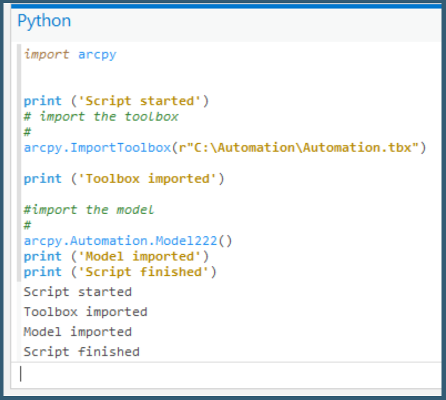
Code Block

Data type: Boolean

Input Values: Output Value

Stop when inputs are: True

pro.arcgis.com

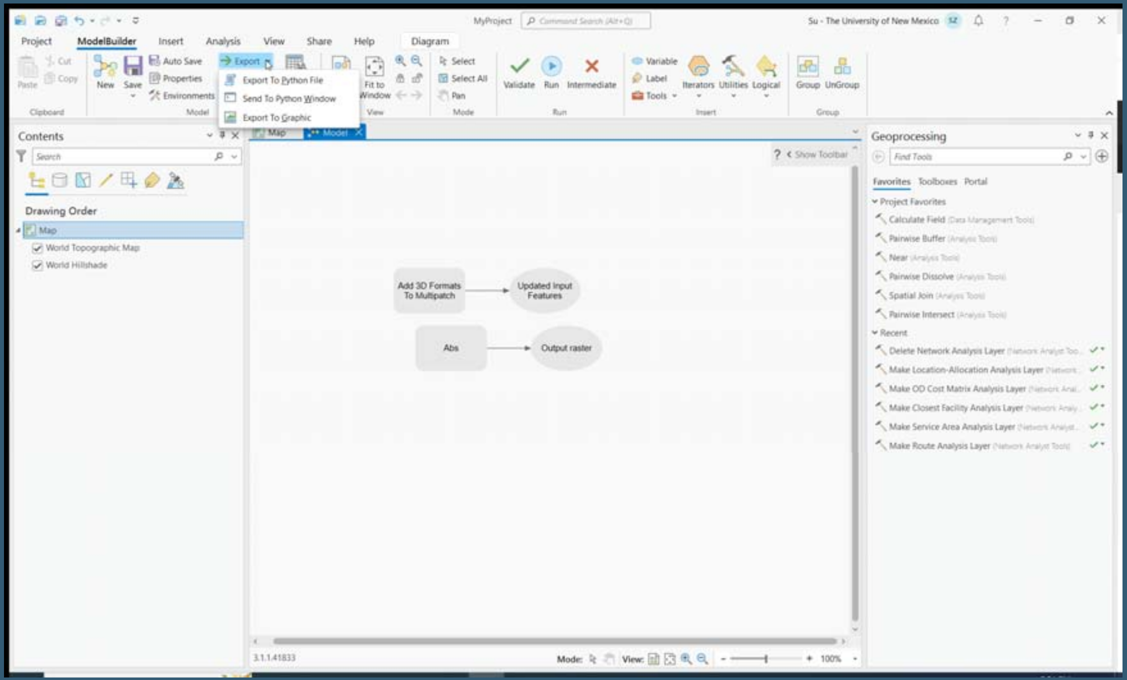


Python

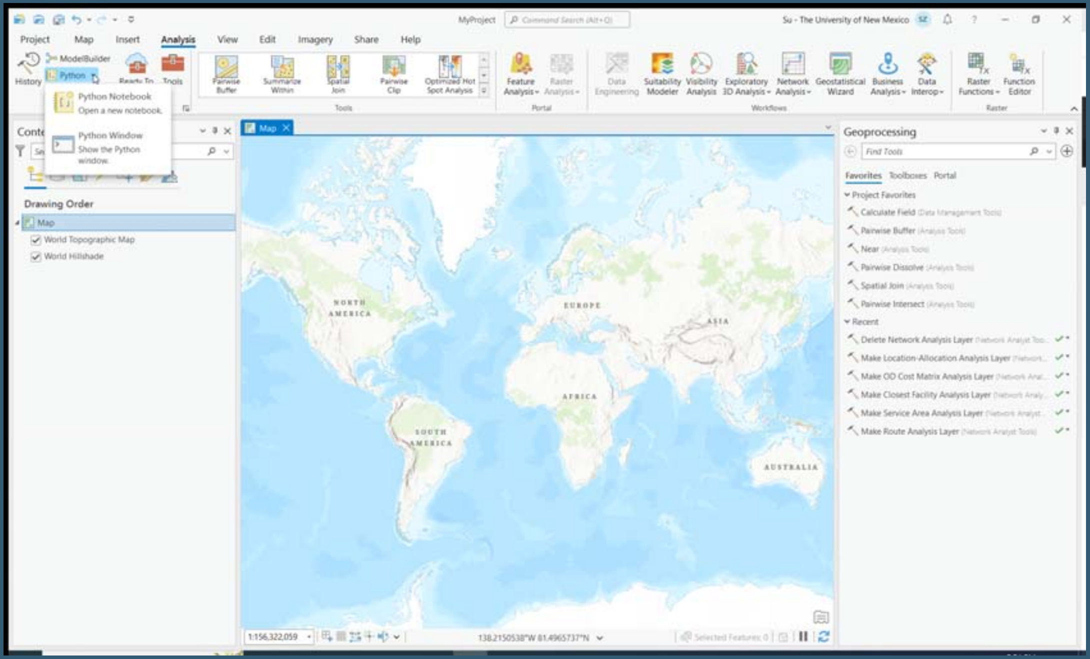
```
import arcpy


print ('Script started')
# import the toolbox
#
arcpy.ImportToolbox(r"C:\Automation\Automation.tbx")

print ('Toolbox imported')

#import the model
#
arcpy.Automation.Model222()
print ('Model imported')
print ('Script finished')
Script started
Toolbox imported
Model imported
Script finished
```

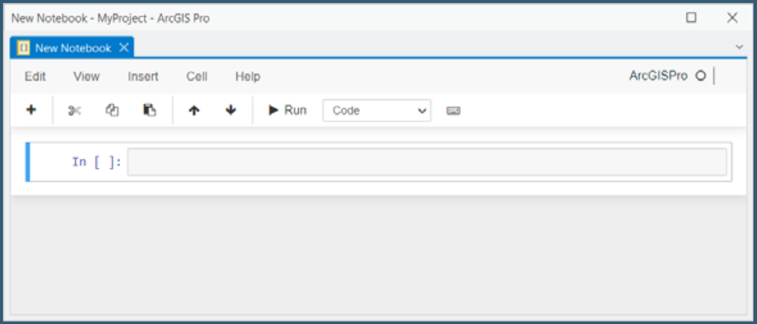
MODELBUILDER VS PYTHON

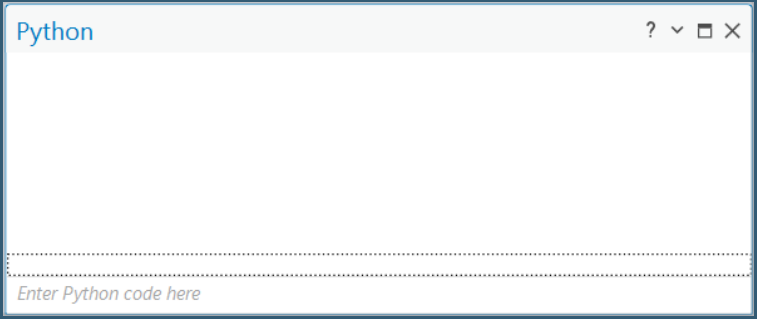
- Models can be converted to scripts, but not vice versa
- ModelBuilder has limitations and cannot perform more complex geoprocessing operations



PYTHON WINDOW IN ARCGIS DESKTOP

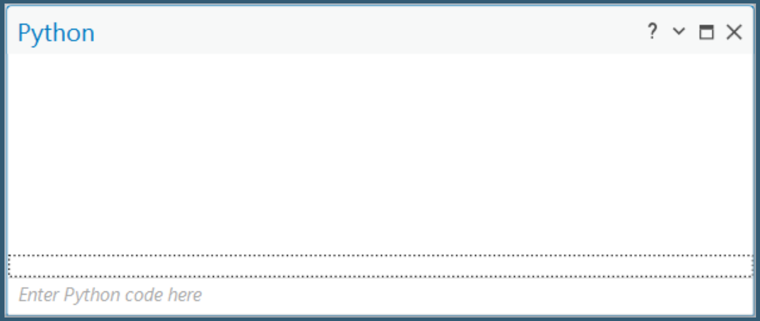






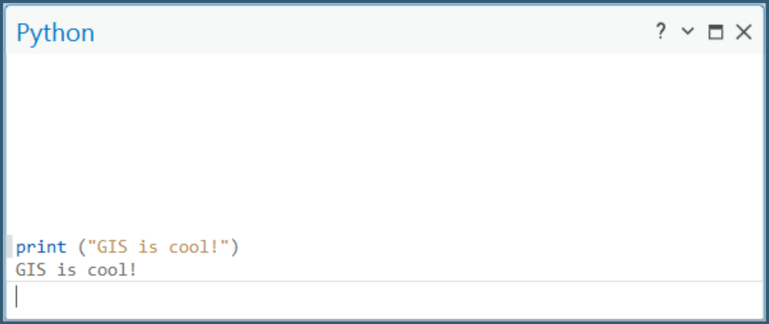
PYTHON WINDOW IN ARCGIS PRO

- The bottom session of the Python window is called the prompt, where you can type your code
- The top session of the Python window is called the transcript, which is intially blank. The transcript provides a record of previously entered code and its results.
- Right click on the codes and select Clear Transcript to delete codes.

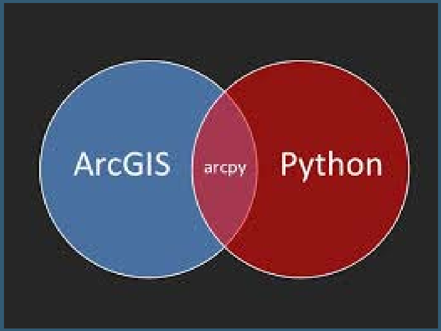


PYTHON WINDOW IN ARCGIS DESKTOP

- Single line code can be executed with ENTER command at the end of each line
- Multiple line code can be executed with ENTER command two times
- Result is printed to top session and bottom session starts with a new prompt
- Multiline code uses secondary prompt to complete the code. Secondary prompt is automatically added when pressing the ENTER key at the end of a line of code
- All geoprocessing tools can be accessed by importing ArcPy site package



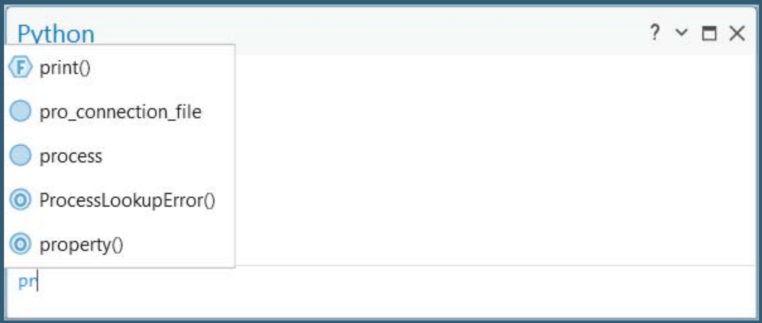
```
print ("GIS is cool!")
GIS is cool!
```



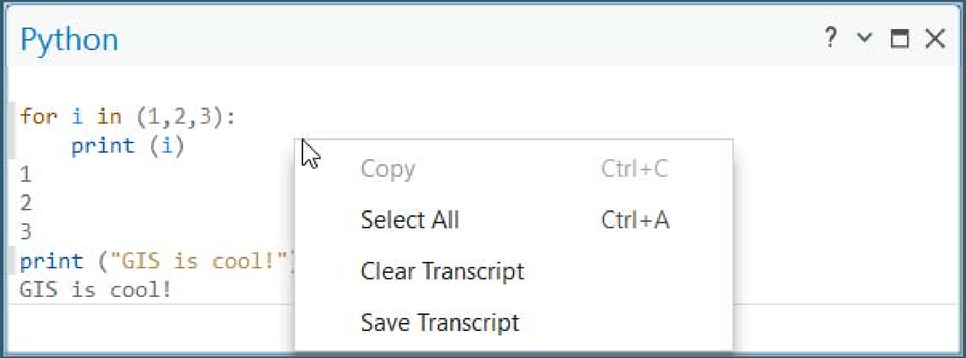
desktop.arcgis.com

PYTHON WINDOW IN ARCGIS DESKTOP

- Supports Autocomplete functionality
- Conditional execution can be performed using if-then-else logic
- Iteration can be done with for and while loops
- Python provides access to third party modules for data manipulation
- Python code blocks written in the window can be saved to a python or text file
- Allows to load code from another file



```
pr
```



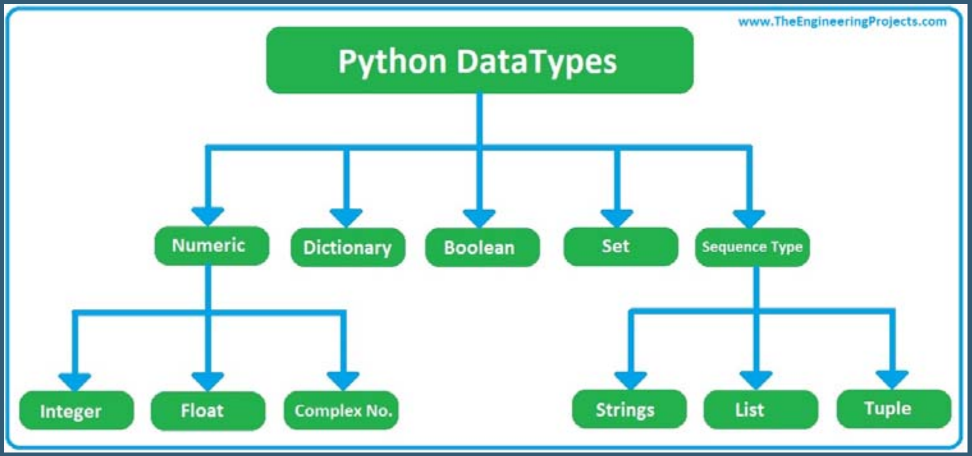
```
for i in (1,2,3):
    print (i)
1
2
3
print ("GIS is cool!"
GIS is cool!
```

PYTHON BASICS

Basics of Python

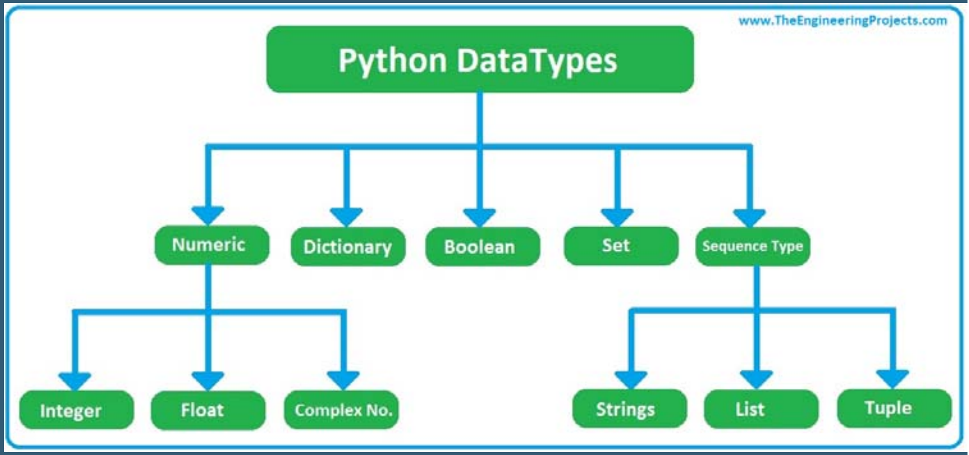
DATA TYPES

- Strings: texts such as “GIS”
- Lists: stores a sequence of items inside brackets separated by commas [1, 2, 4, 8, 16, 32] or [“Ford”, “Chevy”, “Toyota”, “Honda”, “Subaru”]
- Tuples: Similar to lists, uses parentheses, but are immutable, meaning they cannot be changed 1, 2, 4, 6, 8, 16, 32



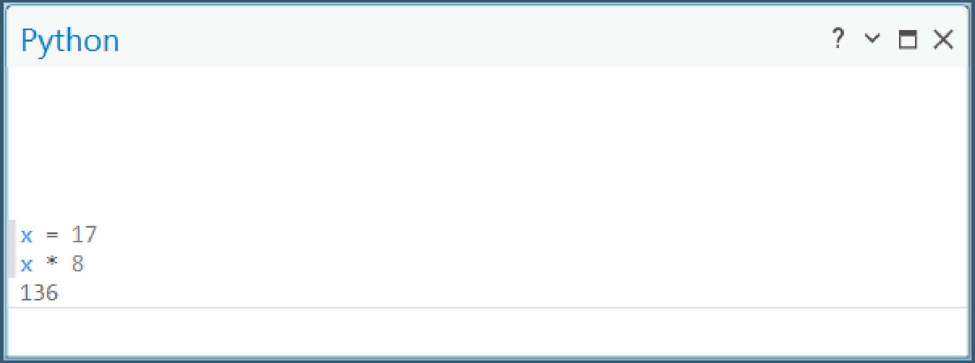
DATA TYPES

- Boolean: True or False
- Dictionaries: stores pairs of items (key and values) {“Austin”: “Texas”, “Baltimore”: “Maryland”, “Cleveland”: “Ohio”, “Denver”: “Colorado”}
- Numeric: 5, 1.3435926, 3+1j
- Set is an unordered collection of data type that is iterable, mutable and has no duplicate elements; The order of elements in a set is undefined though it may consist of various elements [1, 2, 'Geeks', 4, 'For', 6]



DATA TYPES

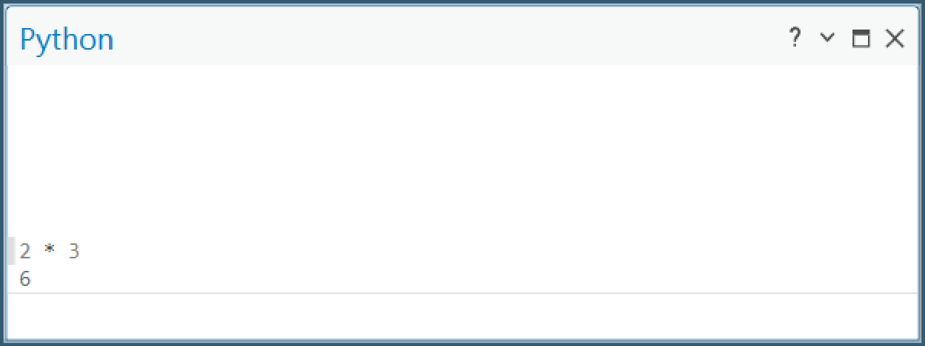
- Python scripts uses variables to store information
- Python will know your variable's type by the value you assigned to
  - If 17 then it is integer
  - If "GIS" then it is string
  - If 3.1415926 then it is float
- No need to declare a variable and define its type before you can actually use it
  - Int age = 20 (variable type, variable name, and then variable value)



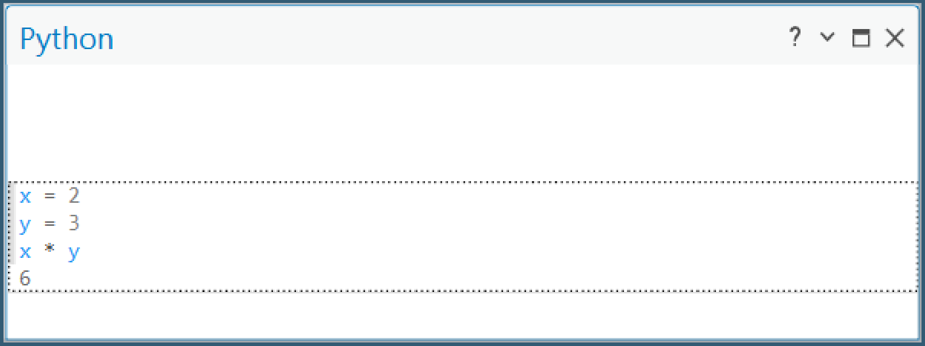
```
x = 17
x * 8
136
```

EXPRESSIONS, STATEMENTS, FUNCTIONS, METHODS

- An expression is a value
  - Ex: 2 * 3 is an expression returning the value of 6
- Expressions can be built using operators and functions
- Expressions can contain variables



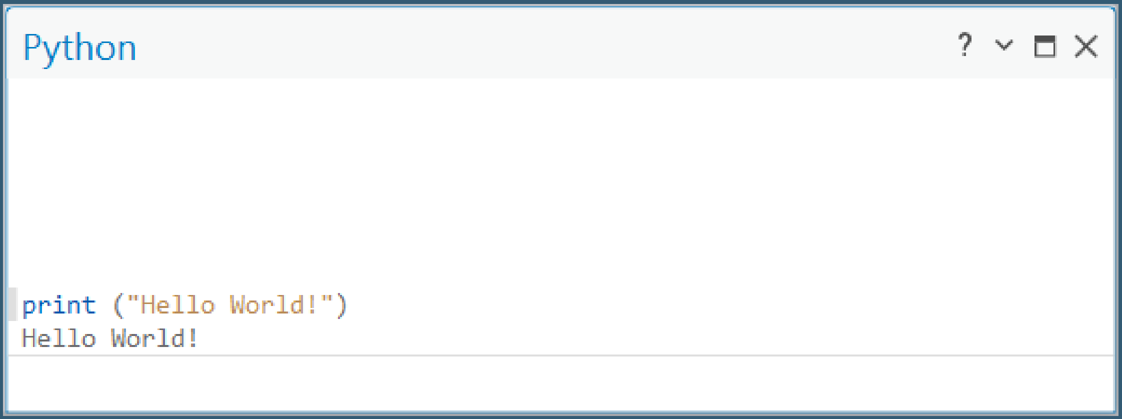
```
2 * 3
6
```



```
x = 2
y = 3
x * y
6
```

EXPRESSIONS, STATEMENTS, FUNCTIONS, METHODS

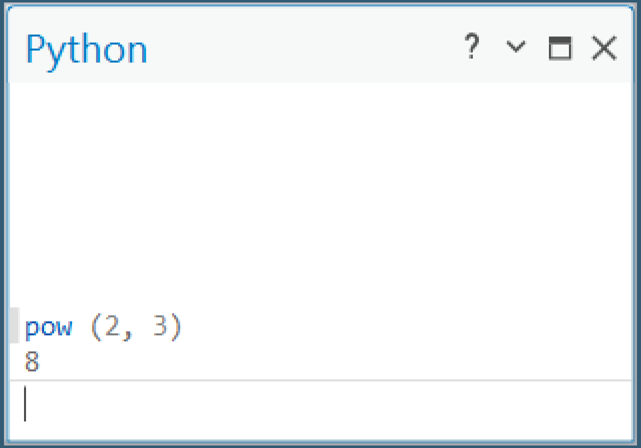
- A statement is an instruction to perform something
  - Ex: x = 2 * 3
  - Ex: Print, for, if-then-else, ...
- A statement do not return a value (exception print)



```
print ("Hello World!")
Hello World!
```

EXPRESSIONS, STATEMENTS, FUNCTIONS, METHODS

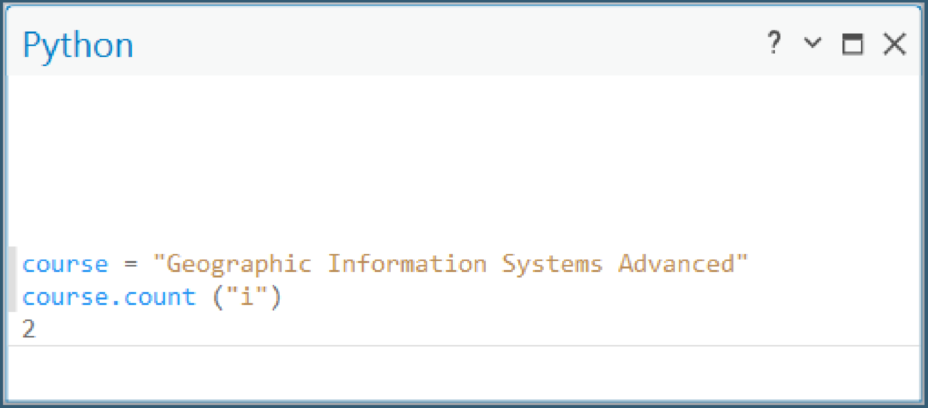
- A function is similar to a statement (to do something) and returns a value
- A function is a small program used to carry out a certain action.
  - Ex: pow (2, 3)
- Python includes some built-in functions with installation
- Many functions are available in python than the built-in functions. Using them requires modules
  - Ex: import os, import math



```
pow (2, 3)
8
```

EXPRESSIONS, STATEMENTS, FUNCTIONS, METHODS

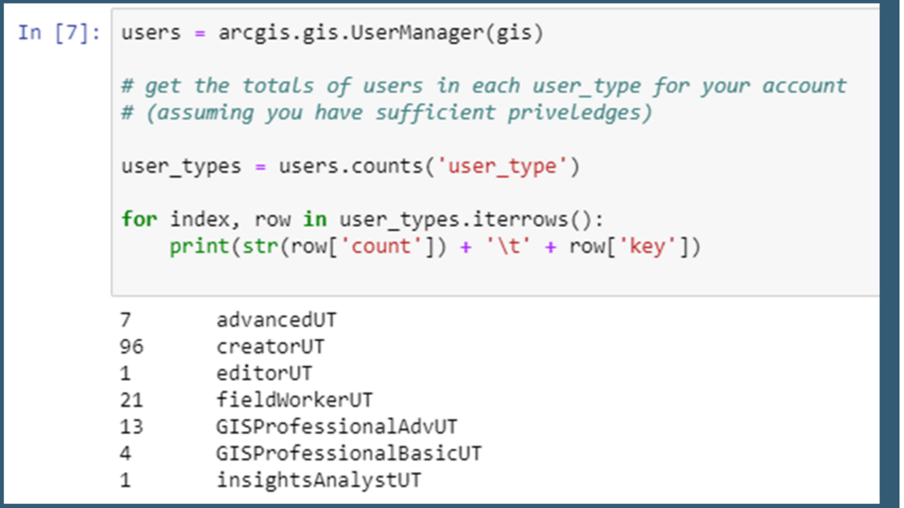
- Methods are similar to functions
- A method is a function that is closely coupled to an object
- Methods are case sensitive
  - Ex: <object>.<method>(<arguments>)
  - >>> course = “Geographic Information Systems Advanced”
  - >>> course.count(“i”)
  - 2



```
course = "Geographic Information Systems Advanced"
course.count ("i")
2
```

CODING TIPS

- Python scripting is case sensitive
- Variable names should be all lowercase and contain only characters, digits, and the underscore (_)
- Indentation:
  - Use of four spaces is recommended to define each indentation level
  - never mix tabs and spaces
- Comments:
  - Scripts should contain adequate commenting
  - Each script tool or function should have a header that contains script name, a description of how the script works, its requirements, who wrote it and when

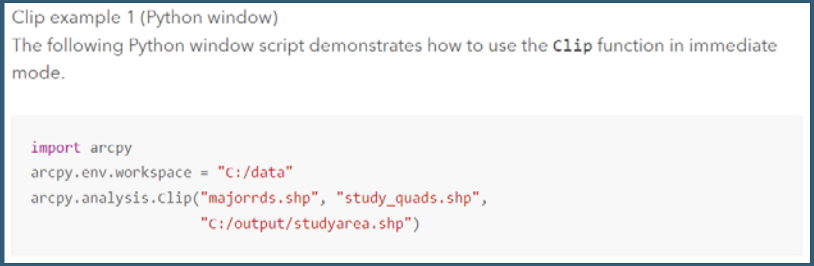


```
In [7]: users = arcgis.gis.UserManager(gis)

        # get the totals of users in each user_type for your account
        # (assuming you have sufficient priveledges)

        user_types = users.counts('user_type')

        for index, row in user_types.iterrows():
            print(str(row['count']) + '\t' + row['key'])

7       advancedUT
96      creatorUT
1       editorUT
21      fieldWorkerUT
13      GISProfessionalAdvUT
4       GISProfessionalBasicUT
1       insightsAnalystUT
```

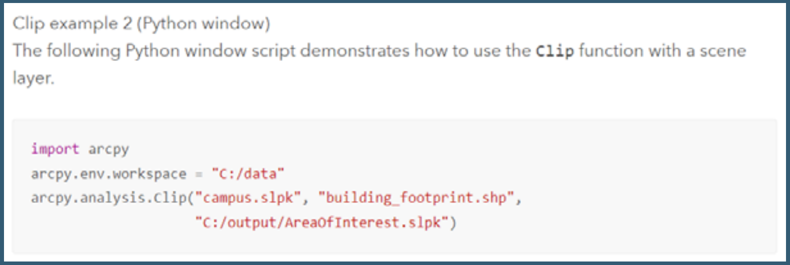
esriuk.com

PYTHON SYNTAX IN ARCGIS PRO

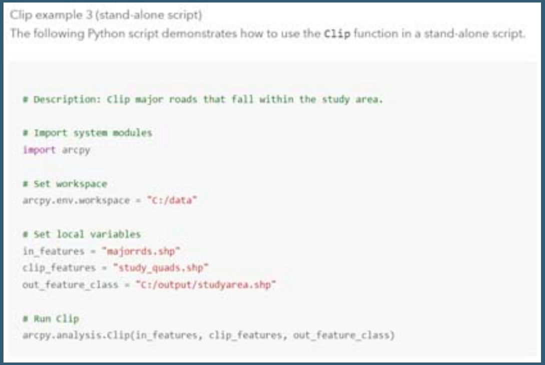
- Clip
  - Syntax: arcpy.analysis.Clip(in_features, clip_features, out_feature_class, {cluster_tolerance})


Clip example 1 (Python window)
The following Python window script demonstrates how to use the **Clip** function in immediate mode.

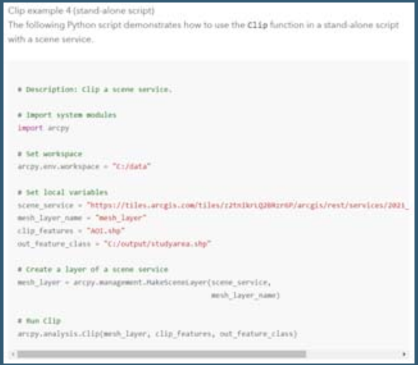
```
import arcpy
arcpy.env.workspace = "C:/data"
arcpy.analysis.Clip("majorrds.shp", "study_quads.shp",
                    "C:/output/studyarea.shp")
```


Clip example 2 (Python window)
The following Python window script demonstrates how to use the **Clip** function with a scene layer.

```
import arcpy
arcpy.env.workspace = "C:/data"
arcpy.analysis.Clip("campus.slpk", "building_footprint.shp",
                    "C:/output/AreaOfInterest.slpk")
```


Clip example 3 (stand-alone script)
The following Python script demonstrates how to use the **Clip** function in a stand-alone script.

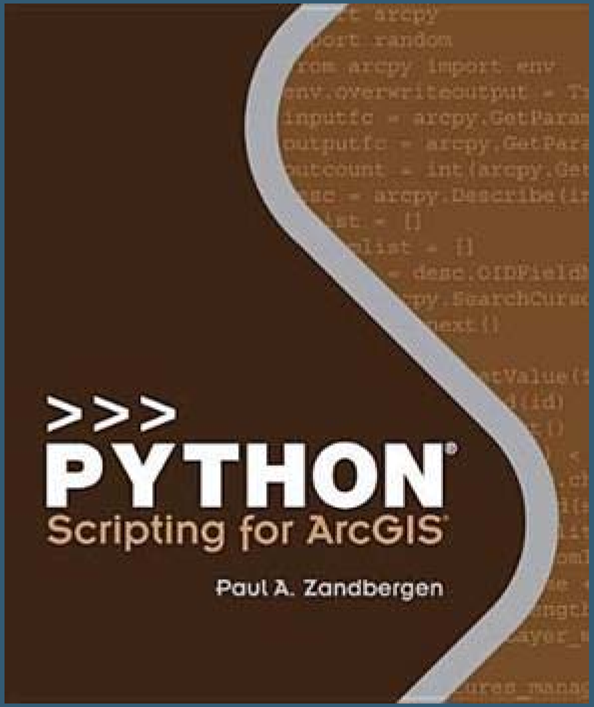
```
# Description: Clip major roads that fall within the study area.

# Import system modules
import arcpy

# Set workspace
arcpy.env.workspace = "C:/data"

# Set local variables
in_features = "majorrds.shp"
clip_features = "study_quads.shp"
out_feature_class = "C:/output/studyarea.shp"

# Run Clip
arcpy.analysis.Clip(in_features, clip_features, out_feature_class)
```


Clip example 4 (stand-alone script)
The following Python script demonstrates how to use the **Clip** function in a stand-alone script with a scene service.

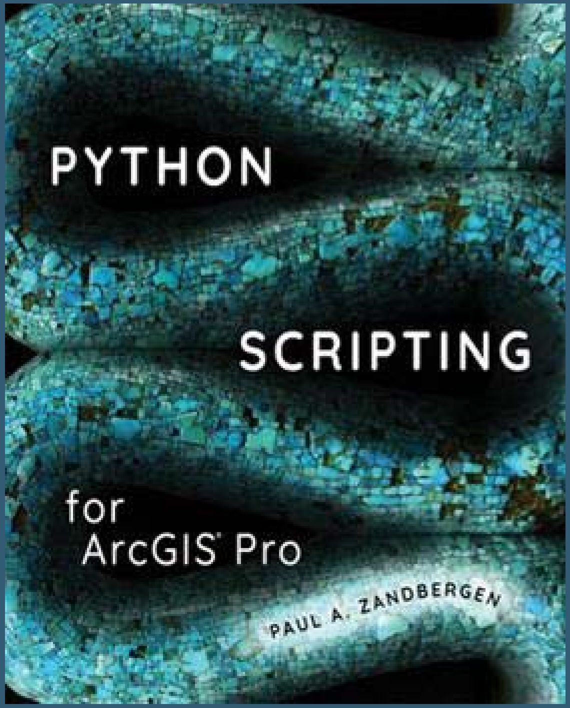
```
# Description: Clip a scene service.

# Import system modules
import arcpy

# Set workspace
arcpy.env.workspace = "C:/data"

# Set local variables
scene_service = "https://tiles.arcgis.com/tiles/z2tnIkrLQ2BRzr6P/arcgis/rest/services/2021_
mesh_layer_name = "mesh_layer"
clip_features = "AOI.shp"
out_feature_class = "C:/output/studyarea.shp"

# Create a layer of a scene service
mesh_layer = arcpy.management.MakeSceneLayer(scene_service,
                                             mesh_layer_name)

# Run Clip
arcpy.analysis.Clip(mesh_layer, clip_features, out_feature_class)
```

https://pro.arcgis.com/en/pro-app/latest/tool-reference/analysis/clip.htm

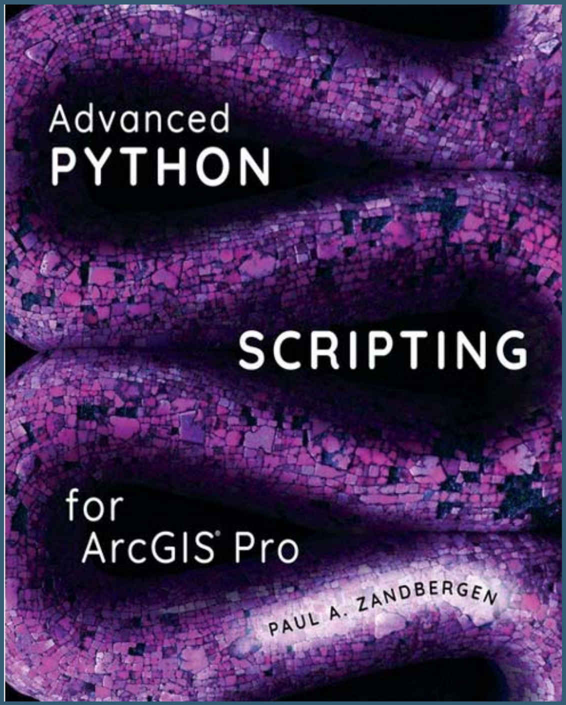
PYTHON BOOKS



PYTHON BOOKS



PYTHON BOOKS


Advanced
PYTHON
SCRIPTING
for
ArcGIS® Pro
PAUL A. ZANDBERGEN