

Python and ArcGIS (Under Construction)



This guide was created by the staff of the GIS/Data Center at Rice University and is to be used for individual educational purposes only.

The steps outlined in this guide require access to ArcGIS Pro software and data that is available both online and at Fondren Library.

- Dataset for this course.
- Python Crash Course
 - What is Python?
 - Setting Up a Python Script in ArcGIS Pro
 - Python Data Types
 - Numbers
 - Booleans
 - Lists
 - Strings
 - Dictionaries
 - Functions
 - Conditional Statements
 - Loops
- Using Python in ArcGIS
- Field Calculations
 - Fields in ArcGIS Pro
 - Why Use Python?
 - Switching Data Types
 - Defining a Field from Other Fields
- Notebooks
 - Jupyter Notebooks
 - Jupyter Notebooks in ArcGIS Pro
 - Automating Workflow Using Jupyter Notebooks



The following text styles are used throughout the guide:

Explanatory text appears in a regular font.

1. Instruction text is numbered.
2. Required actions are underlined.
3. **Objects of the actions are in bold.**

Folder and file names are in italics.

Names of Programs, Windows, Panes, Views, or Buttons are Capitalized.

'Names of windows or entry fields are in single quotation marks.'

"Text to be typed appears in double quotation marks."



The following step-by-step instructions and screenshots are based on the Windows 10 operating system with the Windows Classic desktop theme and ArcGIS Pro 3.0.0 software. If your personal system configuration varies, you may experience minor differences from the instructions and screenshots. Some of the features covered in this tutorial may not be available if using an older version of ArcGIS Pro.

Dataset for this course.

Python Crash Course

What is Python?

Python is a high-level scripting programming language often used for data analysis, among numerous other applications. In ArcGIS Pro, Python is used to define many of the tools in the background, and can also be utilized by the user to automate part or all of a data analysis project in ArcGIS Pro. This automation is done by writing Python scripts, which are instructions written by the programmer for Python to carry out.

Setting Up a Python Script in ArcGIS Pro

The easiest (and most convenient) way to write Python scripts in ArcGIS Pro is to use notebooks in ArcGIS Pro. We will first set up a notebook in ArcGIS Pro to go over some Python basics, and learn more about notebooks later on.

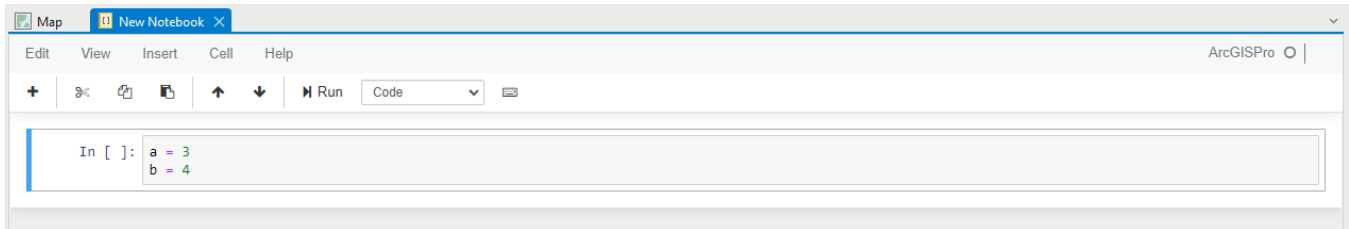
(setup notebook)

Python Data Types

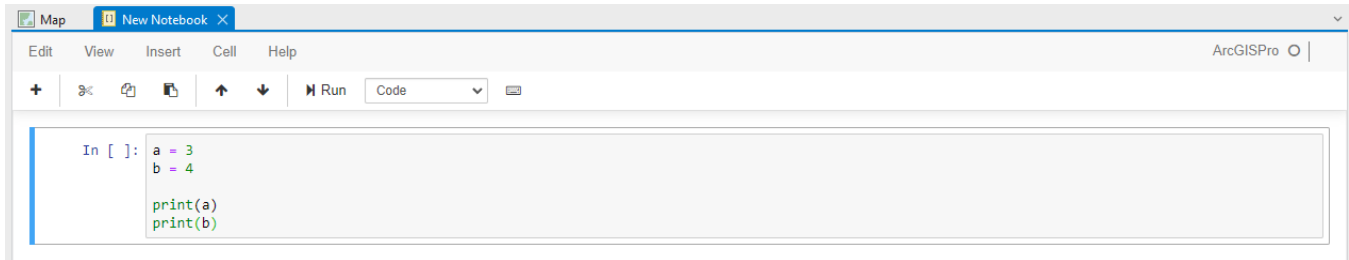
The building blocks of a program are variables, which store data in a variety of forms. This data can be made available to the programmer and used later in the program in multiple ways.

Numbers

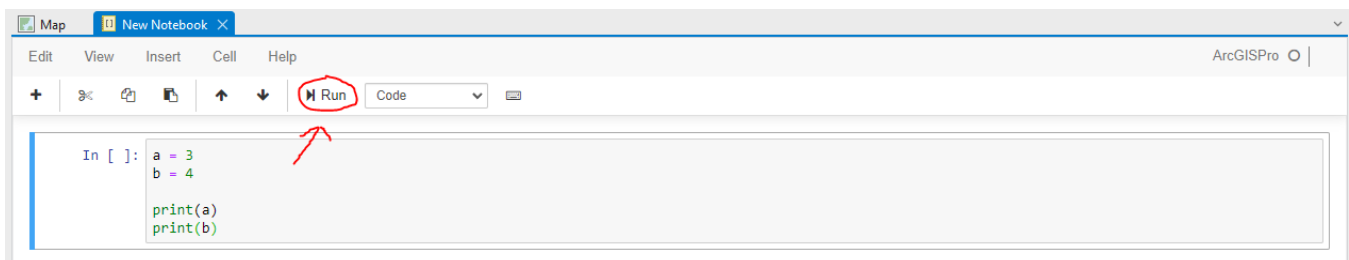
First, let's work with numbers in Python. By typing the code shown below, we can store the number 3 in variable *a*, and the number 4 in variable *b*.



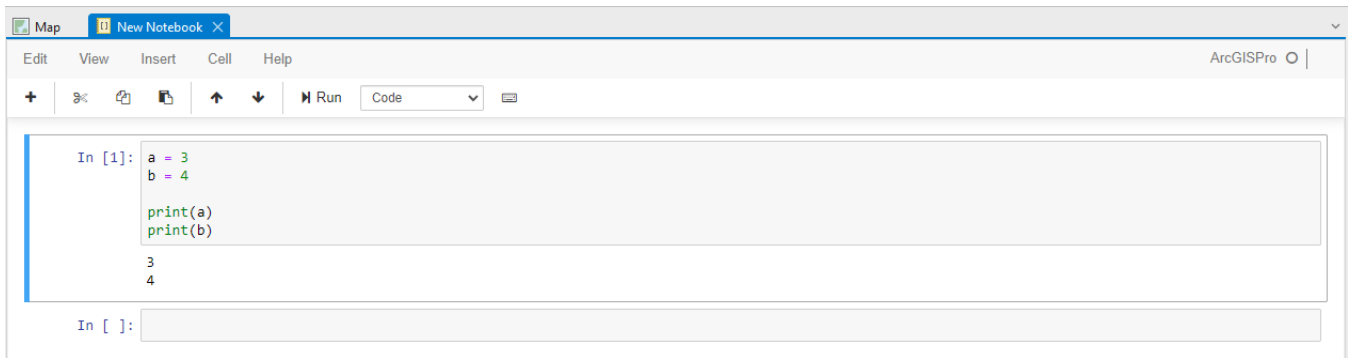
We can check the values of these variables by using the print function as shown below.



Click the "Run" button to run the code in the current cell as indicated below.



After running this cell, we can see the values 3 and 4 appear just below the cell that has been run.



The screenshot shows the ArcGIS Pro Notebook interface. The top menu bar includes 'Map', 'New Notebook', 'Edit', 'View', 'Insert', 'Cell', and 'Help'. The right side of the menu bar shows 'ArcGISPro' and a search icon. Below the menu bar is a toolbar with icons for adding a new cell, undo, redo, and running the code. The main area contains a code cell labeled 'In [1]:'. The code in the cell is:

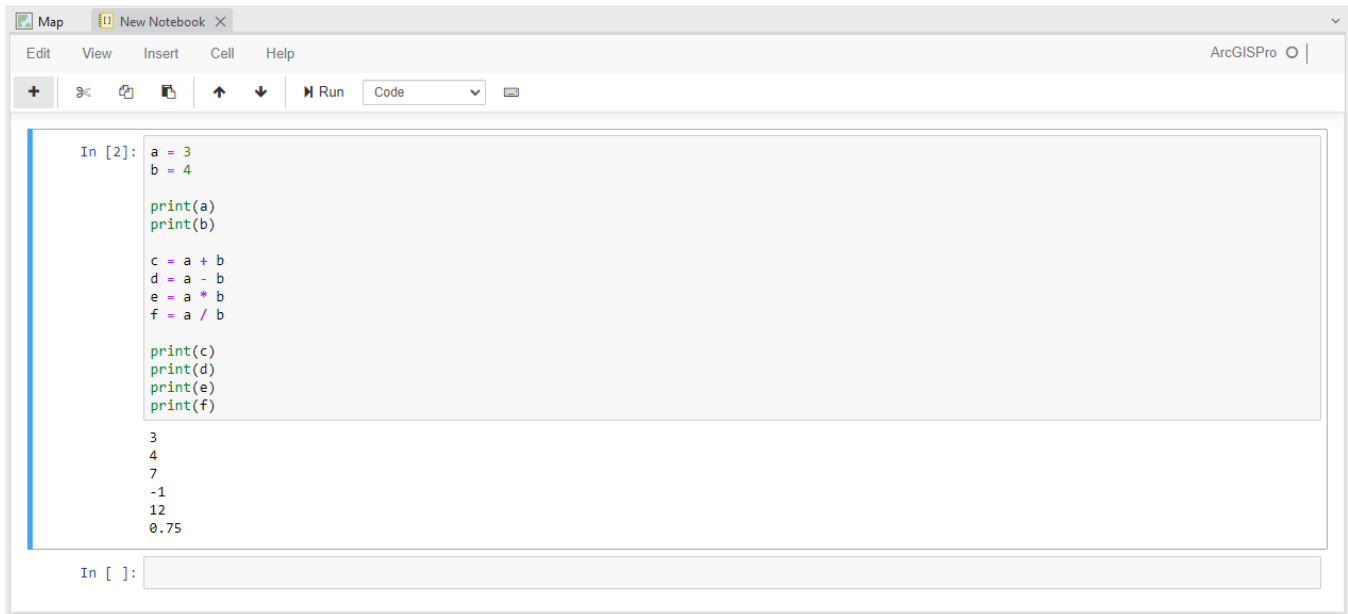
```
a = 3
b = 4
print(a)
print(b)
```

The output of the code is displayed below the code cell:

```
3
4
```

Below the output, there is an input field for the next code cell, labeled 'In []:'.

Now, we can run some basic arithmetic operations with these numbers. The operations shown below correspond to addition, subtraction, multiplication, and division, respectively. After printing each of the corresponding variables, we can see the results.



The screenshot shows the ArcGIS Pro Notebook interface. The top menu bar includes 'Map', 'New Notebook', 'Edit', 'View', 'Insert', 'Cell', and 'Help'. The right side of the menu bar shows 'ArcGISPro' and a search icon. Below the menu bar is a toolbar with icons for adding a new cell, undo, redo, and running the code. The main area contains a code cell labeled 'In [2]:'. The code in the cell is:

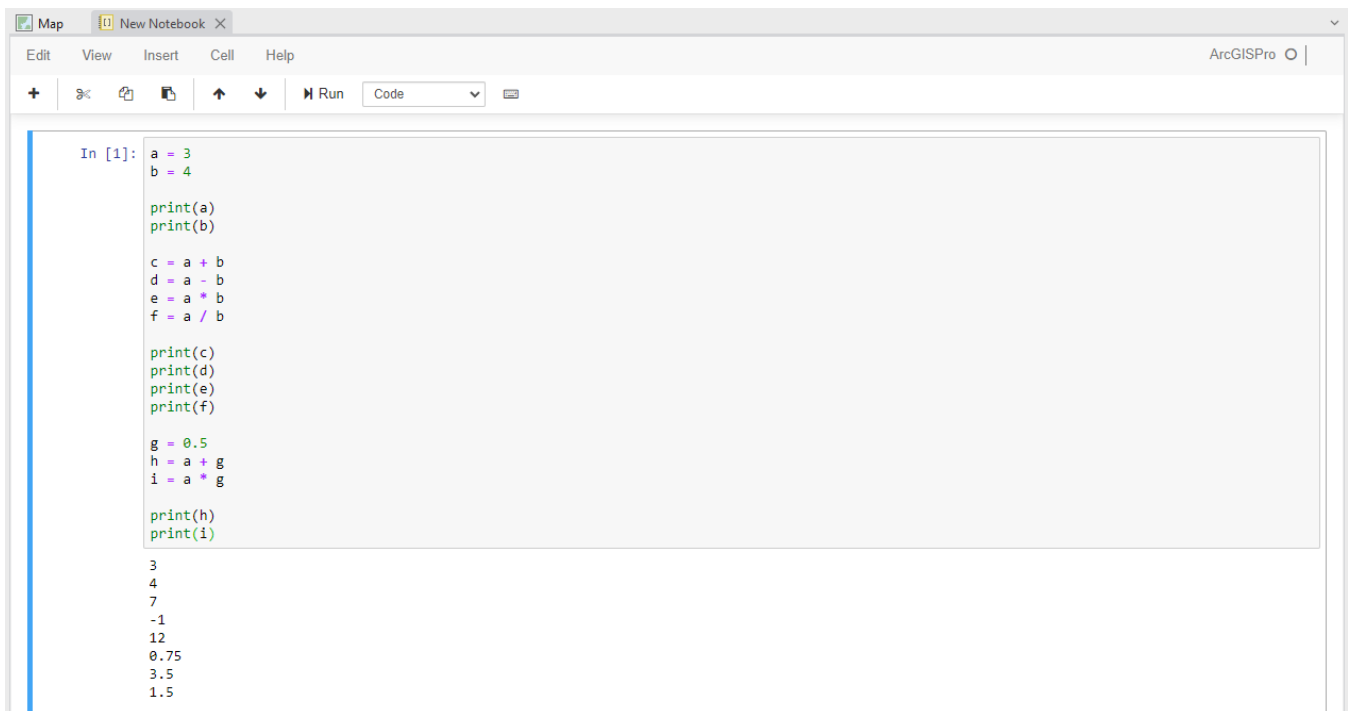
```
a = 3
b = 4
print(a)
print(b)
c = a + b
d = a - b
e = a * b
f = a / b
print(c)
print(d)
print(e)
print(f)
```

The output of the code is displayed below the code cell:

```
3
4
7
-1
12
0.75
```

Below the output, there is an input field for the next code cell, labeled 'In []:'.

We can treat decimal numbers in a very similar way, and combine them with integers.



```
In [1]: a = 3
b = 4

print(a)
print(b)

c = a + b
d = a - b
e = a * b
f = a / b

print(c)
print(d)
print(e)
print(f)

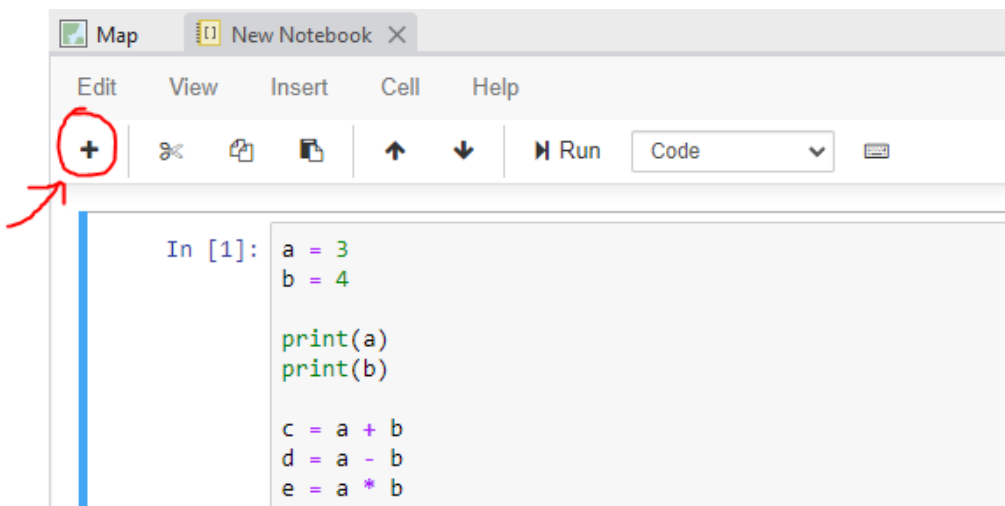
g = 0.5
h = a + g
i = a * g

print(h)
print(i)
```

3
4
7
-1
12
0.75
3.5
1.5

Booleans

Before we start looking at our next data type (Booleans), we can start a new cell to separate our previous code with our new code. To do this, press the "+" button in the Notebook menu.



```
In [1]: a = 3
b = 4

print(a)
print(b)

c = a + b
d = a - b
e = a * b
```

Boolean variables are the simplest type of variables- any Boolean is either True or False. We can code them as such here:



```
In [2]: j = True
k = False

print(j)
print(k)
```

True
False

We can combine them using keywords "and" and "or". If two Boolean variables *i* and *j* are both True, then "*i* and *j*" results in True. In any other case, "*i* and *j*" results in False.

```
In [3]: j = True
        k = False

        print(j)
        print(k)

        l = True
        print(j and k)
        print(j and l)

True
False
False
True
```

If either of the two Boolean variables *i* and *j* are True, then "*i* or *j*" results in True. If both *i* and *j* are False, then "*i* or *j*" results in False.

```
In [4]: j = True
        k = False

        print(j)
        print(k)

        l = True
        print(j and k)
        print(j and l)

        m = False
        print(j or k)
        print(k or m)

True
False
False
True
True
False
```

Lists

Python lists store multiple other Python variables inside one variable- simply put, they are simply lists of variables. It is recommended, though not enforced, that all of these variables are the same type.

Strings

Strings in Python are simply lists of characters, which are letters, numbers, or other symbols enclosed in quotes.

Dictionaries

Dictionaries in Python

Functions

Conditional Statements

Loops

Using Python in ArcGIS

Field Calculations

Fields in ArcGIS Pro

Why Use Python?

Switching Data Types

Defining a Field from Other Fields

Notebooks

Jupyter Notebooks

Jupyter Notebooks in ArcGIS Pro

Automating Workflow Using Jupyter Notebooks

Here, we'll go through a short sample project to practice using some simple tools using Python in ArcGIS Pro notebooks.

Our goal for this project will be to take a collection of schools throughout the United States, and create a layer containing the average center of these schools for each state / territory. The approach we'll take isn't necessarily the most efficient way of doing this, but will allow us to look at a wide variety of ways to use Python in ArcGIS Pro.

The collection of schools we'll be working with can be found at O:\GDCTraining\1_Short_Courses\Python_and_ArcGIS\StateSchools.gdb\PublicSchools10K. The layer of US states / territories can be found at O:\GDCTraining\1_Short_Courses\Python_and_ArcGIS\StateSchools.gdb\USStates. The original source of the schools is from (hifld-geoplatform.opendata.arcgis.com/datasets/public-schools), which was reduced to a collection of 10,000 schools for the sake of time, as these tools can take a long time to run, regardless of whether or not Python is being used. Alternatively, these datasets can be downloaded from this [file](#).

screengrabs:

```
In [20]: # Identify a List of all states to work with
states = arcpy.ListFeatureClasses()
print(states)

['USStates', 'PublicSchools10K', 'Florida', 'Illinois', 'Minnesota', 'Maryland', 'Idaho', 'New_Hampshire', 'North_Carolina', 'New_Mexico', 'California', 'New_Jersey', 'Oregon', 'Nebraska', 'Washington', 'Louisiana', 'Alabama', 'Utah', 'Ohio', 'Texas', 'Oklahoma', 'Tennessee', 'Wyoming', 'North_Dakota', 'Kentucky', 'Guam', 'New_York', 'Nevada', 'Alaska', 'Michigan', 'Arkansas', 'Mississippi', 'Missouri', 'Montana', 'Kansas', 'Indiana', 'Puerto_Rico', 'South_Dakota', 'Massachusetts', 'District_of_Columbia', 'Iowa', 'Arizona', 'West_Virginia', 'Rhode_Island', 'Connecticut', 'Delaware', 'Wisconsin', 'Pennsylvania', 'Georgia', 'Colorado', 'South_Carolina', 'Maine', 'Virginia', 'Vermont', 'Hawaii', 'United_States_Virgin_Islands', 'American_Samoa']
```

```
In [22]: # Identify a List of all states to work with
states = arcpy.ListFeatureClasses()
# print(states)

states = states[2:]
print(states)

['Florida', 'Illinois', 'Minnesota', 'Maryland', 'Idaho', 'New_Hampshire', 'North_Carolina', 'New_Mexico', 'California', 'New_Jersey', 'Oregon', 'Nebraska', 'Washington', 'Louisiana', 'Alabama', 'Utah', 'Ohio', 'Texas', 'Oklahoma', 'Tennessee', 'Wyoming', 'North_Dakota', 'Kentucky', 'Guam', 'New_York', 'Nevada', 'Alaska', 'Michigan', 'Arkansas', 'Mississippi', 'Missouri', 'Montana', 'Kansas', 'Indiana', 'Puerto_Rico', 'South_Dakota', 'Massachusetts', 'District_of_Columbia', 'Iowa', 'Arizona', 'West_Virginia', 'Rhode_Island', 'Connecticut', 'Delaware', 'Wisconsin', 'Pennsylvania', 'Georgia', 'Colorado', 'South_Carolina', 'Maine', 'Virginia', 'Vermont', 'Hawaii', 'United_States_Virgin_Islands', 'American_Samoa']
```

```
In [23]: # Go through all states and find the mean center
for state in states:
    input_fc = out_workspace + "\\" + state
    output_fc = "P:\\Staff_Folders\\Jacob\\PythonCourseSandbox\\MeanCenters.gdb\\" + state + "MC"
    arcpy.stats.MeanCenter(input_fc, output_fc)
```

```
In [25]: # Identify all Mean Center feature classes
arcpy.env.workspace = "P:\\Staff_Folders\\Jacob\\PythonCourseSandbox\\MeanCenters.gdb"
mc_fcs = []
for mc in arcpy.ListFeatureClasses():
    mc_fcs.append(mc)
print(mc_fcs)

['FloridaMC', 'IllinoisMC', 'MinnesotaMC', 'MarylandMC', 'IdahoMC', 'New_HampshireMC', 'North_CarolinaMC', 'New_MexicoMC', 'CaliforniaMC',
'New_JerseyMC', 'OregonMC', 'NebraskaMC', 'WashingtonMC', 'LouisianaMC', 'AlabamaMC', 'UtahMC', 'OhioMC', 'TexasMC', 'OklahomaMC', 'Tennesse
eMC', 'WyomingMC', 'North_DakotaMC', 'KentuckyMC', 'GuamMC', 'New_YorkMC', 'NevadaMC', 'AlaskaMC', 'MichiganMC', 'ArkansasMC', 'MississippiM
C', 'MissouriMC', 'MontanaMC', 'KansasMC', 'IndianaMC', 'Puerto_RicoMC', 'South_DakotaMC', 'MassachusettsMC', 'District_of_ColumbiaMC', 'Iow
aMC', 'ArizonaMC', 'West_VirginiaMC', 'Rhode_IslandMC', 'ConnecticutMC', 'DelawareMC', 'WisconsinMC', 'PennsylvaniaMC', 'GeorgiaMC', 'Colora
doMC', 'South_CarolinaMC', 'MaineMC', 'VirginiaMC', 'VermontMC', 'HawaiiMC', 'United_States_Virgin_IslandsMC', 'American_SamoaMC']
```