Last updated by Taylor Hixson, 17 November 2019

Python for ArcGIS

Terms to know

IDE: [programming] Acronym for **integrated development environment**. A software development tool for creating applications, such as desktop and Web applications. IDEs blend user interface design and layout tools with coding and debugging tools. [Source: Esri]

Geodatabase: [ESRI software] A database or file structure used primarily to store, query, and manipulate spatial data. Geodatabases store geometry, a spatial reference system, attributes, and behavioral rules for data. Various types of geographic datasets can be collected within a geodatabase, including feature classes, attribute tables, raster datasets, network datasets, topologies, and many others. Geodatabases can be stored in IBM DB2, IBM Informix, Oracle, Microsoft Access, Microsoft SQL Server, and PostgreSQL relational database management systems, or in a system of files, such as a file geodatabase. [Source: Esri]

Feature class: [ESRI software] In ArcGIS, a collection of geographic features with the same geometry type (such as point, line, or polygon), the same attributes, and the same spatial reference. Feature classes can be stored in geodatabases, shapefiles, coverages, or other data formats. Feature classes allow homogeneous features to be grouped into a single unit for data storage purposes. For example, highways, primary roads, and secondary roads can be grouped into a line feature class named "roads." In a geodatabase, feature classes can also store annotation and dimensions. [Source: Esri]

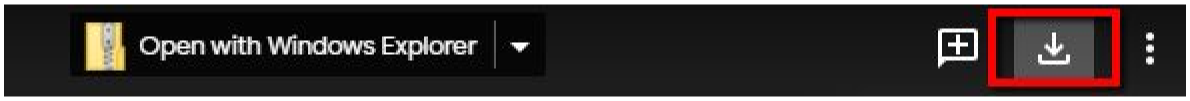
Python: Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level built in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together. Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity and code reuse. The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms, and can be freely distributed. [Source: Python.org]

ArcMap: The central application used in ArcGIS. ArcMap is where you display and explore GIS datasets for your study area, where you assign symbols, and where you create map layouts for printing or publication. ArcMap is also the application you use to create and edit datasets. [Source: Esri]

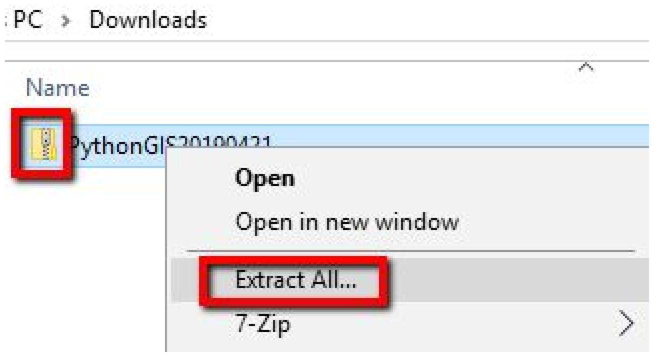
ArcPy: a site package that builds on (and is a successor to) the successful arcgisscripting module. Its goal is to create the cornerstone for a useful and productive way to perform geographic data analysis, data conversion, data management, and map automation with Python. ArcPy provides Python access for all geoprocessing tools, including extensions, as well as a wide variety of useful functions and classes for working with and interrogating GIS data. Using Python and ArcPy, you can develop an infinite number of useful programs that operate on geographic data. [Source: Esri]

Downloading workshop data

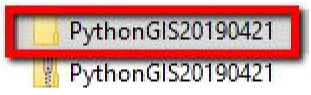
1. **Download** the data for this workshop from the link: **tiny.cc/0b974y**
   Click the **down arrow** in the top right to download the zip file.



2. **Navigate** to where the folder is downloaded. For example, the **Downloads** folder.
3. **Unzip** the file by **right clicking** it and choosing **Extract All...**.



4. Drag the **unzipped** folder into either an **external flash drive**, the computer's **C: drive**, the **D:/SAVE HERE drive**, or any root directory (e.g., E: drive, D: drive, but **NOT** Google Drive).



Make sure data is in a root directory.

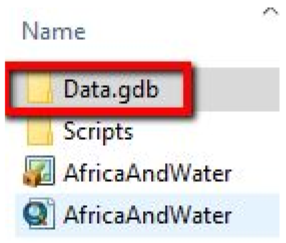
The **data** for this workshop **MUST** be located in a root directory. To follow along with the tutorial, **data** (Data.gdb folder) **MUST** be in a folder directly below the root directory, or something that looks like **C:\** or **D:\**, or any letter followed by a colon and a backslash (but **not** a cloud drive like Google Drive).

! This **will work**: **C:\PythonGIS\Data.gdb**

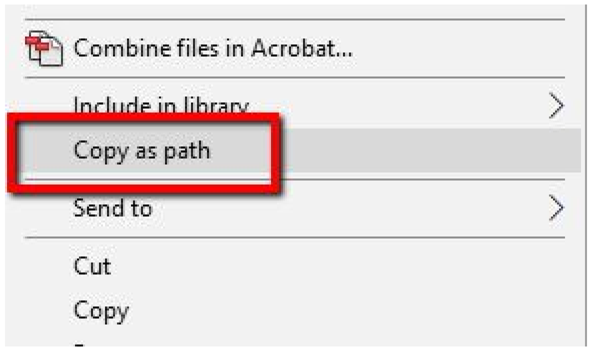
X This will **NOT** work: C:\Users\user2\Desktop\PythonGIS\Data.gdb

Running Python Scripts from an IDE

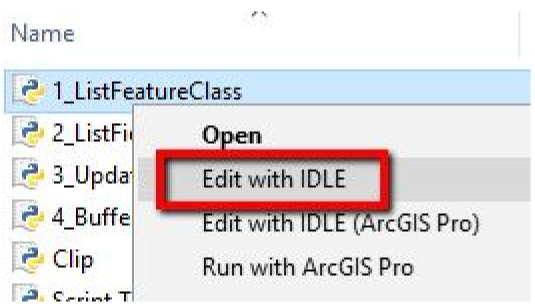
5. Navigate to the **workshop** folder, which should be in a root directory.
6. Hold down **shift** on the keyboard and **right click** the **Data.gdb folder.**



7. Click **Copy as path**.

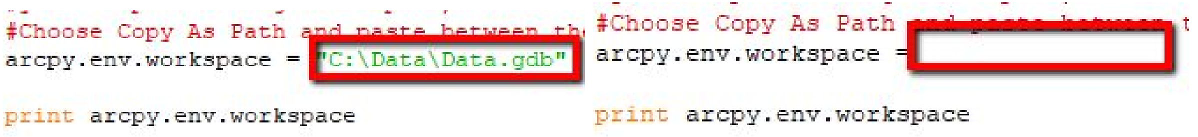


8. Open the **Scripts folder**.
9. Right click **1_ListFeatureClass** and choose **Edit with IDLE**.



10. **Delete** the file **path** in **Line 7**. (Image below)

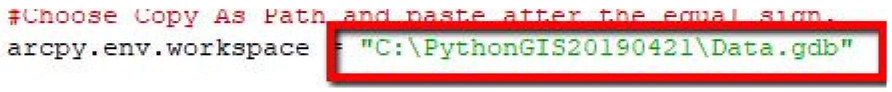
    The line number the cursor is on is shown in the bottom right corner of IDLE.



```
#Choose Copy As Path and paste between th
arcpy.env.workspace = "C:\Data\Data.gdb"

print arcpy.env.workspace
```

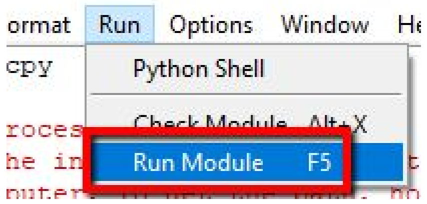
```
#Choose Copy As Path and paste between t
arcpy.env.workspace =

print arcpy.env.workspace
```

11. **Paste** the path that was copied in **Step 6** after the equal sign. The file path should show up as **green** and within **quotation marks**.

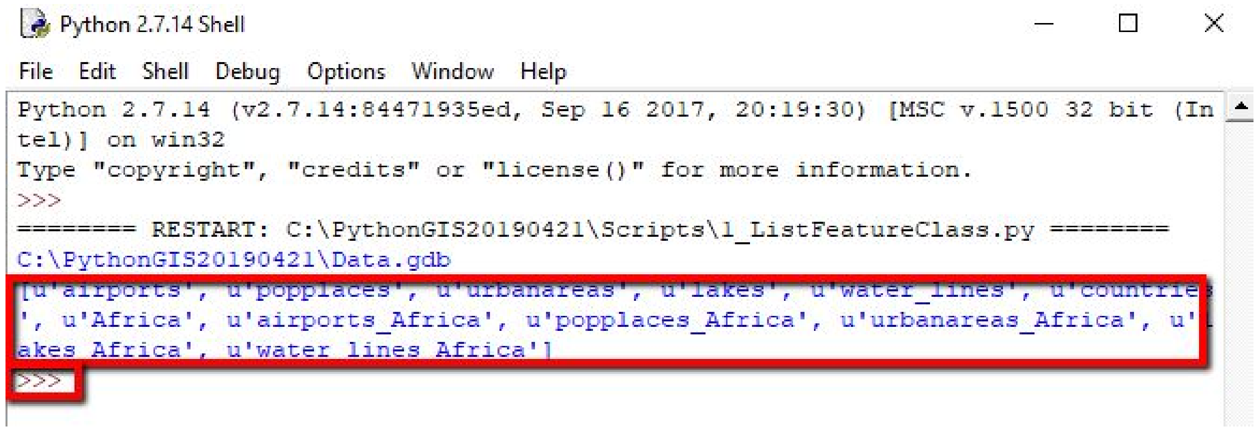


```
#Choose Copy As Path and paste after the equal sign.
arcpy.env.workspace = "C:\PythonGIS20190421\Data.gdb"
```

12. Save the updated script using **Ctrl + S** or **File>Save**.
13. From the menu, select **Run > Run Module**, or press **F5** on the keyboard.



    A new window, which is called the **Shell**, appears. It will take a moment for the script to run and for the feature classes to be listed.

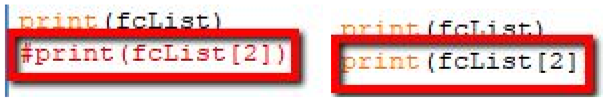


```
Python 2.7.14 (v2.7.14:84471935ed, Sep 16 2017, 20:19:30) [MSC v.1500 32 bit (In
tel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
======== RESTART: C:\PythonGIS20190421\Scripts\1_ListFeatureClass.py ========
C:\PythonGIS20190421\Data.gdb
[u'airports', u'popplaces', u'urbanareas', u'lakes', u'water_lines', u'countrie
', u'Africa', u'airports_Africa', u'popplaces_Africa', u'urbanareas_Africa', u'l
akes_Africa', u'water_lines_Africa']
>>>
```

    You will also know a script has **run completely** when the three arrows (**>>>**) appears at the bottom of the shell. The three arrows **do not** indicate success, just that the script has run.

14. Back in the script editor (look for 1_ListFeatureClass.py) , **uncomment** the last line by deleting the **#**.

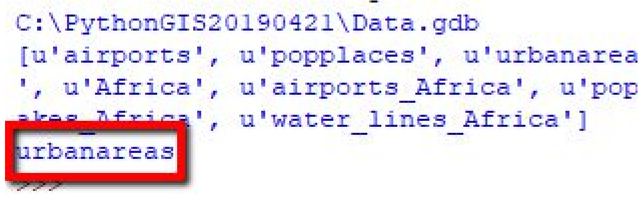
When the hashmark (#) is deleted, it will go from red to orange and black.



```
print(fcList)
#print(fcList[2])
```

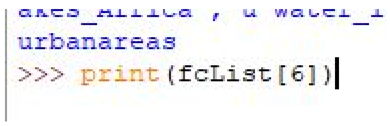
```
print(fcList)
print(fcList[2]
```

15. Save the script by pressing **Ctrl + S** or **File > Save**.
16. Run the script again by pressing **F5** or clicking **Run > Run Module**. Return to **Step 12** for an image if needed.
17. In the **Shell** window, the output is now not only the list of all feature classes in the geodatabase but also the feature class at **index 2** (or the **third** feature class), which is **urbanareas**, is printed.



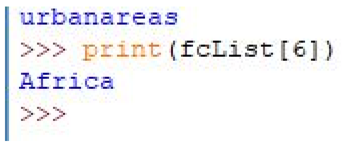
```
C:\PythonGIS20190421\Data.gdb
[u'airports', u'popplaces', u'urbanarea
', u'Africa', u'airports_Africa', u'pop
akes_Africa', u'water_lines_Africa']
urbanareas
```

18. In the **Shell**, click to the right of the three arrows (**>>>**), and type: **print(fcList[6])**



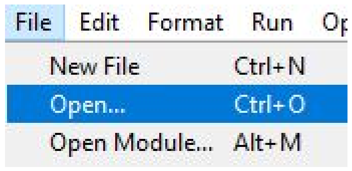
```
urbanareas
>>> print(fcList[6])
```

19. Press **Enter** on the keyboard. What feature class, or dataset, is at the sixth index in the geodatabase?

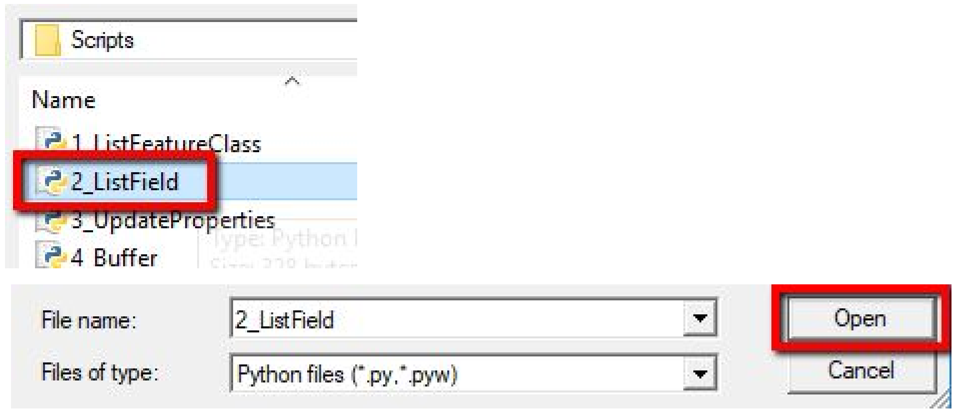


```
urbanareas
>>> print(fcList[6])
Africa
>>>
```

20. Close the **Shell** window.
21. From the IDLE editor (look for **1_ListFeatureClass.py**), click **File > Open…**.

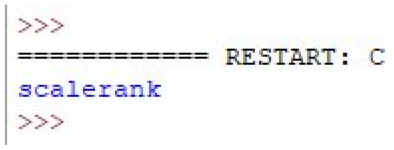


22. Navigate to the scripts folder for this workshop and click **2_ListField** and **Open**.



File name: 2_ListField
Files of type: Python files (*.py,*.pyw)

23. In the new window that appears (look for 2_ListField.py), change the information after **arcpy.env.workspace =** to the location of **your workspace**. Copy from 1_ListFeatureClass.py if it is still open, or return to **Steps 6 and 7.**
24. Press **F5**.

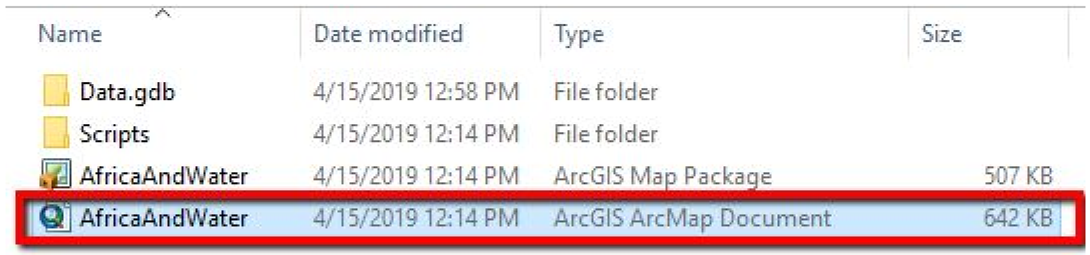


```
>>>
============ RESTART: C
scalerank
>>>
```

> **What is this script doing?**
> This script creates the variable fieldList, which specifies a feature class to list the fields (columns) using arcpy.ListFields(). Calling fieldList[2] will identify the column name at the specified index, which in this case is 2. Indices start at 0, so asking for index 2 will give the third column.
> In **line 11** where the **print(fieldList[2]**) command is, **change** the **2** to **3**. Save and run the script again. What is the name of the Lakes feature class's column at index 3, or the fourth column? Being able to list datasets and dataset fields (columns) is great to do when needing to know a small amount of information about large datasets that take long to open and visualize in ArcMap.

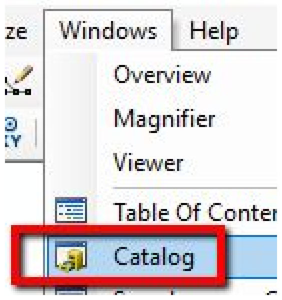
Using Python to manipulate map labels

25. Navigate to the unzipped **workshop** folder.
26. Double click the **AfricaAndWater** file that has the type **ArcGIS ArcMap Document**.
    If the layers listed in the right side Table of Contents **show up with red x's**, open the AfricaAndWater file with the **Type ArcGIS Map Package**.

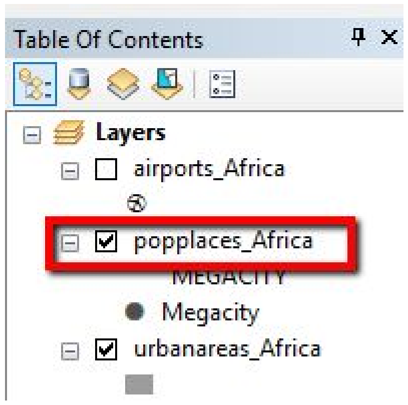


Name	Date modified	Type	Size
Data.gdb	4/15/2019 12:58 PM	File folder	
Scripts	4/15/2019 12:14 PM	File folder	
AfricaAndWater	4/15/2019 12:14 PM	ArcGIS Map Package	507 KB
AfricaAndWater	4/15/2019 12:14 PM	ArcGIS ArcMap Document	642 KB

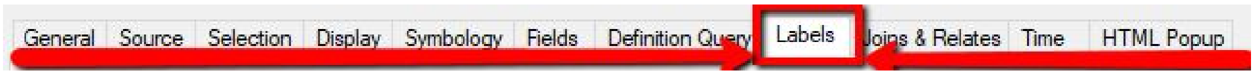
27. When ArcMap opens. Make sure the ArcCatalog is open. If ArcCatalog is not open or you are not certain, in the top menu click **Windows > Catalog**.



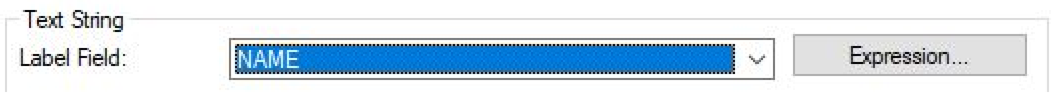
28. If the **Table of Contents** panel is not open, in top menu click **Windows > Table of Contents**.
29. From the **Table of Contents** panel, double click **popplaces_Africa**.



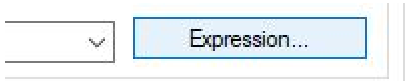
30. In the **Properties** window that appears, click the **Labels tab**.



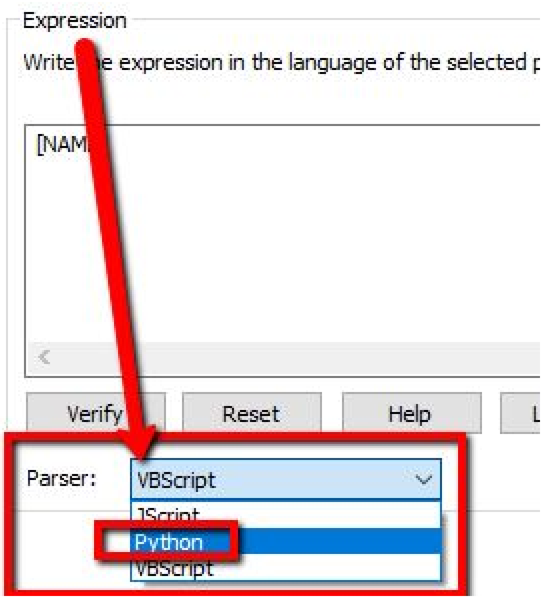
31. In the **Text String** section, click the Label Field dropdown and choose **NAME**.


Text String
Label Field: NAME

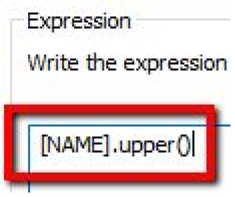
32. Once **NAME** is selected in the dropdown, click **Expression**.



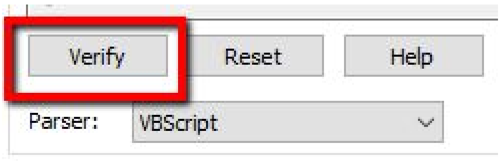
33. In the **Expression** section of the **Label Expression** window, click the **Parser** dropdown and choose **Python**.



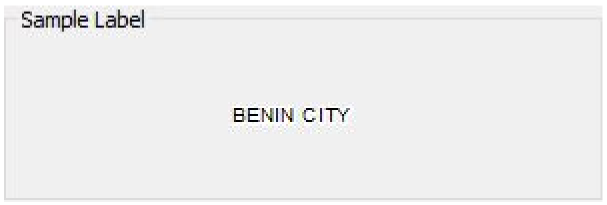
34. In the **Expression** box after the bracketed expression type **.upper()** after **[Name]** so that it appears as **[Name].upper()**.



35. Click **Verify**.

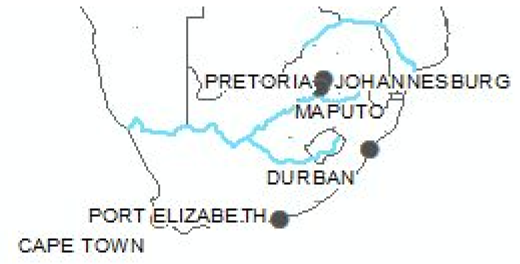


36. The window will say if the expression syntax is correct (the syntax may be correct, but it may not always be what you want, so the **Expression Verification** window will show what a **Sample Label** looks like.



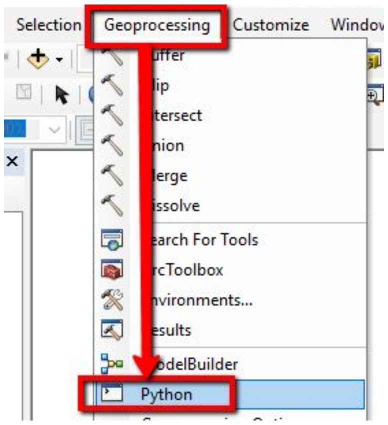
37. If correct, click **OK.**
38. Click **OK** again in the **Label Expression** window.
39. Click **OK** again in the **Layer Properties** window.

    Now, all the Megacities in the map are capitalized. This may not be visually ideal, but it is one example of integrating Python into a geospatial workflow beyond analytics and outside of an IDE.

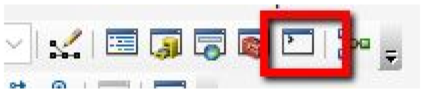


Running Scripts in ArcMap

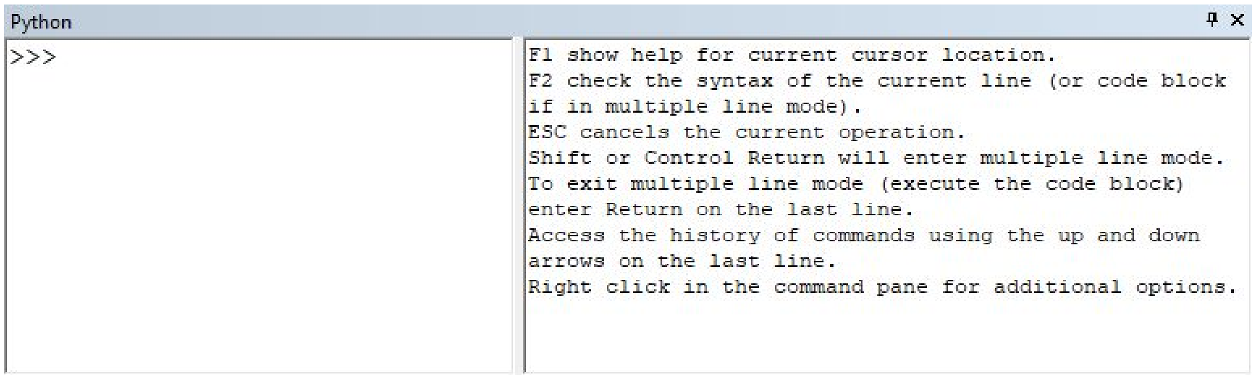
40. From the top menu, click **Geoprocessing > Python** to open the built-in Python environment in ArcMap.



Alternatively, click the **Python icon** in **Standard** toolbar.



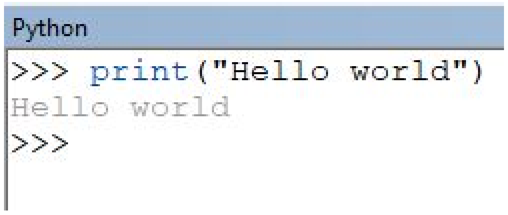
The Python window will either be floating or docked somewhere within the ArcMap interface.



```
>>>
```

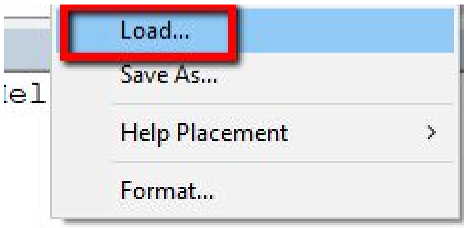
```
F1 show help for current cursor location.
F2 check the syntax of the current line (or code block
if in multiple line mode).
ESC cancels the current operation.
Shift or Control Return will enter multiple line mode.
To exit multiple line mode (execute the code block)
enter Return on the last line.
Access the history of commands using the up and down
arrows on the last line.
Right click in the command pane for additional options.
```

41. After the three arrows (**>>>**), type in: **print "Hello world"** (for best results, do not copy this text). This will print the string Hello world.

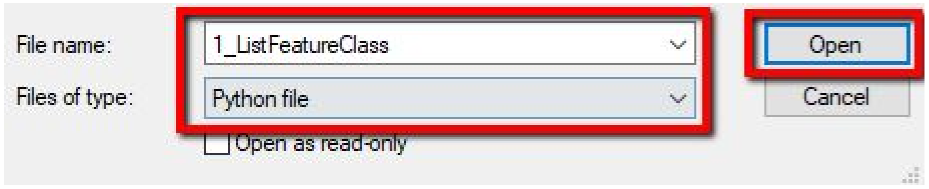


```
Python
>>> print("Hello world")
Hello world
>>>
```

42. **Right click** in the Python **window**, and choose **Load...**.

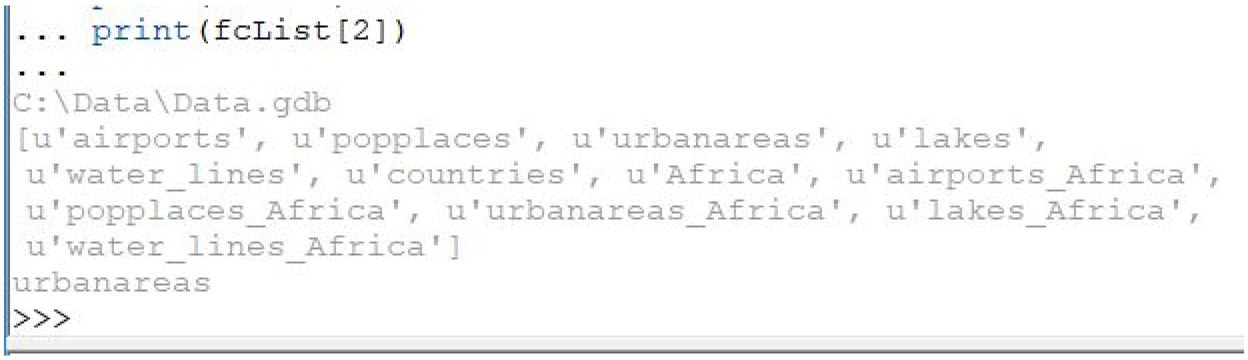


43. In the window that appears, **navigate** to the **Scripts** folder within the workshop folder, and find **1_ListFeatureClass** Python script.
44. Click on it **once** so it appears in the **File name** dropdown box, and click **Open**.



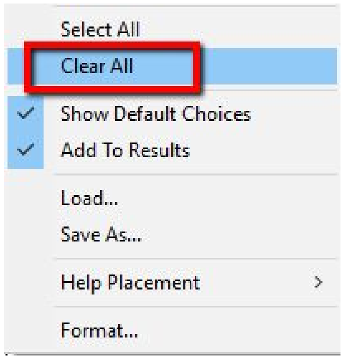
File name: 1_ListFeatureClass
Files of type: Python file

45. Press **Enter** on the keyboard.
    Notice that these are the same results from when the script was run in IDLE.



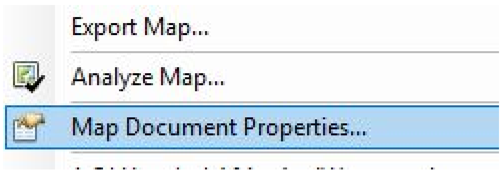
```
... print(fcList[2])
...
C:\Data\Data.gdb
[u'airports', u'popplaces', u'urbanareas', u'lakes',
 u'water_lines', u'countries', u'Africa', u'airports_Africa',
 u'popplaces_Africa', u'urbanareas_Africa', u'lakes_Africa',
 u'water_lines_Africa']
urbanareas
>>>
```

46. **Right click** in the Python window, and choose **Clear All**.



Using Python to update map properties

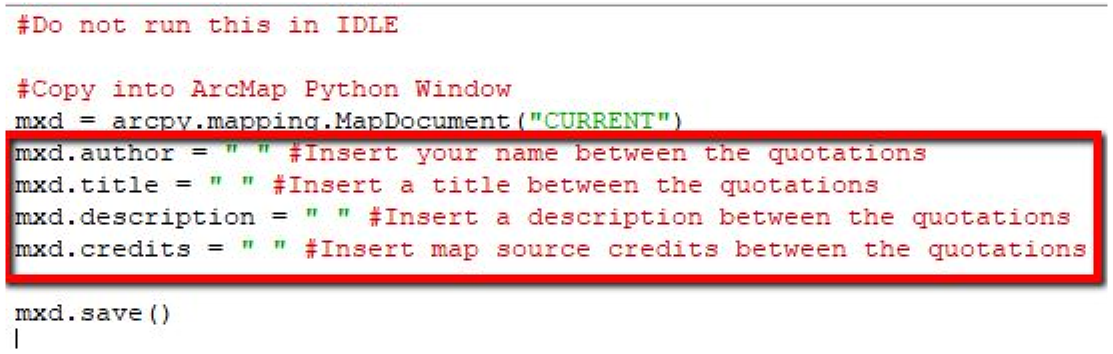
47. In the top-level menu, click **File > Map Document Properties....**



Notice the author is currently Taylor Hixson. Let's **change** it to your name **using Python**! Click **OK** to close the window.

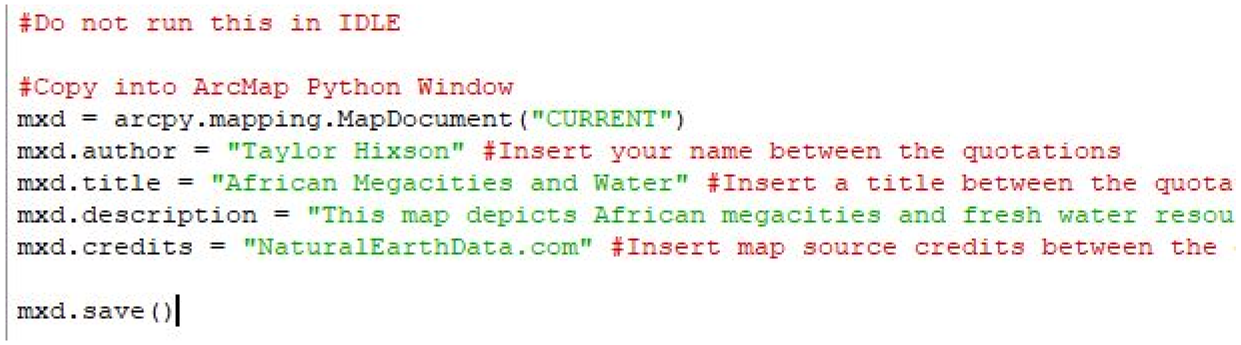
48. Outside of ArcMap, navigate to the **Scripts** folder.
49. Right click **3_UpdateProperties** and choose **Edit with IDLE**.
50. Follow the comments to insert the author, title, description, and credits Map Document Properties.

Before:



```
#Do not run this in IDLE

#Copy into ArcMap Python Window
mxd = arcpy.mapping.MapDocument("CURRENT")
mxd.author = " " #Insert your name between the quotations
mxd.title = " " #Insert a title between the quotations
mxd.description = " " #Insert a description between the quotations
mxd.credits = " " #Insert map source credits between the quotations

mxd.save()
```

After:

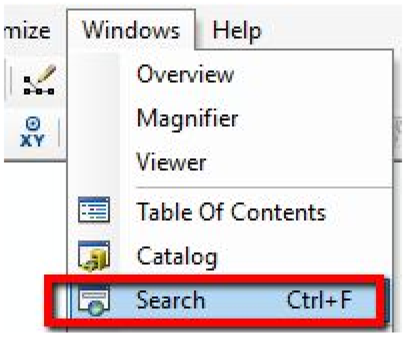


```
#Do not run this in IDLE

#Copy into ArcMap Python Window
mxd = arcpy.mapping.MapDocument("CURRENT")
mxd.author = "Taylor Hixson" #Insert your name between the quotations
mxd.title = "African Megacities and Water" #Insert a title between the quota
mxd.description = "This map depicts African megacities and fresh water resou
mxd.credits = "NaturalEarthData.com" #Insert map source credits between the

mxd.save()
```

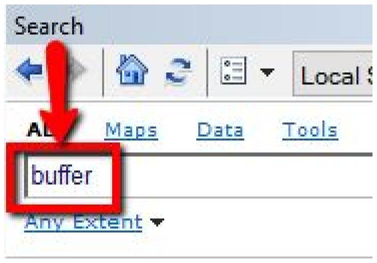
51. Once the information is complete **copy and paste** the **entire script**, including notes, into the **ArcPy window.** Press enter **twice**.
Alternatively, save and close the script, and right click within the ArcPy window, choosing **Load**. Press enter **once**.
52. Click **File > Map Document Properties...** again. Did the information change?

Generating a Python Snippet

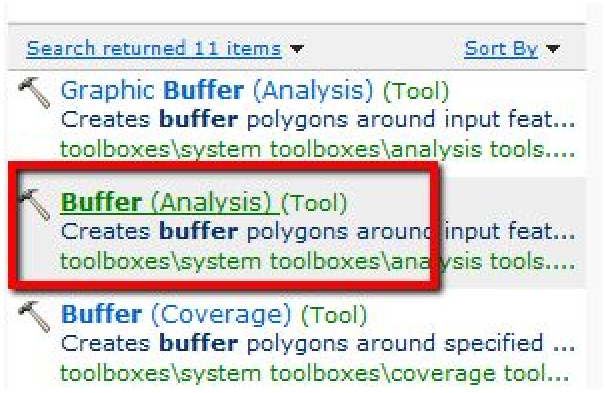
53. In the top menu click **Windows > Search**. Alternatively, press **Ctrl + F**. The **Search** panel will appear either docked near the Catalog or floating.



54. In the **Search** panel type **buffer** and press **Enter** or the magnifying glass to search.



55. From the results, hover over the second result **Buffer(Analysis)(Tool)** so that the **cursor** turns into a **hand.**

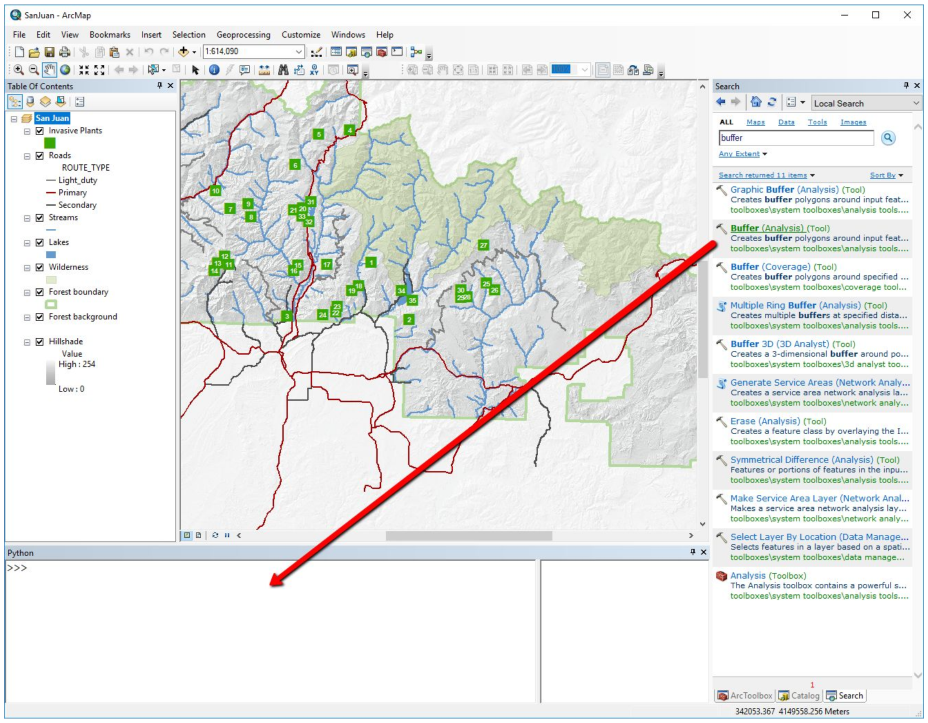

Search returned 11 items
Sort By

Graphic **Buffer** (Analysis) (Tool)
Creates **buffer** polygons around input feat...
toolboxes\system toolboxes\analysis tools....

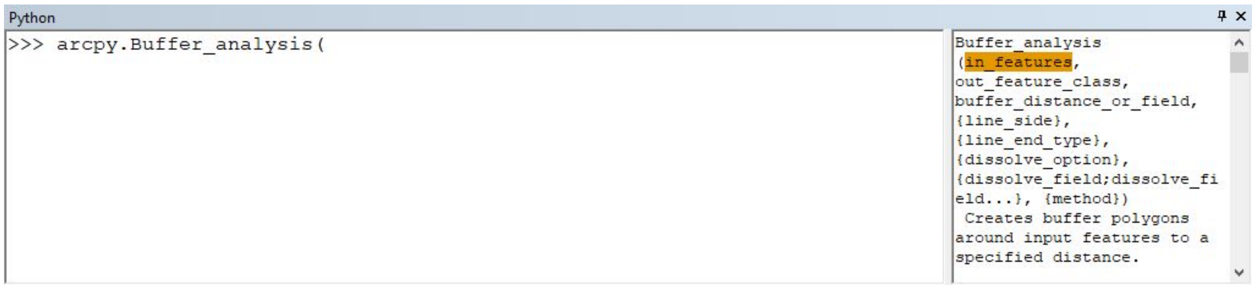
Buffer (Analysis) (Tool)
Creates **buffer** polygons around input feat...
toolboxes\system toolboxes\analysis tools....

Buffer (Coverage) (Tool)
Creates **buffer** polygons around specified ...
toolboxes\system toolboxes\coverage tool...

56. Click and drag that into the **Python window** (image follows).


SanJuan - ArcMap
File Edit View Bookmarks Insert Selection Geoprocessing Customize Windows Help
Table Of Contents
San Juan
Invasive Plants
Roads
ROUTE_TYPE
Light_duty
Primary
Secondary
Streams
Lakes
Wilderness
Forest boundary
Forest background
Hillshade
Value
High : 254
Low : 0
Search
Local Search
ALL Maps Data Tools Images
buffer
Search returned 11 items
Graphic Buffer (Analysis) (Tool)
Buffer (Analysis) (Tool)
Buffer (Coverage) (Tool)
Multiple Ring Buffer (Analysis) (Tool)
Buffer 3D (3D Analyst) (Tool)
Generate Service Areas (Network Analy...
Erase (Analysis) (Tool)
Symmetrical Difference (Analysis) (Tool)
Make Service Area Layer (Network Anal...
Select Layer By Location (Data Manage...
Analysis (Toolbox)
Python
>>>
ArcToolbox Catalog Search
342053.367 4149558.256 Meters

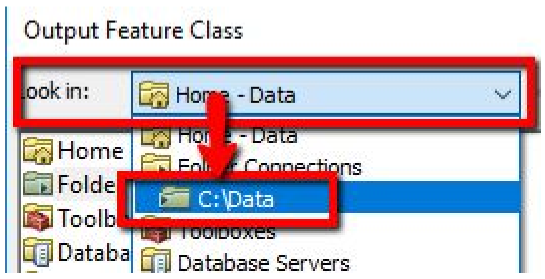
Notice that the window populates with the beginning of the Buffer analysis tool and the help box populates with the required fields for the Buffer analysis.


Python

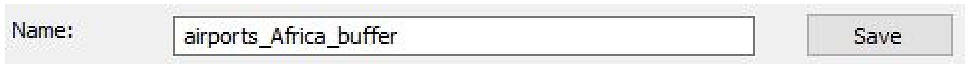
```
>>> arcpy.Buffer_analysis(
```

```
Buffer_analysis
(in_features,
out_feature_class,
buffer_distance_or_field,
{line_side},
{line_end_type},
{dissolve_option},
{dissolve_field;dissolve_fi
eld...}, {method})
 Creates buffer polygons
around input features to a
specified distance.
```

57. For now, **Right click** and choose **Clear all** OR **delete the line** in the **Python window**. We will perform the Buffer analysis through the graphical interface to get more **familiar** with the analysis **parameters**, which is a common method before deciding on how to automate or batch process data for your own analysis in ArcMap.
58. In the **Search** panel **click** the **Buffer(Analysis)(Tool)** link. It will open a new window. Alternatively, click **Geoprocessing > Buffer** from the top toolbar.
59. Fill in the parameters as follows:
    - **Input Features**: choose **airports_Africa** from the dropdown
    - **Output Feature Class**: click the yellow folder and **navigate** to the Data.gdb.



- Double click **Data.gdb**.
- Next to Name, type: airports_Africa_buffer

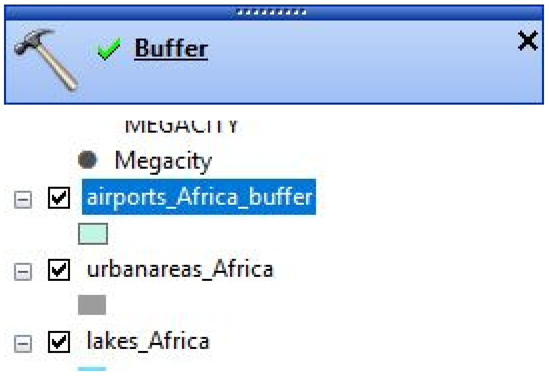


- Click **Save**.
- Under **Distance [value or field]**, check **Linear unit**.
- In the box below **Linear unit**, type **50**.
- In the drop down next to the box under **Linear unit**, choose **Kilometers**
- Leave all other parameters as is.
- Click **OK**.

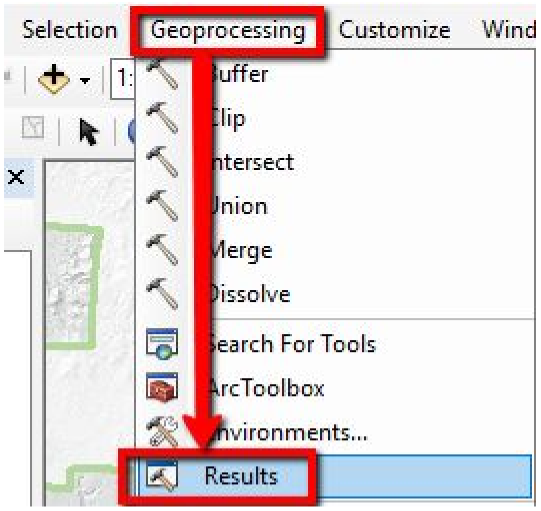
Is my data processing?

Processing data in ArcMap can take up to a few minutes depending on the computer speed and file size. You may not receive any visual indication of whether or not the process is working depending on your current window configuration. That is the **Results** panel is required to see any indication of processing time.

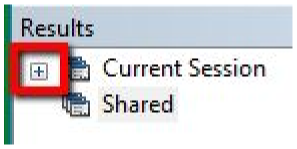
When a process, task, or analysis ends, you may receive a pop up with a **green check mark** and the analysis or geoprocessing name (e.g., Buffer) and a new layer appears in the Table of Contents called airports_Africa_buffer. New layers **do not** always appear in the Table of Contents, but it will with Buffer.



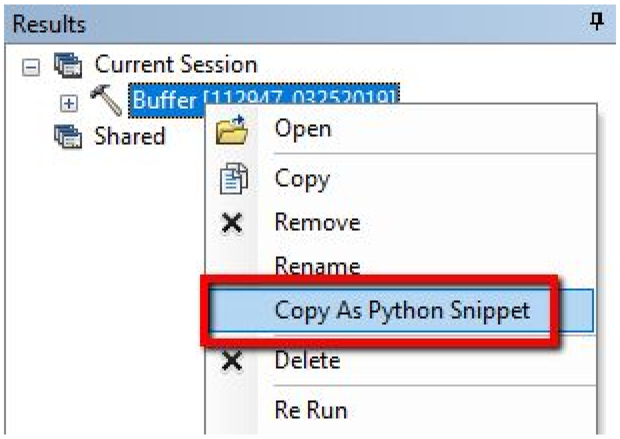
60. From the top menu, select **Geoprocessing > Results**.



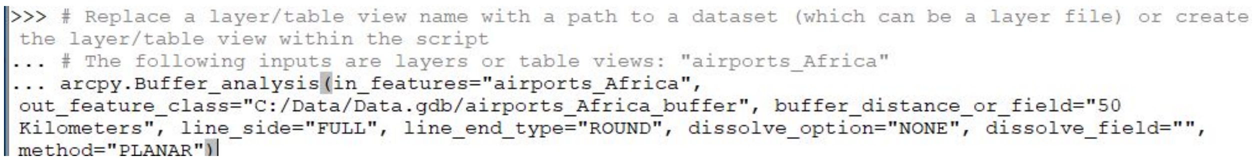
61. This opens a new panel labeled **Results**.
62. Click the tiny **plus sign** to the left of **Current Session** to reveal all geoprocessing that has taken place during this ArcMap session.



63. Right click **Buffer**.
64. Choose **Copy As Python Snippet**.



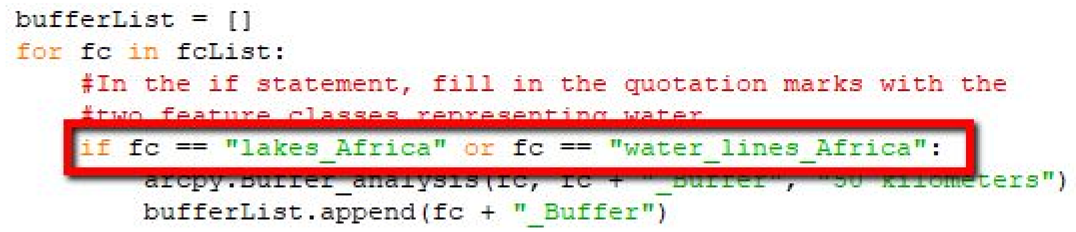
65. **Paste** (Ctrl + V) this into the **Python Window**. All the parameters are now filled in.
    It is now possible to use this snippet as the **basis** to automate the creation of buffers for the rest of the feature classes instead of going through one by one.



```
>>> # Replace a layer/table view name with a path to a dataset (which can be a layer file) or create
 the layer/table view within the script
... # The following inputs are layers or table views: "airports_Africa"
... arcpy.Buffer_analysis(in_features="airports_Africa",
 out_feature_class="C:/Data/Data.gdb/airports_Africa_buffer", buffer_distance_or_field="50
 Kilometers", line_side="FULL", line_end_type="ROUND", dissolve_option="NONE", dissolve_field="",
 method="PLANAR")
```

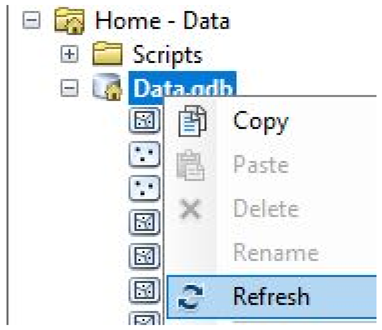
Batch geoprocessing

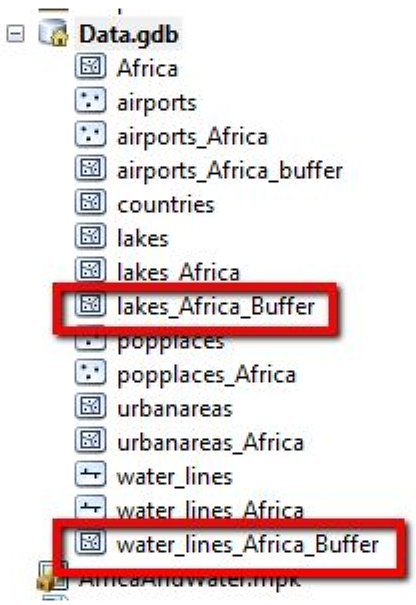
66. Return to the **Scripts** folder outside of ArcMap.
67. Right click **4_Buffer.py** and choose **Edit with IDLE**
68. Make sure the **file path** after the equal sign (=) of arcpy.env.workspace leads to where you have **Data.gdb** saved for this workshop.
69. In **line 15**, or where the **if statement** is **(if fc ==...)**, fill in the **two sets of quotations** with the **feature classes** in the geodatabase that represent **water**, which are **lakes_Africa** and **water_lines_Africa**.
    It should look like the below image when complete:



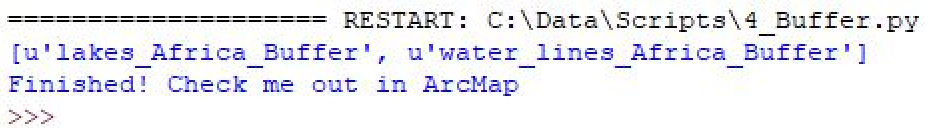
```
bufferList = []
for fc in fcList:
    #In the if statement, fill in the quotation marks with the
    #two feature classes representing water
    if fc == "lakes_Africa" or fc == "water_lines_Africa":
        arcpy.Buffer_analysis(fc, fc + "_Buffer", "50 kilometers")
        bufferList.append(fc + "_Buffer")
```

70. The **Shell** will **print** the **name** of the newly created feature classes when the Buffer analysis is complete.
    Alternatively, in the ArcMap catalog right click Data.gdb and choose **Refresh** to see if these were added to the geodatabase.





71. In lines **25** and **26** of **4_Buffer.py**, (starting with #arcpy.Union_analysis…), **delete the #.**
72. **Save** the code (Ctrl + S or File>Save).
73. **Run** the Code (F5 or Run > Run Module)
    If it ran successfully, the Shell should show the bufferList from the first task and a print statement saying "Finished! Check me out in ArcMap":

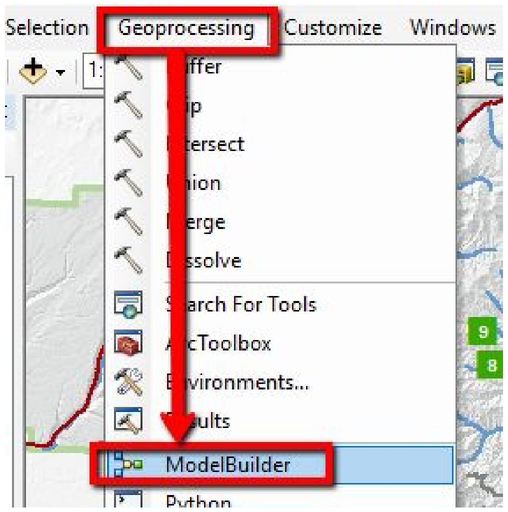


```
==================== RESTART: C:\Data\Scripts\4_Buffer.py
[u'lakes_Africa_Buffer', u'water_lines_Africa_Buffer']
Finished! Check me out in ArcMap
>>>
```

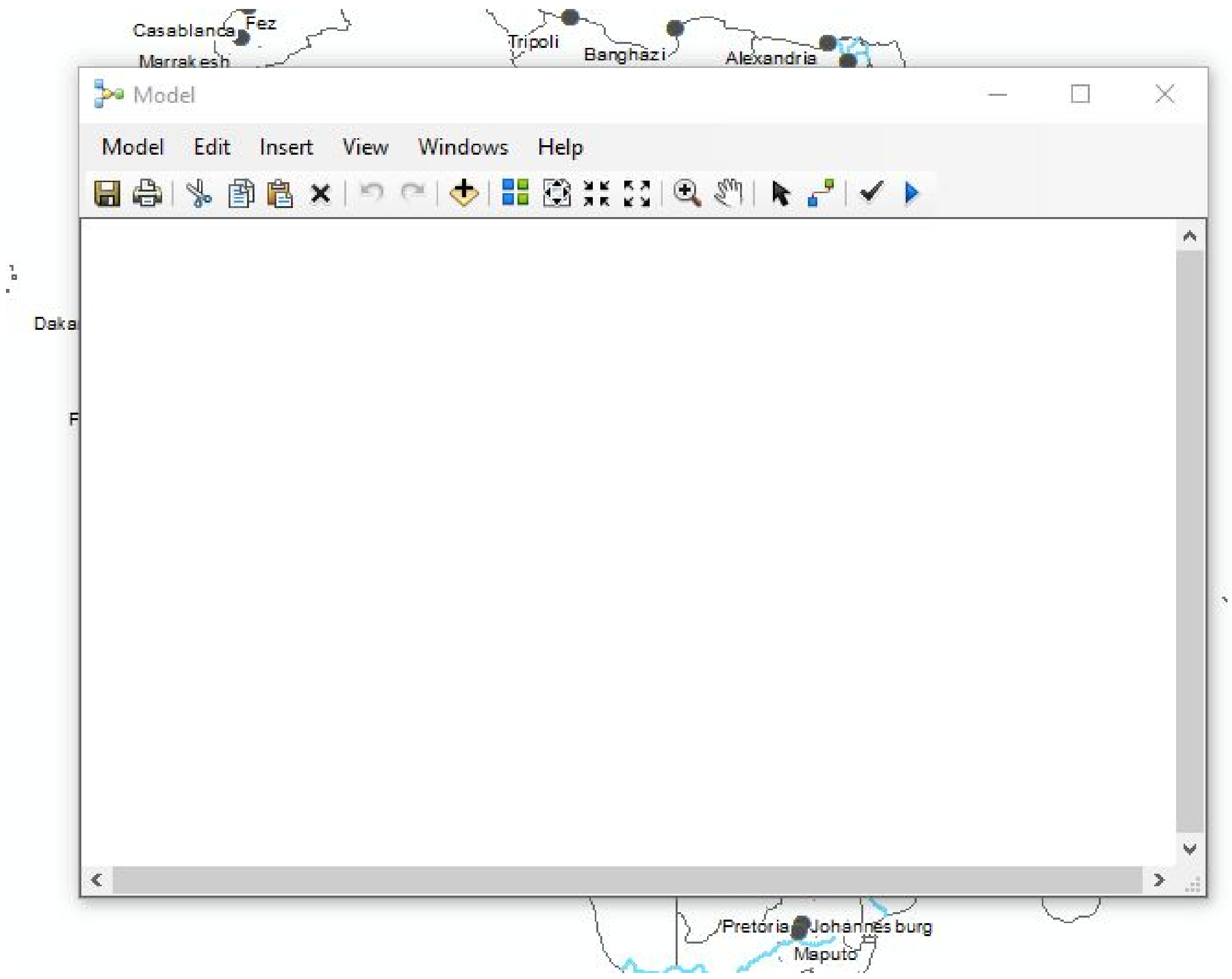
74. In the **Catalog** window of ArcMap, right click **Data.gdb** and choose **Refresh** to see if the new **WaterBufferAll** appears.
75. Drag **WaterBufferAll** from the Catalog into the Table of Contents. Double check the work by dragging in the water_lines_Africa_Buffer and lakes_Africa_Buffer. Do these align?

Export Python Scripts from ArcGIS Model Builder

76. From the top menu click **Geoprocessing>Model Builder**.

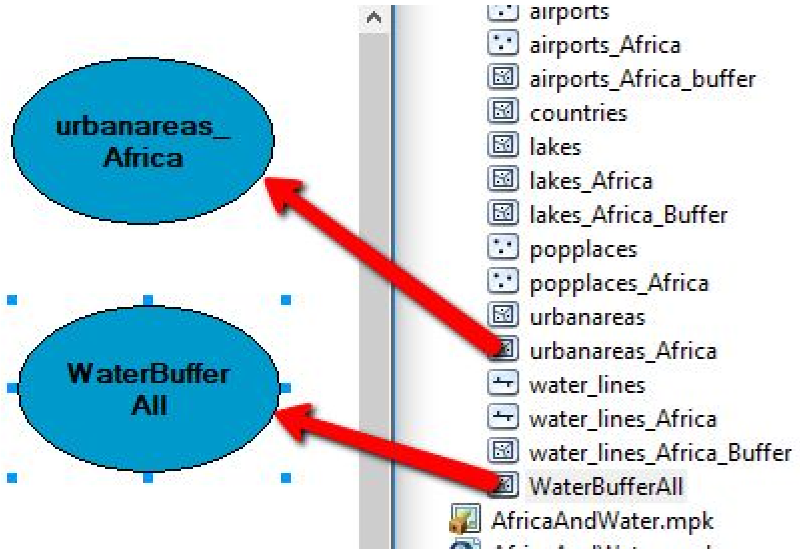


The **Model** window should appear floating.

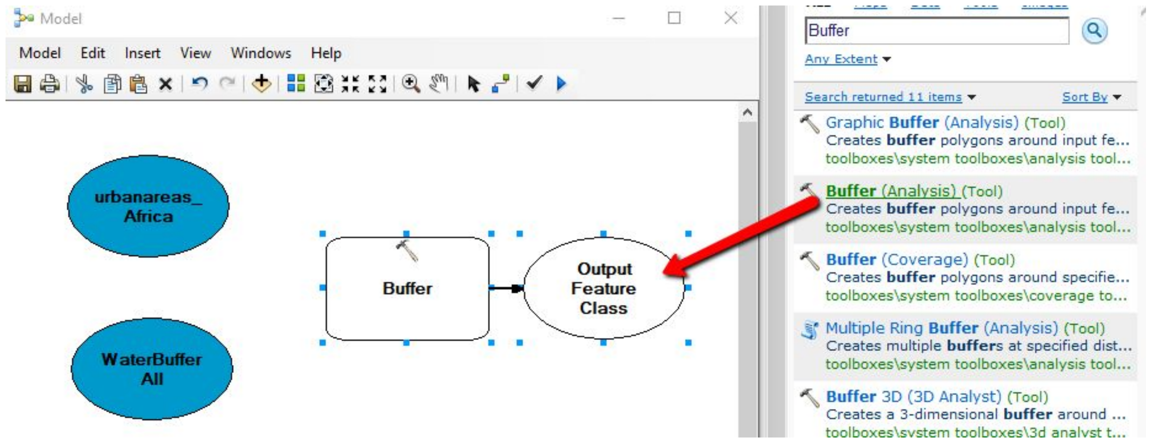


77. From the **Catalog** window find the **Data.gdb**. If needed, click the **plus sign** to reveal the data **feature classes**.
78. Click and drag **urbanareas_Africa** from the **Catalog** into the **Model** window.
79. Click and drag **WaterBufferAll** from the **Catalog** into the **Model** window.

If you did not generate the **WaterBufferAll** from the previous section, drag in any Buffered layer (e.g., airports_Africa_buffer, lakes_Africa_Buffer, or water_lines_Africa_Buffer)



80. From the **Search** panel, type **Buffer**.
    If **Search** is not open, use **Ctrl + F** or **Windows>Search**.
81. Click and drag the second result **Buffer(Analysis)(Tool)** into the blank workspace in the **Model window**.



Graphic **Buffer** (Analysis) (Tool)
Creates **buffer** polygons around input fe...
toolboxes\system toolboxes\analysis tool...

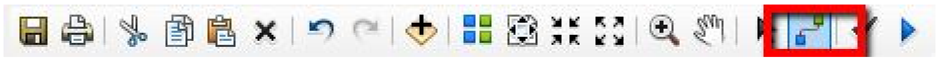
Buffer (Analysis) (Tool)
Creates **buffer** polygons around input fe...
toolboxes\system toolboxes\analysis tool...

Buffer (Coverage) (Tool)
Creates **buffer** polygons around specifie...
toolboxes\system toolboxes\coverage to...

Multiple Ring **Buffer** (Analysis) (Tool)
Creates multiple **buffers** at specified dist...
toolboxes\system toolboxes\analysis tool...

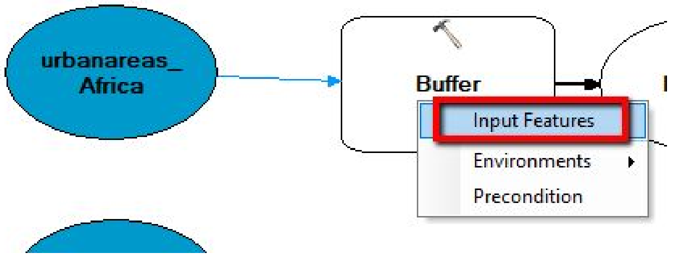
Buffer 3D (3D Analyst) (Tool)
Creates a 3-dimensional **buffer** around ...
toolboxes\system toolboxes\3d analyst t...

82. From the **Model window toolbar**, click the **third button** on the right: **Connect**.

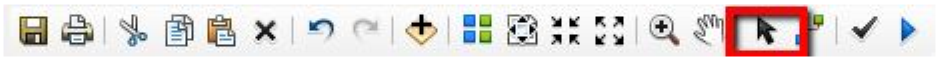


83. This turns the cursor into a wand.
84. Click once on **urbanareas_Africa** and then click on **Buffer** to connect the two.

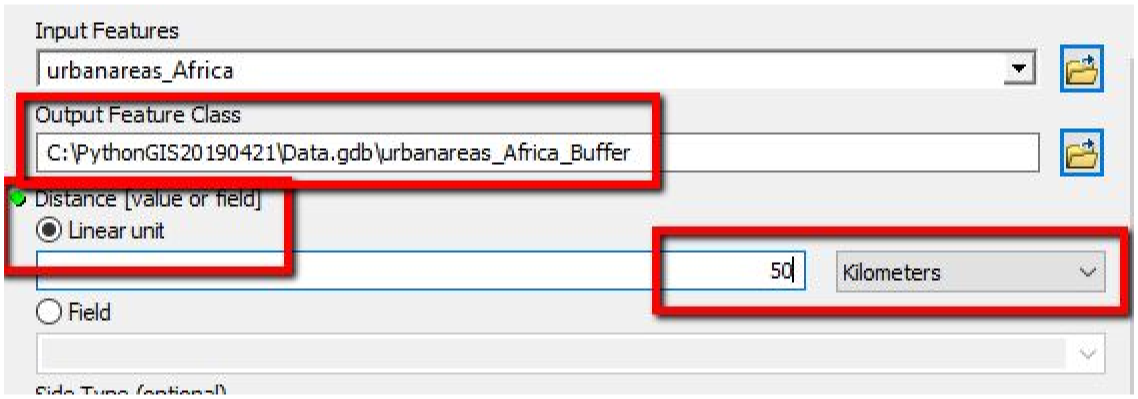
85. When **Buffer** is clicked, choose **Input Features**.



86. Click the **black cursor** next to **Connect** in the Model toolbar.

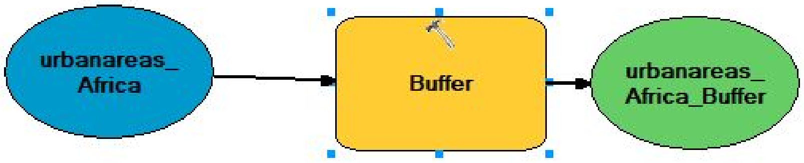


87. Double click **Buffer** and set the parameters--revist **Step 57** if needed.

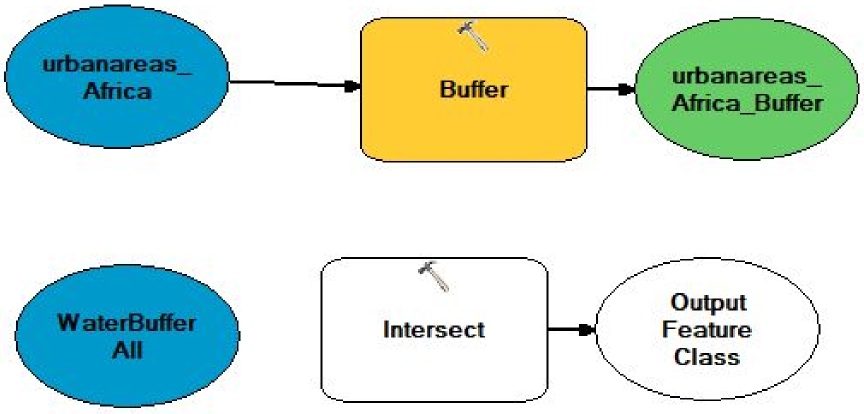


Input Features
urbanareas_Africa
Output Feature Class
C:\PythonGIS20190421\Data.gdb\urbanareas_Africa_Buffer
Distance [value or field]
Linear unit
50 Kilometers
Field

88. Click **OK** when finished. The Buffer model is now filled in.

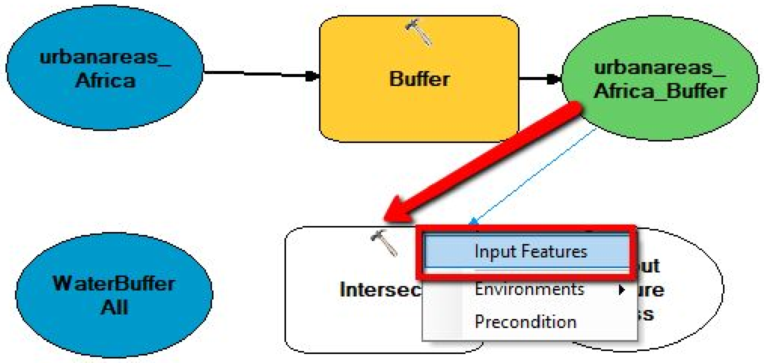


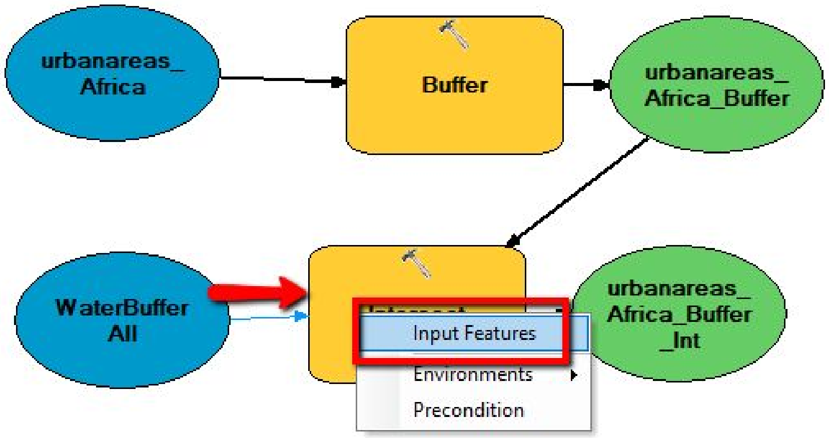
89. From the **Search** panel, type **Intersect**.
90. Click and drag the first result **Intersect(Analysis)(Tool)** in to the **Model window workspace**, too



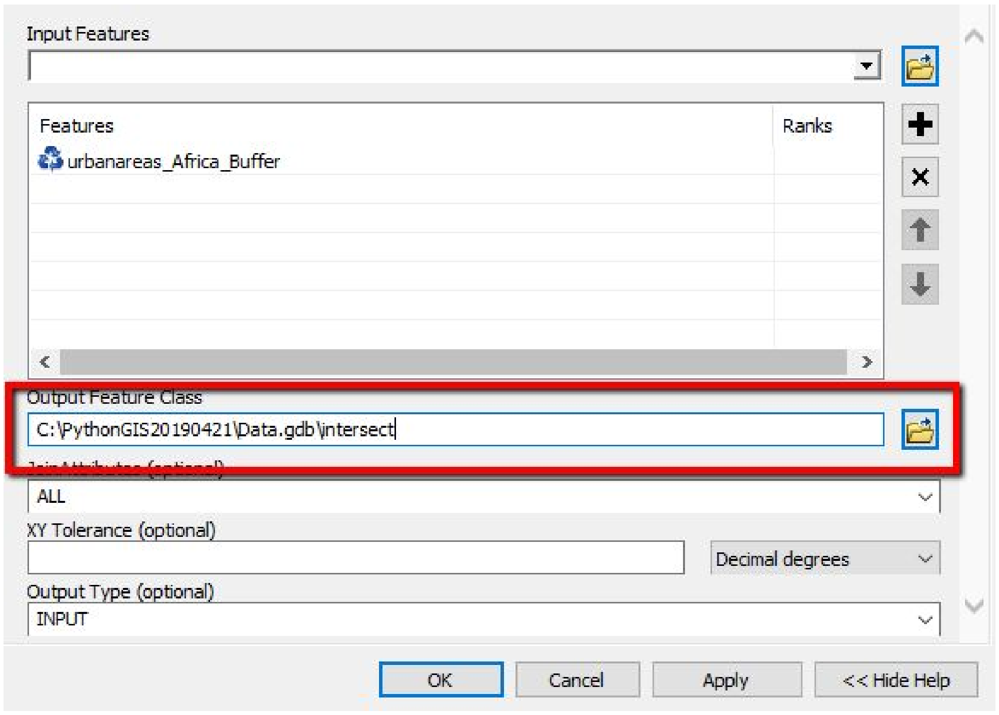
Intersect requires **two** inputs instead of one like with the **Buffer**.

91. Click the **Connector** to and make a connection line between **urbanareas_Africa_Buffer** AND **Intersect** and **Lakes** AND **Intersect.**





92. Click the **black cursor** next to **Connect** in the Model toolbar.
93. Double click **Intersect** to set the parameters, namely the output feature class. That is, make sure it is going somewhere you can find it.


Input Features

Features	Ranks
urbanareas_Africa_Buffer	

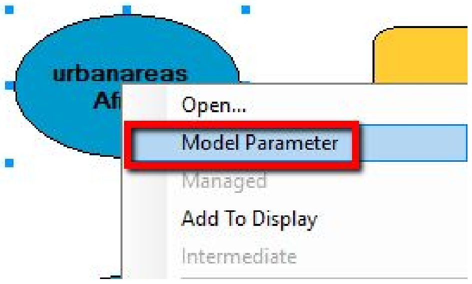
Output Feature Class: C:\PythonGIS20190421\Data.gdb\intersect

JoinAttributes (optional): ALL

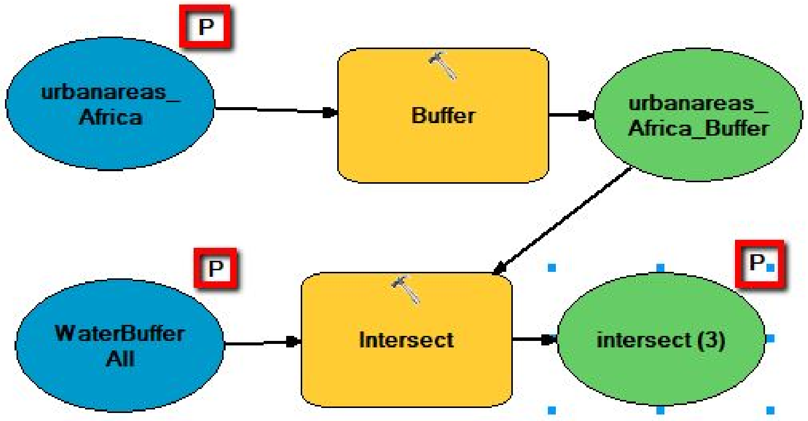
XY Tolerance (optional): Decimal degrees

Output Type (optional): INPUT

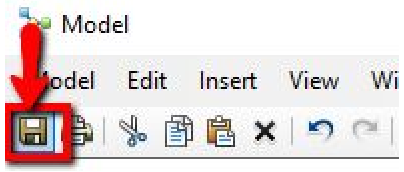
94. Click **OK**.
95. Right click **urbanareas_Africa** and choose **ModelParameter**.



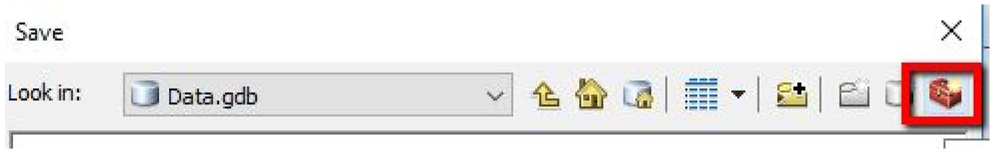
96. Do this for **Lakes** and the **Intersect output** (what comes after the yellow Intersect box), too.



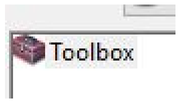
97. Click **Save** in the Model window toolbar.



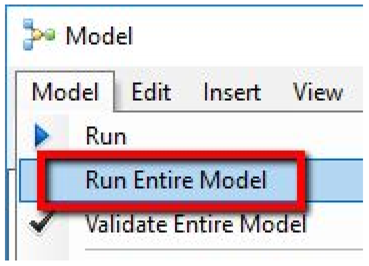
98. Click the **Toolbox** icon in the far right of the Save window.



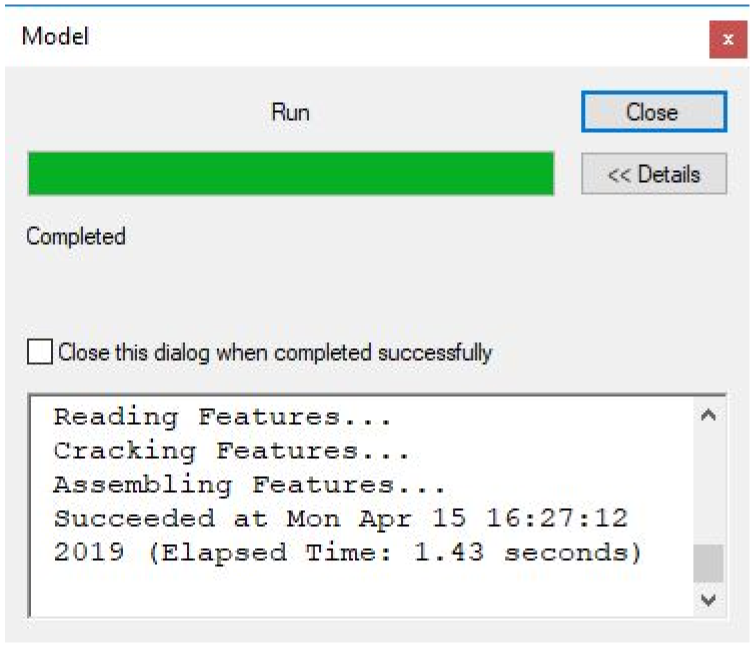
99. Double click the Toolbox.



100. Click **Save**--change the Name, if you prefer.
101. In the Model window menu click **Model > Run Entire Model**.

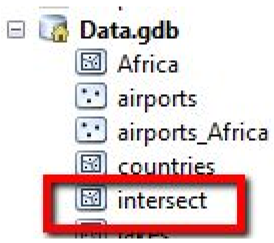


102. A dialog box appears to display the process. When it shows **Completed**, click **Close**.

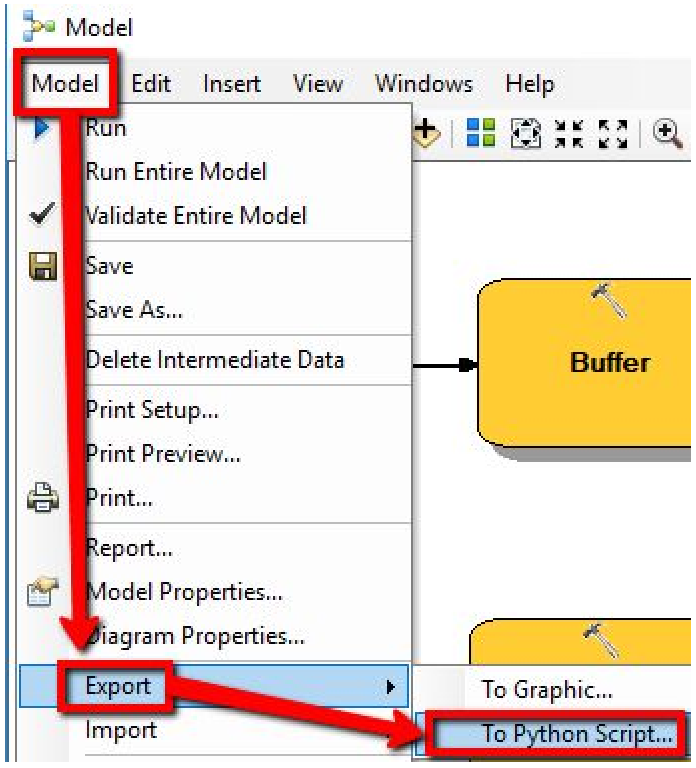


```
Reading Features...
Cracking Features...
Assembling Features...
Succeeded at Mon Apr 15 16:27:12
2019 (Elapsed Time: 1.43 seconds)
```

103. Once the model runs, it should appear in the **Table of Contents**. You can also **refresh** the **Data.gdb** in the **Catalog** to check for the output file (**intersect**).



104. From the Model window toolbar, click **Model > Export > To Python Script**….



105. In the **Save As window**, provide a filename and save the model script where it is findable again. Click **Save**.

106. Outside of ArcMap, navigate to the Python script, right click it, and choose **Edit With IDLE**. It is now possible to either edit this model directly for further analysis and/or share this code. You can also create a shareable ArcGIS toolbox with a graphical user interface.

Further reading and resources

Resources

Contact: Taylor Hixson (taylor.hixson@nyu.edu), NYUAD Librarian for Geospatial Data Services
Workshop slides: tiny.cc/yvt74y
Workshop data source: naturalearthdata.com
NYUAD GIS Guide: guides.nyu.edu/NYUADGIS

Last updated by Taylor Hixson, 17 November 2019

Access ArcGIS Online and Esri Training: email taylor.hixson@nyu.edu for an NYU account
Access ArcGIS at NYUAD: find ArcGIS on this list to see where it is installed across campus
Get ArcGIS: Center for Academic Technology (nyuad.academictech@nyu.edu)

- **ArcGIS is only compatible with Windows. Installing a partition on a Mac is not recommended. Using the university's virtual computer lab is not recommended.**
- **Python 2.7 will install with ArcGIS**

Learning

Training

Esri Training Catalog: email taylor.hixson@nyu.edu for an NYU account to access to courses that show **Requires Maintenance**. Once an account is created, you will have access to all courses that are free or requires maintenance--no prior authorization is necessary after the ArcGIS Online account is created. The following courses were influential in the creation of this workshop:

- Python for Everyone (**Free**)
- Python for Geoprocessing (**Requires Maintenance**)
- Python Scripting for Map Automation (**Requires Maintenance**)

Lynda.com GIS From the Ground Up Playlist: login with NYU NetID and password
DataCamp.com: Freemium service for learning Python. Not NYUAD supported, but highly recommended.

Reading

ArcMap: What is ArcPy?: A starting place to learn more about ArcPy and find **code sample**s and **syntax**
ArcMap: Analyze Introduction: More about geoprocessing and spatial analysis
ArcGIS API for Python: Go beyond ArcPy with the ArcGIS API for Python

Find code and help

ArcGIS Code Sharing
GIS Stack Exchange: Find or ask for help with code
GeoNet: The ArcGIS and Esri community