Using Python in ArcGIS

Estimated time: 45 minutes

Copyright © ESRI

All rights reserved.

Course version . VERSION RELEASE DATE NOT SET.

Printed in the United States of America.

The information contained in this document is the exclusive property of ESRI. This work is protected under United States copyright law and other international copyright treaties and conventions. No part of this work may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying and recording, or by any information storage or retrieval system, except as expressly permitted in writing by ESRI. All requests should be sent to Attention: Contracts and Legal Services Manager, ESRI, 380 New York Street, Redlands, CA 92373-8100 USA.

EXPORT NOTICE: Use of these Materials is subject to U.S. export control laws and regulations including the U.S. Department of Commerce Export Administration Regulations (EAR). Diversion of these Materials contrary to U.S. law is prohibited.

The information contained in this document is subject to change without notice.

U. S. GOVERNMENT RESTRICTED/LIMITED RIGHTS

Any software, documentation, and/or data delivered hereunder is subject to the terms of the License Agreement. The commercial license rights in the License Agreement strictly govern Licensee's use, reproduction, or disclosure of the software, data, and documentation. In no event shall the U.S. Government acquire greater than RESTRICTED/LIMITED RIGHTS. At a minimum, use, duplication, or disclosure by the U.S. Government is subject to restrictions as set forth in FAR §52.227-14 Alternates I, II, and III (DEC 2007); FAR §52.227-19(b) (DEC 2007) and/or FAR §12.211/12.212 (Commercial Technical Data/Computer Software); and DFARS §252.227-7015 (NOV 1995) (Technical Data) and/or DFARS §227.7202 (Computer Software), as applicable. Contractor/Manufacturer is ESRI, 380 New York Street, Redlands, CA 92373-8100, USA.

@esri.com, 3D Analyst, ACORN, Address Coder, ADF, AML, ArcAtlas, ArcCAD, ArcCatalog, ArcCOGO, ArcData, ArcDoc, ArcEdit, ArcEditor, ArcEurope, ArcExplorer, ArcExpress, ArcGIS, ArcGlobe, ArcGrid, ArcIMS, ARC/INFO, ArcInfo, ArcInfo Librarian, ArcLessons, ArcLocation, ArcLogistics, ArcMap, ArcNetwork, *ArcNews*, ArcObjects, ArcOpen, ArcPad, ArcPlot, ArcPress, ArcReader, ArcScan, ArcScene, ArcSchool, ArcScripts, ArcSDE, ArcSdl, ArcSketch, ArcStorm, ArcSurvey, ArcTIN, ArcToolbox, ArcTools, ArcUSA, *ArcUser*, ArcView, ArcVoyager, ArcWatch, ArcWeb, ArcWorld, ArcXML, Atlas GIS, AtlasWare, Avenue, BAO, Business Analyst, Business Analyst Online, BusinessMAP, CommunityInfo, Database Integrator, DBI Kit, EDN, ESRI, ESRI—Team GIS, ESRI—The GIS Company, ESRI—The GIS People, ESRI—The GIS Software Leader, FormEdit, GeoCollector, Geographic Design System, Geography Matters, Geography Network, GIS by ESRI, GIS Day, GIS for Everyone, GISData Server, JTX, MapIt, Maplex, MapObjects, MapStudio, ModelBuilder, MOLE, MPS—Atlas, PLTS, Rent-a-Tech, SDE, SML, Sourcebook·America, Spatial Database Engine, StreetMap, Tapestry, the ARC/INFO logo, the ArcGIS logo, the ArcGIS Explorer logo, the ArcPad logo, the ESRI globe logo, the ESRI Press logo, the GIS Day logo, the MapIt logo, The Geographic Advantage, The Geographic Approach, The World's Leading Desktop GIS, *Water Writes*, www.esri.com, www.geographynetwork.com, www.gis.com, www.gisday.com, and Your Personal Geographic Information System are trademarks, registered trademarks, or service marks of ESRI in the United States, the European Community, or certain other jurisdictions.

Other companies and products mentioned herein may be trademarks or registered trademarks of their respective trademark owners.

Use the Python Scripting Environment

Estimated time: 45 minutes

All scripting languages have a standard set of functionality. For example, every scripting language has the ability to comment code, to execute loops, to concatenate strings, to run decision making statements, and to execute a set of built-in functions. The main differences between scripting languages are the syntax and the type of builtin functionality that is available.

In the first part of this exercise, you will become familiar with the Python scripting language. You will learn about comments, variables, built-in functions, modules, concatenation, decision making, and loops.

In the second part of this exercise, you will work with ESRI's Geoprocessing object. This object gives script language programmers access to the power of GIS. For example the Geoprocessing object opens up all the tools in ArcToolbox and also gives programmers an easy way to get to GIS data for edits, query, manipulation, analysis, and more.

Step 1: Create a script

In this step, you will create a new Python script and add some comments to it.

☐ Start PythonWin.

PythonWin is one of the integrated development environments (IDE) that you can use for writing Python scripts. When you open PythonWin, it opens with the Interactive Window. The Interactive Window has many purposes: (1) it provides a quick way to execute and test individual lines of code without saving that code, (2) it outputs error messages from scripts, (3) and it shows the output from print statements.

☐ Click File > New.

☐ In the New dialog, verify that Python Script is selected.

☐ Click OK.

Currently, you have two windows in your application: the Interactive Window and the Script1 window. You can write lines of code in either window. However, if you write lines of code in the Interactive Window, the code is not saved. For this reason, most of your code will be written in a script window.

Copyright © ESRI

☐ Resize the Script1 and Interactive windows so that you can see both at the same time and that neither one covers up the other.

> ***Note:*** You can click on Window > Tile to do this automatically.

☐ Click File > Save As.

☐ In the Save As dialog, navigate to the C:\EDUC\Python folder, and for the file name type **MyFirstPython.py**.

☐ Click Save.

The first lines of code in any script should be comments. Comments are lines of unexecutable code that are used to document a script. Comments throughout your code that explain individual lines or blocks of related code make your scripts easy for others to read and understand.

In Python, any line of code preceded by one or many pound signs (#) is a comment. Python ignores all commented lines and only executes uncommented lines of code.

☐ In the MyFirstPython.py window, type the following.

```
# Name: <your name>
# Purpose: This is my first script.
```

Python's commented code is green and italicized.

Step 2: Use variables

Programming languages uses variables. Variables hold on to a value while a program or script runs. Throughout the execution of a script, the values in a variable can change. Variables can store many different types of data, including strings, numbers, lists, and more. In this step, you will use variables to store numbers, strings, and lists.

First, you will create a numeric variable.

☐ In the MyFirstPython.py window, type the following code under the comments.

```
a = 5
print a
```

Notice that the word print turns blue and bold. That indicates that you have typed it in correctly. The p in print must be lower case. If you had typed an upper case P for Print, the word would not turn blue or bold and if you ran the code you would get an invalid syntax error message in PythonWin's status bar.

Copyright © ESRI

☐ From the Standard toolbar, click the Run button.

☐ In the Run Script dialog, click OK.

The value stored in a is printed to the Interactive Window. The print statement sends output to the Interactive Window.

Variables can also store the result of math equation.

☐ In the MyFirstPython.py window, type the following code after the print a statement.

```
b = 5 + 8
print b
```

☐ From the Standard toolbar, click the Run button.

☐ In the Run Script dialog, click OK.

The values stored in a and b are printed to the Interactive Window. The value of b holds the number 11, which is the sum of 5 + 8.

Variables can store string values. String values must be enclosed in single or double quotes.

☐ In the MyFirstPython.py window, type the following code after the print b statement.

```
c = "Python is Fun!"
print c
```

☐ Run the script.

Python is Fun is the last line printed to the Interactive Window.

In addition to storing one value inside a variable, you can store multiple values inside a variable. This data type is known as a list. You can extract a single value (or a subset of values) from a list based on the index position.

☐ In the MyFirstPython.py window, type the following code after the print c statement.

```
d = ["Jack", "Clint", "Scott"]
print d[1]
```

☐ Run the script.

Copyright © ESRI

The last line of output in the Interactive Window is Clint. d[0] would print Jack, which is the first value in the list (lists are zero-based). d[2] would print Scott, which is the third value in the list.

Before moving on to functions, you will comment out the existing lines of code.

☐ Highlight all executable lines of code in your script.

☐ In the MyFirstPython.py script window, right-click, point to Source code and click *Comment out region*.

☐ Verify that all lines are commented.

Two pound signs (##) are placed in front of each line of code and the code turns gray. The script's code is now commented out and unexecutable.

Step 3: Use functions

Python has a number of built-in functions that allow you to perform various operations. The round function is just one of many built-in functions.

☐ In the MyFirstPython.py window, type the following code at the end of the script.

```
f = round(3.4)
print f
```

☐ Run the script.

The value of 3.4 is rounded to 3.0. You can access the list of built-in functions using the dir(__builtins__) statement.

☐ In the MyFirstPython.py window, type the following code after the print f statement.

```
print dir(__builtins__)
```

There are two underscores on either side of the word builtins, not one.

☐ Run the script.

☐ In the Interactive Window, scroll to the bottom so that you can see the large list of built in functions.

Can you locate the *round* function in the list?

Copyright © ESRI

There are many built-in functions in Python that can help you while writing any script.

Step 4: Concatenate strings

When writing scripts, there will be times when you want to combine two strings together. You can do this using the plus sign (+).

☐ In the MyFirstPython.py window, type the following code after the print dir(__builtins__) statement.

```
g = "Streets"
h = ".shp"
print g + h
```

☐ Run the script.

The two string values are concatenated together and Streets.shp is printed to the Interactive Window.

Sometimes you might concatenate strings and numbers. To do that you will need a function to convert the number into a string.

☐ Create another variable.

```
m = 4
```

☐ Then add another print statement, but use the *str* function to convert the number 4 into a text string.

```
print g + str(m) + h
```

☐ Run the script.

Streets4.shp prints out in the Interactive Window. If you tried to print g + m + h you would get an error message in the Interactive Window.

☐ Comment all executable lines of code before continuing to the next step.

Step 5: Make decisions

One of the most powerful aspects of scripting is the ability to use decision making statements. Python's syntax for decision making statements is *if-elif-else*.

Copyright © ESRI

☐ In the MyFirstPython.py window comment the previous lines; then, type the following code at the end of the script.

```
x = 5

if x < 5:
     print "The number is less than 5"
elif x > 5:
     print "The number is greater than 5"
else:
     print "The number is 5"
```

> **Make sure you follow the same indentation as above. And remember the colons (:) after the if, elif, and else lines.**

☐ After the last line of code, *print "The number is 5"*, press Enter twice to unindent the code.

☐ Run the script.

Currently, the value of x is equal to 5, therefore the else statement gets executed and *The number is 5* is sent to the Interactive Window.

☐ If you have time, change the value of x to a number greater than 5, then a number less than 5, and rerun the script.

☐ Before continuing on to the next step, comment out the If statement block of code.

Step 6: Run repeatedly

Programming languages have the ability to run a series of statements repeatedly using a syntax called a loop. Python has three types of loops: while loop, list loop, and counted loop. In this step, you will work with one of them.

☐ In the MyFirstPython.py window, type the following code at the end of the script.

```
y = 1

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

> **Make sure you follow the same indentation as above. Indentation is part of the Python languages syntax.**

Copyright © ESRI

☐ After typing in the *y = y + 1* line of code, press Enter twice to unindent the code.

The above four lines of code represent a while loop that will continue to execute as long as its (y < 10) condition is true. A while loop ends when its condition becomes false.

Below is a description of each line of code:

Line 1: y is initially assigned a value of 1.
Line 2: As long as y is less than 10, enter the loop.
Line 3: print the value of y to the Interactive Window.
Line 4: Add the value of 1 to y and go back to the top of the loop (Line2).

The while loop will continue to execute until y is greater than or equal to 10. Once y reaches a value of 10, the loop will terminate.

How many numbers do you expect to be printed to the Interactive Window?

☐ Run the script.

Values 1 to 9 are printed to the Interactive Window. 10 is not included, because the loop only executes if the value of y is less than 10.

☐ Increase the while value from 10 to 1000 and run the script again.

☐ Before continuing on to the next step, comment out the While statement block of code.

In the following steps you will start to interact with some GIS functionality and ArcToolbox.

Step 7: Get Help

In this step, you will create a script that buffers features in a line feature class. First, you will find the buffer tool and its help. The best way to see if a tool exists is to use the ArcToolbox Index and Search tabs.

☐ Start ArcMap with a new empty map, or open an existing .mxd document

☐ Click the Add Data button.

☐ Navigate to the c:\EDUC\Python\SanDiego.gdb geodatabase and add all its feature classes to the map. There should be two of them.

☐ Click the Search tab located on the right side of the ArcMap window.

Copyright © ESRI

☐ In the *Search for* area, type **Buffer**.

☐ Click the Search button.

The search area lists the Buffer (Analysis) and Multiple Ring Buffer (Analysis) tools.

☐ In the search area, click on the Buffer tool to open the tools dialog.

☐ In the Buffer window, if necessary click the Show Help button, and click Tool Help.

The ArcGIS Desktop Help window opens at the Buffer tool help page.

☐ Scroll down to the **Scripting syntax**.

☐ Examine the seven parameters and their descriptions. The first three, input feature class, output feature class, and buffer distance are the most important for this exercise.

☐ Scroll down to the bottom of the help window to the Code Sample to see the Script Example.

```
import arcpy
arcpy.env.workspace = "c:/data"
# Buffer roads.shp based on each road feature's value in the Distance field,
#  and dissolve buffers into groups according to values from Road_Type field.
arcpy.buffer_analysis("roads", "buffered_roads", "Distance", "FULL", "ROUND", "LIST", "Road_Type
```

Technically this script could be copied and pasted into a Python script window. However in order for it to work, you would need either the supporting data mentioned inside the sample or you would need to change the sample's lines of code to replace the sample's data with your data.

Now that you know the syntax for the Buffer tool and you have seen some sample code, you wil create a new buffer script in the next step.

☐ Leave ArcMap and the Help window open.

While you are writing code in the next step you might copy and paste some of the sample code. After your script runs you will verify that the Buffer tool did indeed create a new feature class. Then you will add the new feature class to ArcMap.

☐ Return to PythonWin.

Copyright © ESRI

Step 8: Write a buffer script

In this step, you will write a Python script to buffer a feature class from a personal geodatabase. You will create a new script, import the COM module, create the Geoprocessor, and set the workspace to C:\EDUC\Python.

☐ In PythonWin, click the File Menu and click New.

☐ In the New dialog, make sure Python Script is selected, and click OK.

☐ Click the File menu and click Save As.

☐ Save the script to C:\EDUC\Python\Buffer.py.

☐ Insert comments for the author name and purpose of the script.

```
# Name:
# Purpose:
```

Next you will write the two lines of code to let Python know that you are going to use an object from the ESRI Geoprocessing library. You will import the arcpy.

Python scripts that you write for ArcGIS desktop will most likely always contain these two lines of code.

☐ Import the arcpy module by typing in the following line of code.

```
import arcpy
```

> ***Note:*** Instead of typing this and the next line of code you can copy and paste from the help sample.

☐ Add the following lines of code to set the current Workspace to the SanDiego personal geodatabase.

```
arcpy.env.overwriteOutput = 1
arcpy.env.workspace = r"c:\EDUC\Python\SanDiego.gdb"
```

> ***Note:*** In Python, the "r" is necessary before a string where you are using a single backslash, such as a directory path.

☐ Add the following two lines of code to buffer the roads feature class by 2000 feet to create the RoadBuff feature class.

```
arcpy.Buffer_analysis("Freeways", "FreewayBuf", 2000)
```

Copyright © ESRI

☐ Add one more line of code to indicate that the script has finished running.

```
print "The Freeway buffer has been created."
```

You are now ready to test this script.

☐ Click the Run button.

☐ Click OK, to run the script.

☐ When the print statement prints in the Interactive Window the script has finished.

Next you will verify that your script has create a buffer zone around the major freeways of San Diego.

☐ In ArcMap, click the Add Data button.

☐ Navigate to the c:\EDUC\Python\SanDiego.gdb personal geodatabase. Notice that there is a new feature class in there called FreewayBuf. Add the FreewayBuf feature class to the map.

☐ Notice the buffer zones around each freeway.

In the next steps you will turn this script into an ArcToolbox tool.

Step 9: Turn a script into a tool

In this step you will turn your Buffer script into an ArcToolbox tool.

☐ In PythonWin and in the Buffer.py script locate the following line of code.

```
arcpy.env.workspace = r"c:\EDUC\Python\SanDiego.gdb"
```

☐ After this line of code, add the following two lines to create variables that will store user input values. When a user runs your tool, they will enter these values inside your tool's dialog box. The two variables will hold the values for use later on in the script.

```
output = arcpy.gp.GetParameterAsText(0)
distance = arcpy.gp.GetParameterAsText(1)
```

☐ Now locate the Buffer line of code.

You are going to change its second and third parameters.

Copyright © ESRI

☐ Replace the current values for output feature class and buffer distance with the two variables you added above. When finished the line of code should look like the one below.

```
arcpy.Buffer_analysis("freeways", output, distance)
```

Next you will change the print statement.

When someone runs your Freeway Buffer tool they will be in either ArcMap or ArcCatalog. Your current print message is meaningless in those two application, because it is designed to print messages in PythonWin's Interactive Window. You will alter the message line of code so that it can print in ArcMap or ArcCatalog inside your tool's Progress dialog box.

☐ Replace *print* with: `arcpy.gp.addwarning`

☐ Replace the print text with the longer string, shown below, that concatenates the original text message with the name of the output feature class. The final line of code should look like the one below.

```
arcpy.gp.addwarning ("The Freeway buffer " + output + " has been created." )
```

☐ Click the File menu and click Save As... with the name 'Buffer_Params' to save this script.

```
import arcpy
arcpy.gp.overwriteoutput = 1
arcpy.env.workspace = r"c:\EDUC\python\sandiego.mdb"

output = arcpy.gp.GetParameterAsText(0)
distance = arcpy.gp.GetParameterAsText(1)
arcpy.Buffer_analysis("freeways", output, distance)

arcpy.gp.addwarning ("The Freeway Buffer " + output + " has been created." )
```

☐ Now go to ArcMap and make sure ArcToolbox is open.

☐ In a blank area of ArcToolbox (not on any existing toolbox) right-click and click New Toolbox.

☐ Rename the new toolbox to **MySanDiegoTools**.

☐ Right-click the My San Diego Tools toolbox, and click Add > Script.

☐ In the Add Script dialog, for Name, type **FreewayBuffer**.

> ***Note:*** Be sure not to leave a space in the name.

Copyright © ESRI

☐ For Label, type **Create Freeway Buffer Zones**.

☐ For Description, type **Creates buffer zones around San Diego Freeways**.

☐ Click Next.

☐ For Script File, browse to and add the *C:\EDUC\Python\Buffer_Params.py* script.

☐ Click Next.

Next, you will specify the two script parameters, output and distance, and then set their properties.

☐ In the Add Script dialog box, click the first box under Display Name, in the Display Name column. This opens that box up for data entry.

☐ For Display Name, type **Output feature class name**.

☐ Change the Data Type to **String**.

☐ Click the box below Output feature class name and type **Buffer distance**.

☐ Change the Data Type to **Long**.

☐ Click *Finish* to close the *Add Script* dialog.

Next you will run the tool.

☐ In ArcToolbox go to the My San Diego Tools toolbox and double-click the Create Freeway Buffer Zone tool.

☐ When the tool's dialog box opens, click Show Help.

☐ Notice the help information matches the text that you entered earlier when creating the tool.

☐ For the Output feature class name, type **Buffer1000**.

☐ For the Buffer distance, type **1000.**

☐ Click OK to run the tool.

Copyright © ESRI

While the tool is running pay close attention to the messages in the Progress dialog. The green message is yours, from when you used AddWarning. Warning messages are green, an AddMessage would be black, and an AddError would be red.

☐ Click Close on the Progress dialog.

☐ Click the Add Data button and add the newly created buffer zones, Buffer1000, to ArcMap.

Conclusion

In this exercise you learned the basics of Python, how to write a script to run ArcGIS Desktop tools like Buffer, and then how to turn a script into custom tool for ArcToolbox.

Copyright © ESRI