

Python Basics

Tutorial Overview

Python has become the scripting language for geoprocessing in ArcGIS with the ArcPy site package providing access to the tools, functions, classes, and modules needed to develop and run geoprocessing workflows through scripts. By becoming proficient in the use of Python and specifically ArcPy, you can automate the execution of repetitive tasks as well as customize reusable applications. In this tutorial, you will learn basic concepts about the Python language, including code structure and syntax. **This is not intended as a comprehensive introduction to Python or to programming in general.** You are encouraged to complete general Python tutorials to gain a greater understanding of the language. Please see the References and Readings section for a list of recommended tutorials.

Skills

By completing this tutorial, you will become comfortable with the following skills:

- Using Python script editors, specifically IDLE and Spyder
- Creating and using strings, numbers, and lists in Python
- Building control structures using branching (if-then-else) and looping (for) in Python
- Importing modules and calling functions in Python.

Time Required

The following classroom time is required to complete this tutorial:

• 45 - 60 minutes

Materials Required

Technology:

- Python (included as part of ArcGIS installation)
- Spyder (install from the ArcGIS Pro Python package manager or from <u>https://github.com/spyder-ide/spyder</u>) or another development environment

Data:

- This tutorial does not require any data.
- Sample scripts can be downloaded at http://edu.maps.arcgis.com/home/item.html?id=988d2ab3138b40398f7bd65306ce9da7.

Production Date

The Education and Research Group at Esri Canada makes every effort to present accurate and reliable information. The Web sites and URLs used in this tutorial are from sources that were current at the time of production, but are subject to change without notice to Esri Canada.

• Production Date: February 2014. Updated October 2017.



Background Information

The purpose of the ArcPy package is to help users perform geographic data analysis, data conversion, data management, and map automation with Python. The package consists of hundreds of functions, classes, tools, and modules. Many examples can be found of how to use ArcPy, but it can be difficult to understand how the examples can be adapted to your needs if you do not have at least a basic understanding of programming in Python.

In many programming languages, line breaks and indentations are non-essential and are used solely to improve readability of the code. Python is different in that it uses indentations, instead of curly brackets {}, to determine the relationship between lines of code. For example, a simple *if* statement in Python would follow this format:

```
if condition:
do this
and that
```

whereas in JavaScript it could follow this format:

```
if(condition){do this;and that;}
```

In some ways, this makes Python more intuitive and easier to learn than other languages. However, it also becomes easier to make syntax errors or misassociate instructions with control structures.

Another difference with Python is that objects are created dynamically. This means that to create an object, all you have to do is assign the value of the object to a variable. In programming, an object is an individual item, such as a number, a string of text, or a collection of values, while a variable is a symbolic name used to store information. In Python, the variable type (integer, float, string, etc.) is determined by the value that it was assigned. This is in contrast to other programming languages where the variable that will store the object must first be declared as being a certain type before it can be assigned a value. Again, while this makes it easier to program in Python, typing one wrong character when using a variable could cause a script to fail, or certainly not behave as expected.

Finally, Python is an interpreted rather than compiled language. This means that the code is read and interpreted line by line each time it is run. In contrast, with languages such as C and Java, the entire program is read and compiled into an executable file that is ready to run as a whole. Compiled programs have been converted into a format, often binary code, that cannot be reviewed line by line by a human unless they are decompiled, whereas programs written using interpreted code can always be read and modified by a human in a text editor. Because the code is interpreted each time, executing a Python script can be slower than running a compiled executable program, but it is much easier to make and test changes to the script.

The best way to learn any programming language is to practice. After completing this tutorial, you should be familiar enough with the basic concepts of Python to write your own simple scripts and start working with the ArcPy package.

References and Reading

- Python for ArcGIS <u>http://desktop.arcgis.com/en/analytics/python/</u>
- What is ArcPy? http://pro.arcgis.com/en/pro-app/arcpy/get-started/what-is-arcpy-.htm
- The Python Tutorial <u>http://docs.python.org/2/tutorial/</u>
- Codecademy Python Track http://www.codecademy.com/tracks/python
- Python Scripting for ArcGIS, Paul A. Zandbergen (Esri Press)



Part A: Python Script Editors

The Python Window in ArcGIS Desktop applications is useful for executing single lines of codes or testing code snippets. However, to develop stand-alone Python scripts, you should use a separate development application. There are a large number of applications available, ranging from freely downloadable generic code editors, such as Notepad++ or Sublime Text, to Python-specific editors or integrated development environments (IDEs), such as Spyder and PyCharm, to comprehensive IDEs, such as Aptana.

This section gives you a brief overview of two Python script editors, namely IDLE and Spyder. You may choose to use any application that allows you to create, debug, and run Python scripts, but keep in mind that there will be differences between the application you are using and the steps and screenshots in this tutorial.

IDLE

IDLE comes with every installation of Python, including the one installed with ArcGIS Desktop.

1. To start IDLE, from the Start Menu > All Programs > ArcGIS > Python 2.7 > IDLE (Python GUI).

A window should open similar to the one below.

Python 2.7.12 Shell		8
File Edit Shell Debug Options Window Help		
<pre>Python 2.7.12 (v2.7.12:d33e0cf91556, Jun 27 2016, 15 v.1500 32 bit (Intel)] on win32 Type "copyright", "credits" or "license()" for more >>></pre>		
	Ln: 3	Col: 4

This is the Python shell, which is a Python interpreter. You can enter any Python command at the prompt >>> and press Enter to run it.

2. Type the following line at the prompt and press Enter.

```
print "Hello World"
```

The print command will be discussed in more detail later. For now, all you need to be aware of is how IDLE handles input and output. Notice that the text is colour coded. Python keywords, like print, are orange while Python strings are green. Standard output, what a script prints out to the shell, is blue. You can see and change these and other highlighting settings in the Highlighting tab of the IDLE Preferences window (Options > Configure IDLE).

```
v.1500 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> print "Hello World"
Hello World
>>>
Ln:5 Col:4
```

3. Select File > New File.

This opens a new window where you can create a script, save it, then run it.

```
highered@esri.ca
```



4. Type in the same line of code and then save your script as **helloworld.py**.

You can save the script anywhere – as long as you remember where! It is a good idea to create a specific folder for your scripts so you can find them easily when you need them.

helloworld.py - C:/PythonTutorial/helloworld.py (2.7.12)	
File Edit Format Run Options Window Help	
print "Hello World"	A
	v
	Ln:1 Col:0

5. Select **Run > Check Module** to check the syntax of your script.

If a syntax error is found, you will see this window pop up, for example:

C Syntax	error	X
8	There's an error in you unexpected indent	ur program:
		ОК

The error will then be highlighted in your script.

helloworld.py - C:/PythonTutorial/helloworld.py (2.7.12)	
File Edit Format Run Options Window Help	
print "Hello World"	▲ ▼ n:2_Col:0
	Ln; Z COI; U

The syntax check will stop as soon as an error is encountered. It can only find one syntax error at a time. After correcting the error, save the script and check it again to see if there are any more syntax errors.

6. Select **Run > Run Module** or press **F5** to execute the script.

You should see the message output to the Python Shell.

Bython 2.7.12 Shell	
File Edit Shell Debug Options Window Help	
>>> print "Hello World" Hello World	_
>>> ====== RESTART: C:/PythonTutorial/helloworld.py	
Hello World	
>>>	-
	Ln: 8 Col: 4

7. In the Python Shell window, select **Debug > Debugger**.



A debugger allows you to step through the script to check that each line of code is doing what it is intended to do.

 Run the helloworld script again (i.e., with the script window active, select Run > Run Module or press F5).

The contents of the Debug Control window may not be entirely clear at first, but as you gain experience with Python and debugging, you should come to understand what the debugger is doing. The window is divided into five sections, shown below: debug controls, display controls, current status, stack, and objects. The debug controls determine how the script is stepped through while the display controls toggle other parts of the window on and off.

	👌 Debug Contro		8	
Debug controls	Go Step Over	Out Quit 🔽 Stack 🗆 Stack 🖉 Stack 🖉 Stack 🖉 Stack	ource lobals	Display controls
Current status	helloworld.py:1: <	<module>()</module>		
Stack		00: exec cmd in globals, locals dule>(), line 1: print "Hello Wo	orld"	
		Locals		
	builtins <mo< td=""><td>odule 'builtin_' (built-in)></td><td>A</td><td></td></mo<>	odule 'builtin_' (built-in)>	A	
	doc Non	e		
	file 'C:/P	PythonTutorial/helloworld.py		
Local/Global		nain_'		
objects	package Non	e	-	

9. Click **Step** to execute the current line of code.

The Python shell output should be the same as above, except that tags are added indicating the debugger is on. Note that it may take several "Steps" to execute the code fully.

<pre>File Edit Shell Debug Options Window Help [DEBUG ON] >>> =================================</pre>	Python 2.7.12 Shell	
>>> ==================================	File Edit Shell Debug Options Window Help	
[DEBUG ON]	>>>	world.py =====
Ln: 34 Col: 4	[DEBUG ON]	

10. Exit IDLE.



Spyder

Spyder stands for **S**cientific **P**Ython **D**evelopment **E**nvi**R**onment. It is a free, open source and crossplatform Python development environment. It can be downloaded and installed from the Python Package Manager in ArcGIS Pro 1.3 or higher or by following the installation instructions at https://pythonhosted.org/spyder/.

To install using ArcGIS Pro:

- 1. Start ArcGIS Pro. If necessary, type your user name and password on the ArcGIS Sign In screen and click **Sign in**.
- 2. Click About ArcGIS Pro at the bottom left then select Python from the side menu.
- 3. Choose Add Packages. Seach for spyder or scroll through the list. Select the package and click Install.

Using Spyder

1. Start Spyder.

A window should open similar to the one below. Allow access through domain and private networks, if prompted, and click OK to close any message or warning windows.

 Spyder (Python 3.5) 			
File Edit Search Source Run Debug Consoles Projects	Tools View Help		
🗅 🔈 🛱 🐜 🗮 @ 🕨 🗗 🔂 🌾) 🕻 🚝 🚝 💓 🔳 💽 🔀 🎤 🔶 🔶 »		
Editor 🗗 🖓	Help & X		
temp.py* 🛛	🖞 Source Console 🔻 Object 🗸 🗣 🎎		
1# -*- coding: utf-8 -*- 2""" 3 Spyder Editor 4	Usage		
5 This is a temporary script file. 6 """ 7	Here you can get help of any object by pressing		
	Variable explorer File explorer Help		
	Internal console 🗗 🗙		
	Spyder Internal Console		
	This console is used to report application internal errors and to inspect Spyder internals with the following commands: spy.app, spy.window, dir(spy)		
	Internal console Python console History log IPython console		
Permissions: RW End-of-lines: CRLF E	ncoding: UTF-8 Line: 7 Column: 1 Memory: 41 %		

Note: Because of differences in systems and specifically any previous Python installations, you may find that Spyder isn't behaving as expected. If you have installed Spyder through ArcGIS Pro, you can try launching it through **Start Menu > All Programs > ArcGIS > ArcGIS Pro > Python Command Prompt**. Type **spyder** at the prompt and press Enter. You may also find helpful tips on GeoNet https://geonet.esri.com/community/developers/gis-developers/python.

2. Choose the **IPython** console tab at the bottom right. Type in the same line at the prompt as you did in the IDLE Python shell and press Enter.

print "Hello World"



You should see a syntax error message:

```
In [1]: print "Hello World"
File "<ipython-input-1-085e5f98785e>", line 1
print "Hello World"
^
```

SyntaxError: Missing parentheses in call to 'print'

The version of Python installed with ArcGIS Desktop, i.e., the version used in IDLE and ArcMap, is 2.7 but the version installed with ArcGIS Pro is 3.5. Among the differences between 2.x and 3.x is that in 3.x, print is a function, rather than a statement, and requires brackets (). When you are developing a script, you should ensure that your development environment and target environment are running compatible versions of Python.

3. Place the cursor before or within any instance of the word "print" in the console and press **Ctrl-I** to see documentation for the print function in the Help pane.



4. Place the cursor at the console input prompt and press the up arrow to repeat the previous script line. Add brackets () around "Hello World" and press Enter.

```
In [2]: print ("Hello World")
Hello World
```

- 5. Click seside **temp.py** in the Editor pane to close the temporary script file. This will also open a new script file.
- 6. Type the same line of code as in Step 5, i.e., omit the brackets, in the Editor after the header.

Spyder automatically checks script syntax in the editor as you type each line. If a syntax error is found, a warning icon will be displayed beside the line with the error.

7. Hover over the icon to see the warning message.

0	8	print "Hello World	"
		Code analysis	
		invalid syntax	

As in IDLE, syntax errors are found one at a time. You must clear an error before subsequent lines will be checked. You can use Ctrl-I to get help documentation for functions.

4. Correct the syntax for the print function by adding brackets (). Save your script as

helloworld_spyder.py and then click b or press F5 to execute the script.



Spyder automatically saves your script when you run it so you only need remember to save when you quit if you have made changes without running the script – but it is always prudent to save your work regularly.

The run command will be displayed in the IPython console followed by any output from the script, if it was run without errors. An error message will be displayed if the script failed to run.

```
In [3]: runfile('C:/PythonTutorial/helloworld_spyder.py', wdir='C:/
PythonTutorial')
Hello World
```



The debug command will be displayed in the console. An arrow indicates the current position of the debugger. Instead of an input prompt In [n], you will see the debug prompt ipdb>.

ipdb>

6. Click **Step** ^C to skip over the comment line then **Step Return** ^C to execute the print function.

```
ipdb> Hello World
--Return--
None
```

7. Continue clicking **Step Return** until the debugger exits. Close any script files that were opened by the debugger.

```
ipdb> --Return--
None
> <string>(1)<module>()
ipdb>
ipdb>
```

In [5]:

The additional steps are part of the debugger, not your script. If you do not want to step through these steps, or to exit the debugger at any time, you can click **Stop**.



Part B: Python Data Types and Variables

"Hello World" is an example of a string. String values must be enclosed in single ' ' or double " " quotation marks so that they are not interpreted as variables. In Python, strings are immutable sequences of one or more characters. Immutable means they cannot be changed. You can access individual characters within the string but you cannot modify them. Other immutable data types are numbers and tuples. Tuples are lists with immutable elements in a specific order.

Although sometimes you will use values directly, as you did in Part A, if you will be using the same value multiple times in your script, it is usually more convenient to assign it to a variable and then apply operations and functions to the variable. You may use Spyder, or any IDE with an interactive Python interpreter that is running Python 3.x to complete the remainder of this tutorial. The console prompts may be different but output and error messages should be the same.

1. Type the following into the IPython Console and press Enter.

```
In: print ("Hello World"[0])
```

You should see the letter H printed out. Square brackets are used to select a single element or a set of elements from a sequence. The numbers inside the brackets are index values. The first character in a string or item in a list have an index value of 0.

Note that you do not need to use the print keyword in the interactive window. Simply typing "Hello World" [0] will give the same result, except that the H will be enclosed in single quotation marks to indicate that it is a string output. You will need to remember to include the print keyword in any scripts if you want to have the value output when the script runs.

```
In [5]: print ("Hello World"[0])
H
In [6]: "Hello World"[0]
Out[6]: 'H'
```

2. Assign the string to a variable then print out a range of elements.

```
In: mystring = "Hello World"
In: print (mystring[3:7])
```

With the exception of a few reserved keywords, you can name your variables anything you want. Nevertheless, you should follow a few best practice guidelines for variables: start with a character; use all lowercase; use descriptive names; and keep names short.

The colon within square brackets indicates a range, from the element at the first index value up to but *not including* the second index value. Try specifying different ranges. You can omit the first value, e.g., [:5] or the second value, e.g., [6:].

3. Try assigning a new value to one of the elements.

```
In: mystring[5] = "-"
```

You should see an error similar to the following:

TypeError: 'str' object does not support item assignment

This is because, as noted above, strings are immutable. To change the value of a single element in an immutable data type, the value of the entire variable must be changed.

4. Assign a new value to one element of mystring, using ranges to keep the remaining elements.



In: mystring = mystring[:5] + "-" + mystring[6:]

The expression to the right of the equal sign in a line of script such as the one above is evaluated and then assigned to the variable on the left. To see the current value of a variable, you can output it to the console or view it in the variable explorer pane.

Variable explorer 🛛 🗗 🗙				
초 🖺 🏷 🔹 🔍				
Name	Туре	Size	Value	
mystring	str	1	Hello-World	
Variable explorer File explorer Help				

The simplest way to combine strings in Python is to "add" them together. This is called concatenation. The plus sign + is used both for numerical addition and string concatenation, but numbers and strings cannot be added together directly. A number must first be cast as a string before it can be combined with other strings.

5. Cast a number as a string and add it to mystring.

```
In: mystring = mystring + str(2.0)
```

The function str() is a built-in function in Python. It is passed a number, or any other data type, as a parameter and returns a string. In the line above, it allows the number 2.0 to be concatenated with mystring, which should now have the value Hello-World2.0. The functions int() and float() do the opposite, taking a string and returning a number type. The full list of built-in functions can be found at https://docs.python.org/3.5/library/functions.html or through the help features in Spyder. Another function you may find useful is len(), which returns the number or elements, or length, of a string or other sequence.

6. Get information about the len function and then check the length of the new string.

```
In: help(len)
In: len(mystring)
In [12]: help(len)
Help on built-in function len in module builtins:
len(obj, /)
    Return the number of items in a container.
In [13]: len(mystring)
Out[13]: 14
```

The output of len is an integer. Numbers in Python can be either integers or floats (floating-point or decimal numbers). The number type of the result of any mathematical operation will depend on the number type of the inputs: if all inputs are integers, the result will be an integer; if at least one is a float, the result will be a float.

7. Try performing basic mathematical operations using a mix of integer and float numbers.

```
Multiplication, e.g., 6 * 7
Integer division, e.g., 18/4
Modulus (remainder), e.g., 26%12
```



```
Float division, e.g., 13.5/6.2
Addition, e.g., 5 + 8.0
Subtraction, e.g., 8 - 3
```

8. Assign numbers to variables and use the variables in calculations. Use the variable explorer pane to see the values of each variable.

```
In: x = 1
In: y = 2
In: z = 3
In: v = y * z - x
In: w = x - y * z
```

Lists and tuples are sequences of strings and numbers. Lists, which are mutable, are created using square brackets []. Tuples, which are immutable, are created using parentheses (). Each element is separated by a comma.

9. Create a list that has a tuple as one element.

```
In: mixlist = ['a', 'b', 'c', (x,y,z)]
```

The first three elements are strings, as indicated by the single quotation marks. The fourth element is a tuple of three variables. The variables must already have been defined, otherwise an error will occur.

- 10. Check the length of the list.
 - In: len(mixlist)

The list is only four elements long because the tuple counts as just one element in the list.

11. Check the length of the tuple.

```
In: len(mixlist[3])
```

Elements of lists or tuples within lists or tuples are accessed using a second index value, or a third for lists within lists, and so on.

- 12. Retrieve the first element in the tuple.
 - In: mixlist[3][0]
- 13. Try changing one of the elements in the tuple.
 - In: mixlist[3][1] = w

You should again see an error:

TypeError: 'tuple' object does not support item assignment

14. Replace the tuple element and output the result in the console.

```
In: mixlist[3] = w
In: mixlist
```

ArcPy has many other data types, such as feature classes, layers, and rasters. However, these are generally specified as strings, which are accessed and manipulated using ArcPy tools.

```
highered@esri.ca
```



Part C: Control Structures

Python scripts are interpreted linearly, from start (the first line) to finish (the last line). Control structures allow you to change that flow, either skipping over part of the code with if-then-else statements, for example, or repeating parts of the code with for loops. Recall that syntax is very important, as the indentation of a line determines whether or not it is part of a control structure.

1. Create a variable to test.

In: test = -5

Conditional Statements

if statements evaluate a Boolean expression. If the result is True, the code within the if block is executed. If the result is False, the code is skipped.

2. Use an if statement to check if the value is less than zero, and if so output a message. Press Enter twice to exit the control statement.

```
In : if test < 0:
...: print ("The value is less than 0.")
```

The output in the IPython console should be similar to the image below.

```
In [35]: if test < 0:
    ...: print ("The value is less than 0.")
    ...:
The value is less than 0.
```

The Python interpreter recognizes if as a keyword and looks for the colon : at the end of the line when you press Enter. If it is missing, a syntax error occurs.

```
SyntaxError: invalid syntax
```

If the syntax is correct, the next line will be automatically indented in the console. Similarly, in the Editor pane, the next line after a colon will be automatically indented. More importantly, the Editor will also add the colon for you if you forget.

3. Add an else statement to output an appropriate message if the condition fails. You can use the up and down arrows to recall previous input lines in the console so that you don't have to retype them.

```
In : test = 14
In : if test < 0:
...: print "The value is less than 0."
...: else:
...: print "The value is not less than 0."</pre>
```

Pay attention to the line indentations: if and else are not indented, while both print statements are indented one tab, which is equal to four spaces. Although indents are automatically added, they are not automatically removed so you will have to un-indent else.



You can add more conditions using elif statements (else-if). If more than one condition could be true, you need to consider which is the more important because only the first code block will be executed.

4. Modify the structure to include an elif statement.

```
In : test = -4
In : if test < -5:
...: print "The value is less than -5."
...: elif test < 0:
...: print "The value is less than 0 but not less than -5."
...: else:
...: print "The value is not less than 0."</pre>
```

You can also nest control statements, i.e.,

```
if condition:
if condition:
...
else:
```

For Loops

Because a Python script is interpreted line by line every time it is run, you want to use as few lines of code as possible to keep the execution time as short as possible. Loops execute the same block of code as long as a condition is met.

5. Create a list.

In: control list = [1, 2, 3, 4, 5]

A for loop takes a list and executes the code block once for each element in that list.

6. Use a for loop to print out the numbers in the list.

```
In : for x in control_list:
...: print (x)
```

Notice that the syntax for a loop is the same as for an if-then-else statement. The output in your interactive Python interpreter should be similar to the image below.

```
In [44]: for x in control_list:
    ...: print (x)
    ...:
1
2
3
4
5
```

This approach to using a list in a for loop only retrieves the value from the list; it does not give you access to the list itself. If you need to manipulate the list, you can use the range() function together with len().



7. Use the range () function to print out both the index and the value of each item in control list.

```
In : for i in range(len(control_list)):
    print (i, control list[i])
```

When range() is passed only one parameter, it creates a list from 0 up to, but not including, the parameter value, incrementing by 1. It can also be passed both start and stop parameters and optionally a third parameter for the step size. For example, in the following, the start value is 1, the end value is the length of the list (5), and the step size is 2.

```
In [48]: for i in range(1, len(control_list), 2):
    ...:
    print (i, control_list[i])
    ...:
1 2
3 4
```

There are other control structures, such as switches and while loops. If you are interested in learning more about these, consult the References and Reading list at the beginning of the tutorial for a list of suggested Python resources.

Part D: Modules and Functions

There are a limited number of built-in functions in Python. Others can be made available in your script, or the interactive Python interpreter, by importing additional modules. The functions can then be called using the format <module>.<function>. Although ArcPy is, in fact, a package of modules, it works the same way as any other package.

- 1. Open a new file and save it as **tutorial_partD.py**.
- 2. Add the following line to the file to import the math module into the script.

import math

In Spyder, code auto-completion for imported modules should be available within the Editor pane without you having to do anything. In IDLE and possibly other IDEs, you will need to import non-standard Python modules such as ArcPy in the interactive Python interpreter for the auto-completion to be available.

3. Activate code auto-completion for the math module by typing the following:

math.

As soon as you type the period, a list should pop up of all the available functions in the math module. Scroll through the list, or use the up and down arrows.

4. Select a function by double-clicking it or pressing Tab when it is selected, e.g.,

math.sqrt

5. Open brackets (for the function and use the hint that appears to pass the correct number and type of parameters to the function.



math.sqrt()



Notice that Spyder automatically adds a closing bracket. Remember that you can also get information about a function by pressing Ctrl-I.

 Add more functions as well as variables and print functions to your script so that messages are output when you run it. If you want to get user input while the script is running, you can use the input () function, e.g.,

```
x = input("Enter a value: ")
```

In IDLE and Spyder, this will cause a message to display in the console. The user must type in a value and press Enter. The user's input will be read as a string. You will need to cast it as a number type to use it in any math functions.

```
In [49]: runfile('C:/PythonTutorial/tutorial_partD.py', wdir='C:/
PythonTutorial')
```

Enter a value:

An alternative is to pass in arguments when running the script. In Spyder, open the Run Settings *k* dialog window, check the box beside Command line options and enter the value(s). Click OK to save the settings for later use, or Run to run the script immediately.

🐇 Run Settings	8 23
Select a run configuration:	
C:\PythonTutorial\tutorial_partD.py	•
Console Execute in current Python or IPython console Execute in a new dedicated Python console 	
Execute in an external System terminal	
General settings Clear all variables before execution (IPython consoles) Command line options: Working directory: Enter debugging mode when errors appear during execution	12.5 C:\PythonTutorial
Dedicated Python console Interact with the Python console after execution Show warning when killing running process Command line options:	
ОК	Run Cancel



Tutorial Python Basics

In your script, retrieving arguments requires importing the sys module and using its argv property. Arguments are read in as strings, and the first argument (index 0) is the name of the script. Therefore, to read in a floating-point number that was passed as the first argument, you would use argv[1] then cast the result to the proper type, e.g.

```
import sys
x = float(sys.argv[1])
```

7. Run your script to test it.

You can find samples of completed scripts in the <u>Python tutorial samples and data group</u> in ArcGIS Online.

Future Considerations

Ready to start using the ArcPy site package? Check out the **Introduction to Python Scripting with ArcPy** tutorial.

© 2017 Esri Canada. All rights reserved. Trademarks provided under license from Environmental Systems Research Institute Inc. Other product and company names mentioned herein may be trademarks or registered trademarks of their respective owners. Errors and omissions excepted. This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License. The Education and Research Group at Esri Canada makes every effort to present accurate and reliable information. The Web sites and URLs used in this tutorial are from sources that were current at the time of production but are subject to change without notice to Esri Canada.

