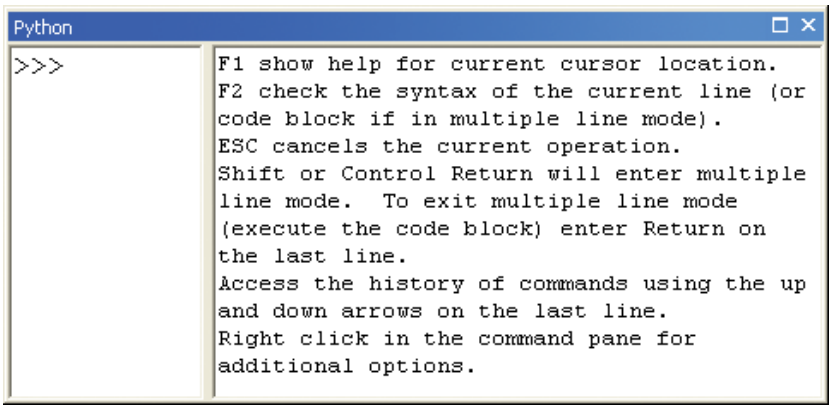
Exercise 3
Using the Python window

Open the Python window

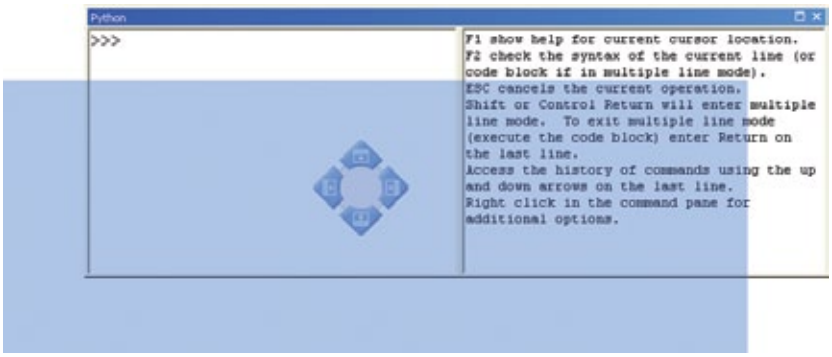
The Python window is an interactive Python interpreter and allows you to run Python code directly from within an ArcGIS for Desktop application.

1 **Start ArcMap. On the Standard toolbar, click the Python button to open the Python window.** ➔

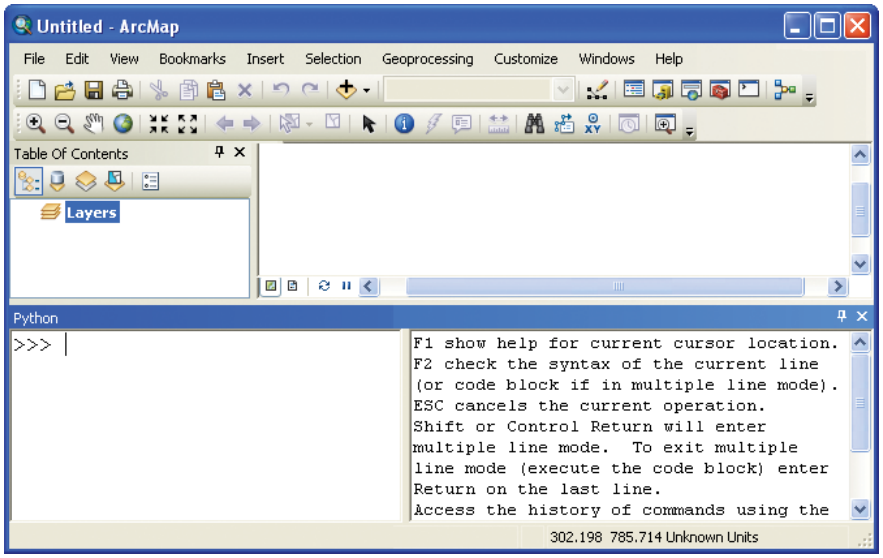


As with other windows in ArcMap, you can leave the Python window floating or dock it on any side of the ArcMap interface. Because you will be typing horizontal lines of code, it makes sense to dock the window at either the top or the bottom of the interface.

2 **To dock the Python window, drag the top bar of the Python window. This brings up eight arrows (four of which are visible here), indicating the various locations where you can dock the Python window.** ➔



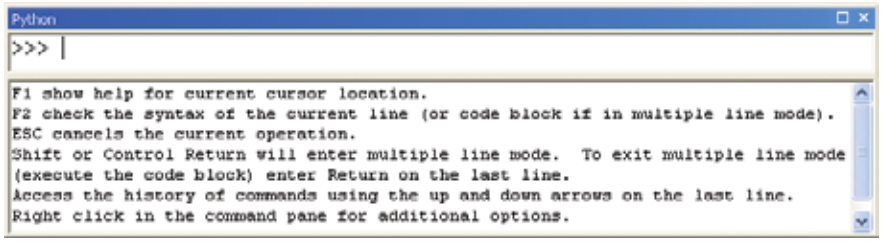
3 **Drop the Python window on the bottom arrow, so that the window is at the bottom of the interface.**



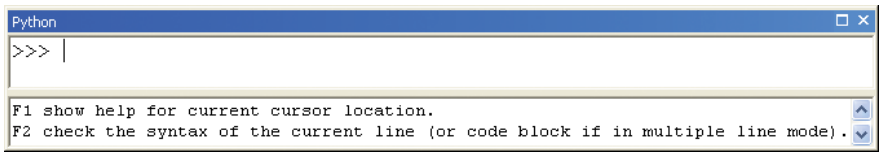
4 **To undock the Python window, drag the top bar again.**

The Python window itself can be resized, and the divider between the code section and the Help and syntax panel can be moved. Placement of the Help and syntax panel can also be controlled.

5 **Right-click in the Python window and click Help Placement > Bottom.**



6 **Drag the divider between the code section and the Help and syntax panel to adjust their sizing the way you like it.**



>>> TIP

Docking the window makes it a bit easier to keep working with your code and manage your layers in the ArcMap table of contents at the same time.

Write and run code

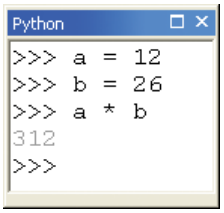
As in any interactive interpreter, Python code is run one line at a time, and the results are printed immediately.

1 Type the following code and press ENTER at the end of each line:

```
>>> a = 12
>>> b = 26
>>> a * b
```

Note: *Whether you use spaces here or not does not influence the execution of the code—that is,* `a = 12` *is the same as* `a=12`*. Spaces are commonly used in Python to make code easier to read but are often not required.*

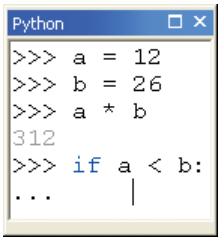
After you press ENTER, the line of code is executed, and the next line automatically starts with a new command prompt. The result of the preceding code is shown in the figure. ➔



Now you can try something different.

2 At the command prompt, type the following code:

```
>>> if a < b:
```



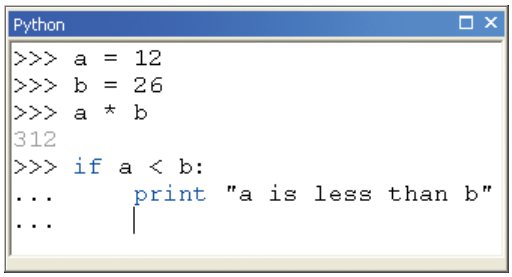
3 Press ENTER to run the line of code. The result is a secondary prompt at the next line, consisting of three dots (...). ➔

It means that the interactive Python interpreter recognized the start of the multiline construct. The `if` statement is the first line of a block of code, and at least one more line of code is needed for the code to run successfully. The Python window automatically indents the next line of code.

4 At the secondary prompt (...), enter the following code. Because this is the block of code following the `if` statement, the code has been indented. (This indentation was not automatic in ArcGIS 10.0 and had to be added manually by typing four spaces.)

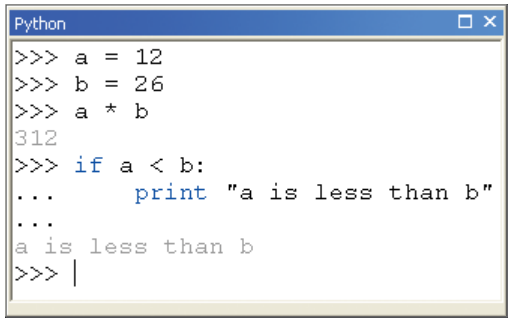
```
... print "a is less than b"
```

5 **At the end of the line of code, press ENTER.** Notice the result. The line of code is not executed, and the next line starts with another secondary prompt. ➔



6 **With the pointer at the start of the next line of code, and without entering any code, press ENTER.** ➔

When you are working with the secondary prompt, code is executed only when you press ENTER twice. This runs all lines of code following the last primary prompt.

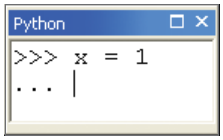


The Python window provides the secondary prompt automatically when it recognizes a multiline construct. However, you can also force the secondary prompt by pressing CTRL + ENTER.

7 **Right-click in the Python window and click Clear All.** This removes all lines of code from the Python window.

8 **At the primary prompt, enter the following code:**

```
>>> x = 1
```



9 **At the end of the line of code, press CTRL+ENTER.** ➔

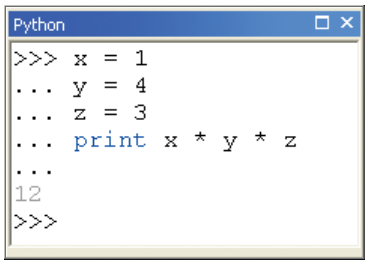
This brings up the secondary prompt. You can keep entering lines of code and pressing ENTER at the end of each line—the secondary prompt continues to appear, and the code is not executed until you press ENTER twice.

10 **At the secondary prompt, enter the following code:**

```
... y = 4
... z = 3
... print x * y * z
```

Note: *There is no need to enter spaces here following the secondary prompt. In an earlier example using the secondary prompt, spaces were needed to create a block of indented code, but this is not the case here.*

11 At the end of the last line of code, press ENTER twice. ➔



The use of CTRL+ENTER makes it possible to complete several lines of code before running it.

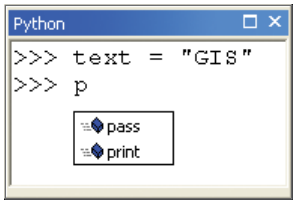
As you were typing the `print` and `if` statements, you probably already noticed the code autocompletion prompts. Next, you will take a closer look at how these prompts work.

12 Right-click in the Python window and click Clear All.

13 At the primary prompt, enter the following code and press ENTER:

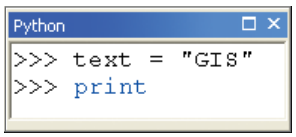
```
>>> text = "GIS"
```

14 At the next line, start typing the `print` statement. ➔



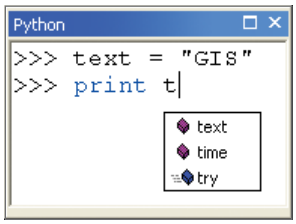
As soon as you start typing the letter *p*, you are prompted by a list of suggestions. These code autocompletion prompts include any text that would be logical based on Python syntax. In this case, there are two Python statements that start with the letter *p*. You can select the option you want using your pointer or by using the UP ARROW and DOWN ARROW keys.

15 Select the `print` statement and press the TAB key. ➔



For a statement as short as `print`, using the prompts does not save much typing, but autocompletion prompts can be very helpful as reminders of the proper syntax, and they can help you avoid making typos.

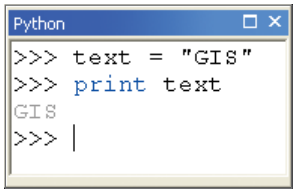
16 After the `print` statement, type a space, followed by the letter *t*. ➔



Notice that the list of suggestions is not limited to built-in Python terms but also includes `text` since this was used earlier in the code.

Note: *The term* `text` *is a variable here, a term that is covered in chapter 4.*

17 Select the `text` variable and press the TAB key. Then press ENTER to run the code. ➔



It is worthwhile to note that you do not have to use the code autocompletion prompts. Instead of selecting one of the suggested terms, you can simply continue typing. You can also turn off the prompts by right-clicking in the Python window and selecting or clearing the Show Default Choices option. However, you can benefit from code

autocompletion prompts, because using them reduces typos, makes writing code faster, and shows all your options in the drop-down list.

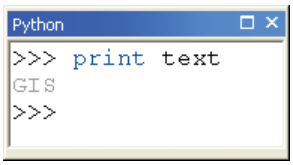
You have already seen how to clear code: right-click in the Python window and click Clear All. However, you can also continue to run lines of code, and the window will start to scroll downward if the lines of code do not fit in the window. Also, clearing the lines of code does not refresh the interactive Python interpreter—rather, the code executed earlier is still in memory.

18 Right-click in the Python window and click Clear All.

19 At the command prompt, type the following code and press ENTER:

```
>>> print text
```

Notice that the code printed the correct text, as shown in the figure, even though the lines of code are no longer visible. ➔



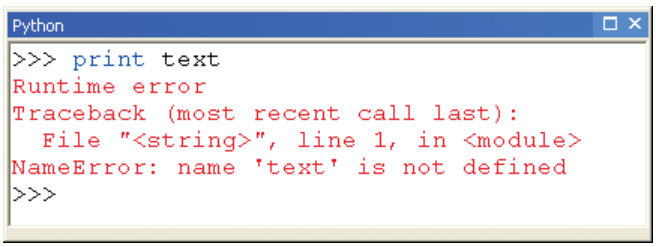
Closing and opening the Python window does not remove the code, nor does it remove the code from the interactive Python interpreter. However, closing ArcMap removes the code from memory, and you can never use it again.

20 Close ArcMap and click No if asked to save any changes to the map document. Start ArcMap again. Open the Python window if necessary.

21 At the command prompt, type the following code and press ENTER:

```
>>> print text
```

Notice the result, as shown in the figure. ➔



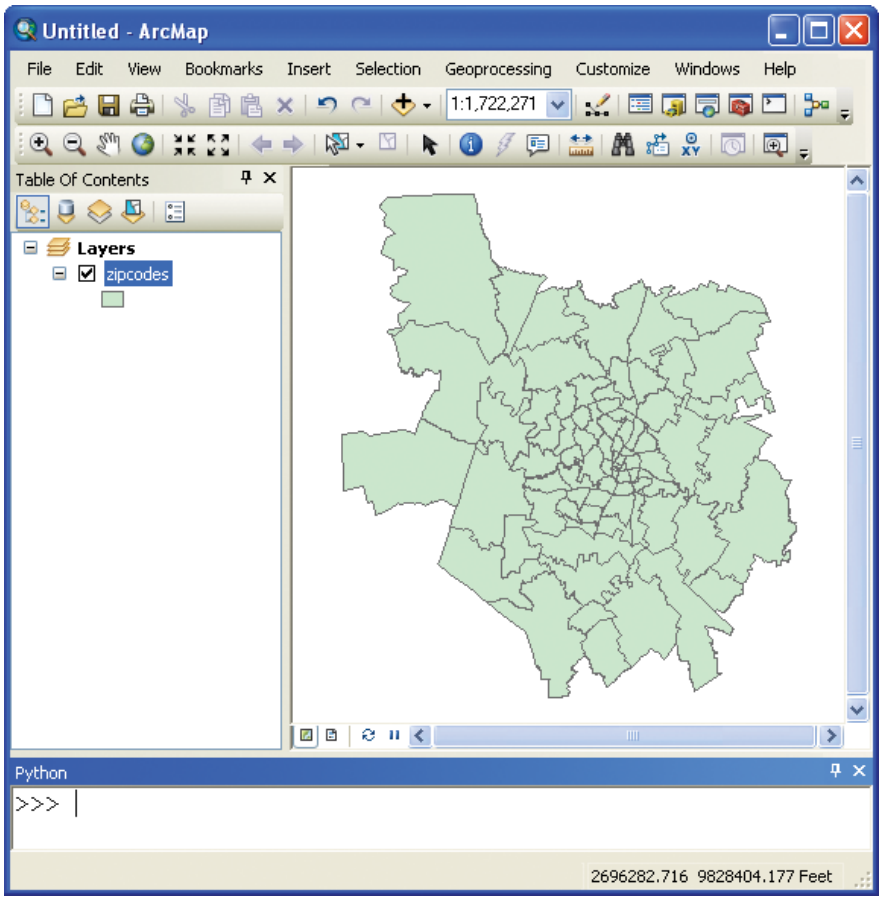
The variable text is not defined and therefore the Python window produces an error message. Code in the Python window is removed from memory when the ArcGIS for Desktop application is closed. Later in this exercise, you will see that there is an option to save the Python code.

Run a geoprocessing tool

Next, you will run a geoprocessing tool from the Python window that works with a layer in ArcMap. Don't worry too much about the syntax of the code for now.

1 **On the ArcMap Standard toolbar, click Add Data and browse to the C:\EsriPress\Python\Data\Exercise03 folder.**

2 **Select the zipcodes.shp file and click Add.**



3 **At the prompt, start typing the following:**

```
>>> count = arcpy.G
```

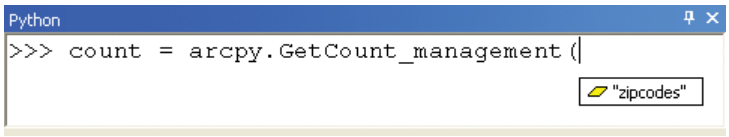
The code prompts you by providing a list of options.

4 **Select the `GetCount_management` option from the list and press the TAB key.**

5 Type an opening, left paren [(]:

```
>>> count = arcpy.GetCount_management(
```

The code prompts you by providing the name of the only layer in ArcMap.



6 Select this layer, press the TAB key, and type a closing, right paren [)]:

```
>>> count = arcpy.GetCount_management("zipcodes")
```

7 Press ENTER to run the line of code. The code runs the Get Count tool. Once it is finished running, a pop-up notification appears in the notification area, at the far right of the taskbar. Geoprocessing messages also appear in the Help and syntax panel of the Python window.

Note: If you have disabled background processing under Geoprocessing Options, no pop-up notification will appear and messages appear only in the Help and syntax panel of the Python window.

8 At the prompt, enter the following code and press ENTER:

```
>>> print count
```

The code prints the result of the Get Count tool: 80.

The specific syntax used in this example is covered in detail in chapter 5. For now, the important thing to remember is that you can run code in the Python window that interacts with spatial data in the map document as well as on disk.

Get help in the Python window

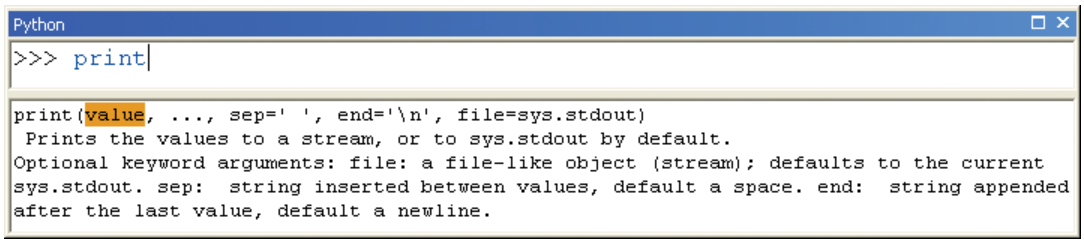
You have already seen several examples of code autocompletion prompts, which can be of great help. These prompts are context sensitive, meaning that the suggestions include only terms that are logical based on Python syntax.

There are several other ways to get assistance.

1 Right-click in the Python window and click Clear All.

2 Make sure the Help and syntax panel is visible. →

3 At the command prompt, type the `print` statement.

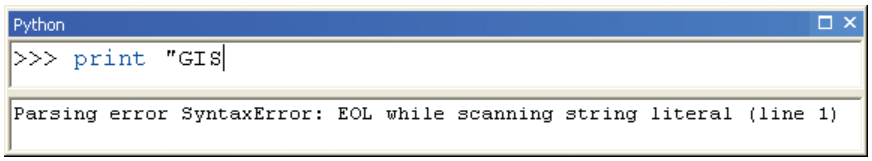


What is shown in the Help and syntax panel is the syntax for the `print` statement. When you are just getting started in Python, the wording may appear a little cryptic.

4 Continue the line of code with the following:

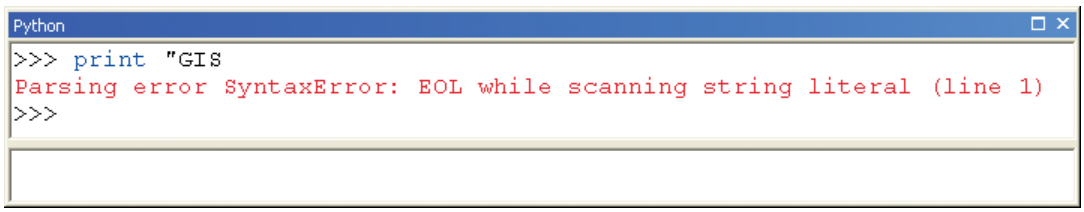
```
>>> print "GIS
```

5 With the pointer at the end of the line of code, press F2.



This brings up syntax checking for the current line of code. In this case, an end-of-line (EOL) error is detected. Pressing F2 effectively prints any syntax errors that will occur when the line of code is executed.

6 Without fixing the syntax error, press ENTER to run the line of code.

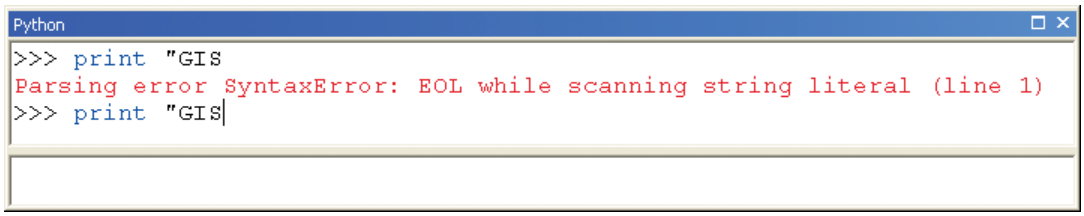


Notice that this is the same error as reported using syntax checking. The F2 key does only syntax checking—other types of errors are discovered only when a tool is actually run.

If you make an error, you may be tempted to correct prior lines of code that have already been run. However, you can only run code at the current command prompt, so fixing prior lines of code is not an option. To save on typing, you can copy the line of code, paste it after the current

command prompt, and fix the error. Since this is such a common task, there are built-in shortcuts to make it easier.

7 At the command prompt, press the UP ARROW key to bring up the previous line of code.



Now you can copy and paste the line of code, fix it, and then run it. The UP ARROW and DOWN ARROW keys can be used to scroll up or down to any previous line of code in the current session.

Save your work

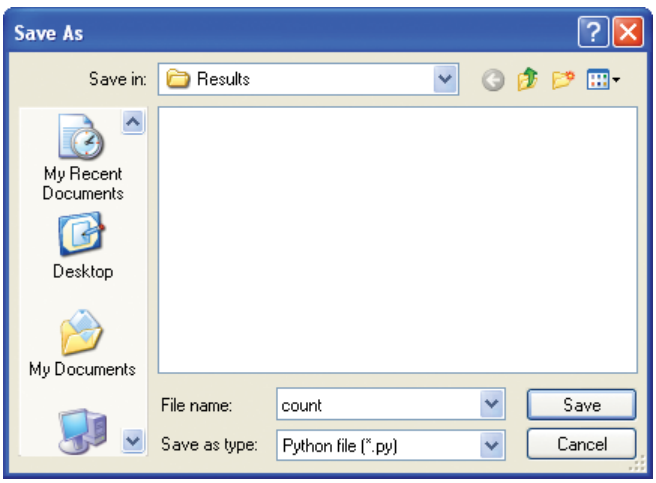
Code in the Python window is not saved with a map document. If you want to reuse your code later, you can save the code to a script.

1 Right-click in the Python window and click Clear All.

2 At the command prompt, enter and run the following lines of code.

```
>>> count = arcpy.GetCount_management("zipcodes")
>>> print count
80
```

You can copy and paste lines of code from the Python window to a text editor or to a script window of a Python editor to save and reuse. However, this also copies text that is not code, such as prompts, results, and messages.



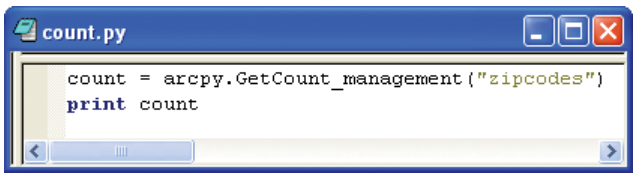
3 Right-click in the Python window and click Save As.

4 On the Save As dialog box, browse to the C:\EsriPress\Python\Data\Exercise03\Results folder and save your file as count.py. ➔

Next, you can open the script file in a Python editor.

5 **Start PythonWin.**

6 **On the Standard toolbar, click the Open button, browse to the Results folder for exercise 3, select the count.py script, and click Open.**



Notice that the resulting script file contains code only.

7 **Close PythonWin.**

Load existing code

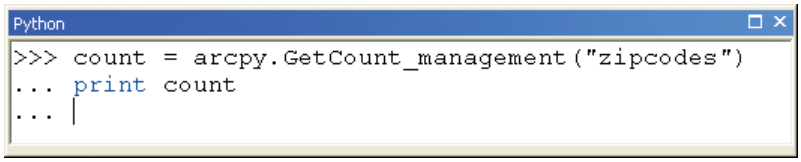
You can also load existing code into the Python window.

1 **Return to ArcMap.**

2 **Right-click in the Python window and click Clear All.**

3 **Right-click in the Python window and click Load.**

4 **On the Open dialog box, browse to the Results folder for exercise 3, select the count.py script, and click Open.**



The Python script loads into the Python window. Notice that the lines of code are preceded by the secondary prompt, meaning that the code is not run line by line. You can now make changes to the code prior to running it.

5 **Close ArcMap. There is no need to save your map document.**