Chapter 1
Introducing Python

1.1 Introduction

Python is a programming language that is both simple and powerful. For those who have struggled with learning programming languages in the past, this may come as a pleasant surprise.

This chapter describes some of the main features of Python and its use as a programming language to write scripts for ArcGIS. The logic and structure of this book and the accompanying exercises is described, followed by some examples of how Python is used. The final part of this chapter introduces Python editors that make it easier to write and organize your code.

1.2 Exploring the features of Python

Python has a number of features that make it the programming language of choice for working with ArcGIS. Among them:

It's simple and easy to learn: Python is easy to learn compared with other highly structured programming languages like C++ or Visual Basic. The syntax is simple, which gives you more time to focus on solving problems than having to learn the language itself.

It's free and open source: Python is free and open source software (FOSS). You can freely distribute copies of the software, read the source code, make changes to it, and use pieces of it in new free programs. One of the reasons Python works so well is that it has been created, and is constantly being improved, by an active and dedicated user community. The FOSS nature of Python makes it possible for Esri to distribute Python with ArcGIS software.

It's cross platform: Python is supported on different platforms, including Windows, Mac, Linux, and many others. Python programs can work on any of these platforms with minimal change or sometimes no change at all. Since the ArcGIS for Desktop application runs only on Windows, this may

not seem like a big advantage, but the user community for Python is large, in part due to its cross-platform nature.

It's interpreted: Many programming languages require that a program be converted from the source language, such as C++ or Visual Basic, into binary code that the computer can understand. This requires a compiler with various options. Python is an interpreted language, which means it does not need compilation to binary code before it can be run. You simply run the program directly from the source code, which makes Python easier to work with and much more portable than other programming languages.

It's object oriented: Python is an object-oriented programming language. An object-oriented program involves a collection of interacting objects, as opposed to the conventional list of tasks. Many modern programming languages support object-oriented programming. ArcGIS is designed to work with object-oriented languages, and Python qualifies in this respect.

1.3 Comparing scripting vs. programming

Although Python is a programming language, it is often referred to as a scripting language. So, what is the difference? In general, a *scripting* language refers to automating certain functionality within another program, while a programming language involves the development of more sophisticated multifunctional applications. Scripting is a programming task that allows you to connect diverse existing components to accomplish a new, related task. Scripting is the "glue" that allows you to put various existing elements together. Programming, on the other hand, allows you to build components from scratch, as well as the applications that incorporate these components. Languages that work with these low-level primitives and the raw resources of the computer are referred to as system languages. Examples of system languages include C++ and .NET languages. Scripting languages use built-in higher-level functions and mask the detail a system language deals with. Examples of scripting languages include Python, Perl, PHP, and Ruby.

Esri, for example, relies primarily on C++ as the programming language to create ArcGIS software and all the different components or *objects*, called ArcObjects, that you find in the software. You can then use C++ to write your own software that utilizes these same ArcObjects as well as create your own objects. However, you can also use scripting to access the existing functionality of ArcGIS and connect functions in new ways to extend that functionality.

One of the strengths of Python is that it is both a scripting and a programming language, although it does not have quite the depth of a system language like C++. You can use it for relatively simple scripts as well as more advanced programming tasks. The focus of this book is writing Python scripts to carry out tasks in ArcGIS. Python can also be used for application development, but these aspects of using Python are not

addressed in this book. Python is used here as an interpreted language to work directly with the existing functions available in ArcGIS.

1.4 Using scripting in ArcGIS

ArcGIS 9 introduced scripting support for many popular scripting languages, including Python, VBScript, JavaScript, JScript, and Perl. ArcGIS is COM compliant, meaning that it uses the widely used Component Object Model (COM) software architecture. Scripting languages are able to access all the functions available in ArcGIS, including any extensions. This makes scripting a very attractive and efficient method for automating tasks. Although the same automation can be accomplished using a system language like C++ or a .NET language, scripting often requires much less effort.

Python scripting has become a fundamental tool for geographic information systems (GIS) professionals to extend the functionality of ArcGIS and automate workflows. Several years ago, probably the most widely used approach was to use the built-in VBA (Visual Basic for Applications) programming tools. Since then, however, Python has emerged as a robust complement and alternative to VBA programming. Starting with ArcGIS 10, the VBA development environment is no longer installed by default, and Esri is actively discouraging the continued use of VBA. Although application development will continue to employ languages such as C++ and .NET, Python has a number of advantages, especially for GIS professionals who are not full-time programmers.

Python is not the only scripting language that can be used with ArcGIS, but it has certainly become the most widely used. This is largely because Python has the ease of use of a scripting language, as well as the programming capability of a complete developer language. Python is included in a typical ArcGIS for Desktop installation. Python has also been directly embedded in many tools in ArcGIS for Desktop. The Spatial Statistics toolbox, for example, consists almost entirely of Python scripts—even though the casual user does not necessarily notice (or need to). ArcGIS 10 has seen further integration of Python within the ArcGIS interface, and Esri has officially embraced Python as the preferred scripting tool for working with ArcGIS. Additional enhancements have been introduced in ArcGIS 10.1.

1.5 Python history and versions

Python was created by Guido van Rossum at the Centrum voor Wiskunde en Informatica (CWI) in the Netherlands and first released in 1991. Van Rossum remains active in the ongoing development of Python but has been joined by a large number of contributors. Compared to other languages,

Python has gone through a limited number of versions, reflecting a philosophy of incremental change and backward compatibility.

Python's features include lists, dictionaries, and strings, as well as more advanced elements such as metaclasses, generators, and list comprehensions. The robustness of Python reflects the need to include the basic features all programmers need as well as the more desired advanced ones that are common in other, more complex programming languages.

The Python version that is recommended for use with ArcGIS 10.1 is Python 2.7. Although you can download and install any version of Python for free, the installation of ArcGIS 10.1 comes with Python 2.7.2. Although version 2.*x* still works fine and continues to be widely used, a major new version of Python was deemed necessary to remove a number of small problems that accumulated over the years and make the language even cleaner. Although there are a number of differences between versions 2.7 and 3.*x*, the basic structure of the language has not changed. Despite the availability of version 3.*x*, ArcGIS 10.1 currently works with version 2.7. If and when version 3.*x* is adopted as the preferred version for use with ArcGIS, various utilities are available to convert code between the two versions.

It should be noted that although the installation of ArcGIS comes with Python, Esri did not develop Python. Esri relies on the FOSS nature of Python to distribute it with ArcGIS as the recommended scripting language. In addition, Esri has created functionality in ArcGIS to make it easy to work with Python. However, Python is widely used for tasks other than writing scripts to work with ArcGIS. The added benefit is that as you start learning more about Python in this book, you will be able to start using it for both ArcGIS and other tasks as well.

1.6 About this book

Python Scripting for ArcGIS consists of two elements:

1. The printed book, which covers the theory of using Python

2. A set of exercises in digital format that accompany the printed book (on the disk in the back)

The printed book consists of 14 chapters that explain the structure and syntax of Python and illustrate how to write scripts for ArcGIS. Sample code is provided throughout the text, but the book itself does not provide detailed step-by-step instructions. The exercises on the accompanying DVD provide detailed instructions for hands-on learning. There is one exercise for each chapter in the book. You are encouraged to first read a chapter from the book, complete the corresponding exercise, and then move on to the next chapter. Most of the exercises include challenges at the end, which allow

you to practice your skills. Solutions for these challenges are included on the Data and Exercises DVD.

The structure of the book

Chapter 1 provides an introduction to Python and gives you the opportunity to get started using Python editors and the Python window in ArcGIS.

Chapter 2 introduces the ArcGIS geoprocessing environment, including the use of tools and the ModelBuilder application. Experienced ArcGIS users will be familiar with most of the material, but a good review will be beneficial. Knowing what is possible with the existing set of tools in the ArcToolbox window will be very helpful in writing effective scripts. Similarly, Python scripts and ModelBuilder are often used in combination, so a good knowledge of ModelBuilder is recommended to get the most out of Python scripting.

Chapter 3 describes the Python window in ArcGIS, which serves as an interactive interpreter. The Python window allows you to run one or several lines of Python code directly from within an ArcGIS for Desktop application. Code written in the Python window can be saved to a script, and existing code from a script can be loaded into the Python window.

Chapter 4 explains the fundamentals of the Python language, including the use of statements, expressions, functions, methods, modules, and controlling workflow, as well as best practices for writing scripts. Chapter 4 covers the basic syntax of Python for the novice Python user. Experienced Python users will be familiar with this material.

Chapter 5 describes the ArcPy site package that was introduced with ArcGIS 10. ArcPy includes numerous modules, classes, and functions, which provide an effective way to integrate ArcGIS and Python. Chapter 5 also includes working with data paths, environment settings, and licensing issues.

Chapter 6 includes techniques for data exploration, including describing data and data structures, as well as working with strings, lists, tuples, and dictionaries to characterize data prior to using it in other operations.

Chapter 7 introduces how to manipulate spatial data, including the use of cursors, searching for data, and working with tables and fields. The lessons in chapter 7 make it possible to use Python to run SQL queries on spatial data.

Chapter 8 describes how to work with the geometric properties of spatial objects, including how to utilize the properties of existing features and how to create new features.

Chapter 9 describes how to work with rasters in Python, including the ArcPy Spatial Analyst extension module. This module contains a number of specialized map algebra operators and other classes for using rasters for spatial analysis.

Chapter 10 describes the ArcPy mapping module for automating mapping tasks. This includes working with map documents, data frames, and layers, as well as exporting and printing maps.

Chapter 11 describes error-handling techniques for anticipating common errors and making your code more robust. The code debugging environment using the PythonWin Debugger, which allows you to test your code interactively, is also covered.

Chapter 12 demonstrates how to create custom functions and classes in Python. This makes it easier to organize more complex code and use parts of your code in multiple scripts.

Chapter 13 explains how to create custom script tools, which make Python scripts available as tools in the ArcToolbox interface. It is one of the preferred methods for sharing Python scripts with other users and also makes it easier to add a Python script as a tool to a larger sequence of operations in ModelBuilder.

Chapter 14 outlines strategies for sharing script tools with others, including how to organize your files, work with paths, and provide documentation for script tools.

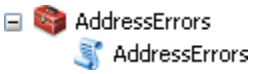
1.7 Exploring how Python is used

This section uses several examples to illustrate how Python is used to create scripts. The examples were obtained from scripts created by Esri and the ArcGIS user community. One of the reasons for presenting these examples is for you to become more familiar with looking at Python code. One of the best ways to learn how to write code is to work with existing code. You are not expected to be able to understand the code at this point, but the examples will give you a flavor of what is to come.

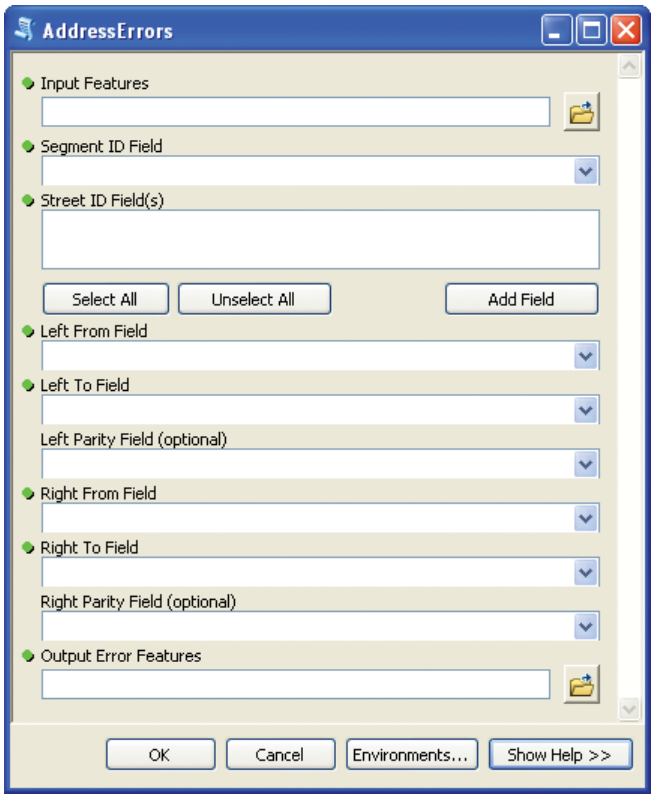
Example 1: Determining address errors

The AddressErrors script tool was created by Bruce Harold, an Esri employee. The AddressErrors tool inspects street centerlines for possible errors in address ranges associated with street segments. A polyline feature class is output that includes one feature for every error detected and contains attribution that profiles that error.

The script is made available as a tool in a toolbox. Although the script was written in Python, the functionality of the script can be accessed the same way as any other tool.

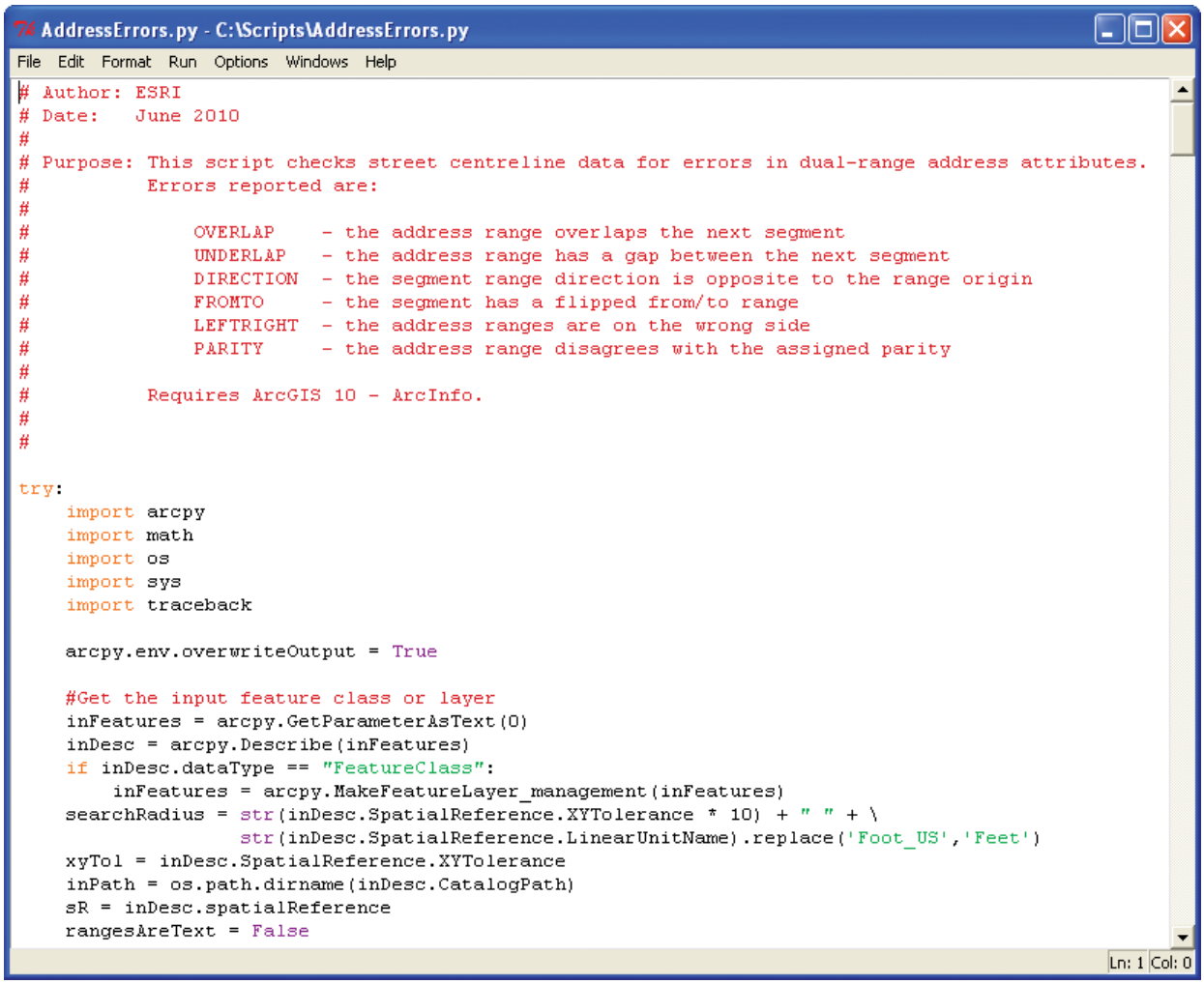


The tool dialog box looks like that of any regular geoprocessing tool.



The tool has 10 inputs, two of which are optional. The input features would typically consist of a polyline representing street centerlines and a number of attributes representing the range of street numbers. The output is a new feature class.

A single Python script is used in this tool, and the script can be opened to get an inside look at what the tool does. When you open the script in a Python editor, it looks like the one in the figure.



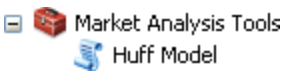
If you have not worked with Python or other programming languages before, this code can look a bit intimidating. However, the purpose of this book and the exercises that go with it is to make you familiar with Python syntax and the logic behind it for carrying out tasks in ArcGIS. So by the end of the book, you will be able not only to understand the preceding script, but also to write scripts of similar complexity.

Example 2: Market analysis using the Huff Model tool

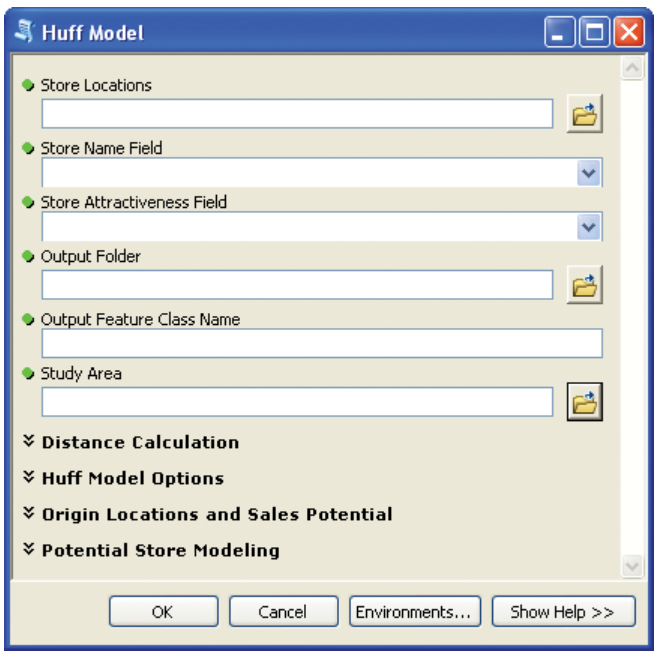
The Huff Model script was created by Drew Flater, an Esri employee. Here is a brief description of the tool as provided in the tool's instructions:

> *The Huff Model is a spatial interaction model that calculates gravity-based probabilities of consumers at each origin location patronizing each store in the store dataset. From these probabilities, sales potential can be calculated for each origin location based on disposable income, population, or other variables. The probability values at each origin location can optionally be used to generate probability surfaces and market areas for each store in the study area.*

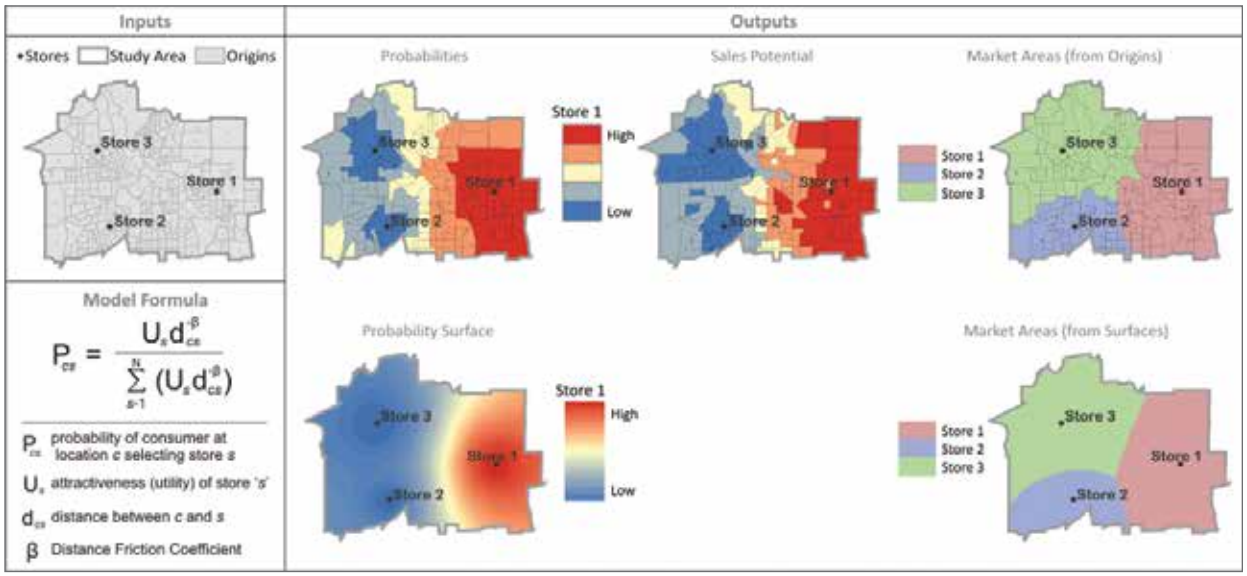
This is a relatively sophisticated script, but like the first example, it is written entirely in Python. The script is also made available as a tool in a toolbox as shown in the figure.



The tool dialog box has a lot of options for inputs, outputs, and analysis settings, reflecting the nature of the Huff Model.



A tool Help file is provided that provides detailed descriptions of how the tool works, including the inputs, model formula, and outputs, as illustrated in the diagram that is part of the Help file and shown in the figure.



All the code for this tool is provided as a single Python script, part of which is shown in the figure.

HuffModel.py - C:\Scripts\HuffModel.py

File Edit Format Run Options Windows Help

```
# ---------------------------------------------------------------------------
# HuffModel.py
# Created: 4/13/2007 by Drew Flater
# Usage: Creating probability-based trade areas for retail stores
# ---------------------------------------------------------------------------

# Import system modules
import sys, string, arcgisscripting, os, traceback, shutil, re

# Create the Geoprocessor object
gp = arcgisscripting.create(93)

# Set overwrite
gp.overwriteoutput = 1

def AddPrintMessage(msg, severity):
    print msg
    if severity == 0: gp.AddMessage(msg)
    elif severity == 1: gp.AddWarning(msg)
    elif severity == 2: gp.AddError(msg)

# Start traceback Try-Except statement:
try:
    # Script parameters...
    stores = gp.getparameterastext(0)
    store_name = gp.getparameterastext(1)
    store_attr = gp.getparameterastext(2)
    outfolder = gp.getparameterastext(3)
    fc_name = gp.getparameterastext(4)
    studyarea = gp.getparameterastext(5)
    blockgroups = gp.getparameterastext(6)
```

Ln: 1 Col: 0

Given the complexity of the Huff Model, the Python script is quite lengthy, at over 700 lines of code. However, most of the tool's basic elements are the same as those found in simpler scripts. So as soon as you become familiar with what these elements are, you can start actually reading and understanding some of the more complex scripts.

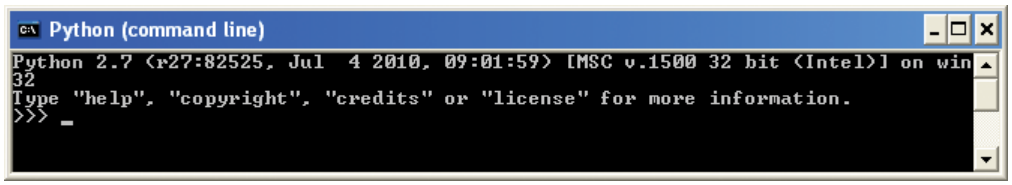
Once you learn the basics, from the chapters in this book, of how to use Python for writing scripts, you will find one of the best ways to keep learning Python scripting is to work with existing code. Using example code can also speed up the process of writing your own scripts.

*Note: The two examples provided here were obtained from the Geoprocessing section of the ArcGIS Resource Center (*http://resources.ArcGIS.com*). Of specific interest is the Geoprocessing Model and Script Tool Gallery (*http://resources.ArcGIS.com/gallery/file/geoprocessing*).*

1.8 Choosing a Python script editor

A computer script is essentially a list of commands that can be run by a certain program or scripting engine. Scripts are usually just plain text documents that have a specific file extension and contain instructions in a particular scripting language. Most scripts can be opened and edited using a basic text editor. However, using a specialized script editor to open a script provides additional functionality when you're writing the script and also allows you to run the script.

You can work with Python in a number of ways. The most basic approach is to use the so-called command line. If you have used other programming languages, you may be familiar with this type of interface. To access the Python command line in Windows, click the Start button, and then, on the Start menu, click All Programs > ArcGIS > Python 2.7 > Python (command line).

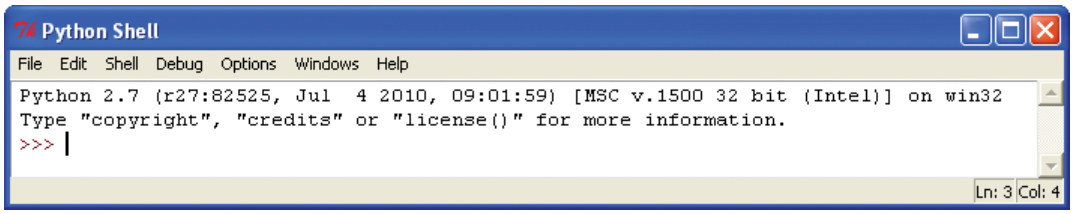


Although this interface provides full access to all of Python's functionality, it provides limited support for writing and testing scripts. So instead of using the command line interface, it is typically much more productive to use a Python script editor. A Python editor has a menu-driven interface and tools for organizing and testing Python scripts to make working with Python easier.

Python editors are also known as integrated development environments (IDEs). There are many different ones, including several open source and commercial packages. Some IDEs are developed for specific platforms, such as Windows, Mac, and Linux, and others are specifically designed to interface with particular programming languages, such as C++ and .NET languages. An extensive list of Python editors can be found on the Python wiki page (http://wiki.python.org/moin/PythonEditors). To a large degree, the

editor you use is a matter of preference, and experienced Python programmers all have their favorite ones. Python syntax remains the same for different editors, which is one of the strengths of Python.

The default IDE that comes with any installation of Python is the integrated development environment (IDLE). To access Python IDLE in Windows, click the Start button, and then, on the Start menu, click All Programs > ArcGIS > Python 2.7 > IDLE (Python GUI). GUI stands for graphic user interface. IDLE is also known as the Python shell.



Descriptions of the various menu items can be obtained from the Help menu. On the Python Shell menu bar, click Help > IDLE Help to view these descriptions. More information on IDLE can be found at http://www.python.org/idle.

Notice that the last line in the Python Shell window starts with >>>. This is the *prompt* of the interactive Python interpreter. It is where you can type code and press ENTER, and the interactive interpreter will carry out your command. Are you ready for your very first line of Python?

```
>>> print "Hello World"
```

When you press ENTER, the following output appears:

```
Hello World
>>>
```

So, what's happening here? When you press ENTER, the interactive Python interpreter reads the input command, prints the string `Hello World` to the next line, and gives you a new prompt on the following line, waiting for the next input. The term *printing* here refers to writing text to the screen.

You have now seen why Python is called an *interpreted* programming language. When you are finished typing your commands and press ENTER, the commands are interpreted and immediately carried out.

What if you type something else that does not make sense to the interactive Python interpreter? For example:

```
>>> I like eggs for breakfast
```

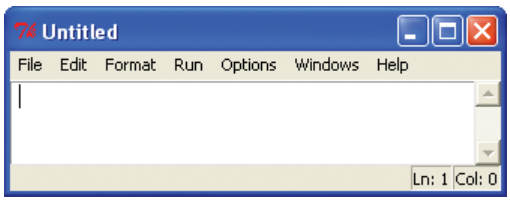
You get immediate feedback that the syntax is invalid:

```
SyntaxError: invalid syntax
>>>
```

You will also notice something else about the interactive Python interpreter. As soon as you type some text, it is given a color based on the nature of the input. For example, as soon as you type the word `print`, the word turns orange. Similarly, the string `"Hello World"` turns green. This is a way for the interactive Python interpreter to show how the input is being interpreted—in this case, orange is for statements and green is for strings. This is called *syntax highlighting* and is a helpful means of error checking as you write code. Be aware, however, that syntax highlighting conventions vary between Python editors, so don't get too used to a particular color scheme.

When you are learning the basics of Python syntax, it is useful to work directly in the interactive Python interpreter. You get immediate results that way, and you can keep going with new lines of code without worrying about having to save your work. However, when you are ready to write slightly more complex, multiline code, it will be more beneficial to write it as a script you can save. So remember that code written to the interactive Python interpreter is not meant to be saved.

Now you can see how writing a script differs from writing code in the interactive Python interpreter. On the Python Shell menu bar, click File > New Window. This opens a new window called Untitled. It is a script window, and there is no prompt.



Now type the same line of code as before:

```
print "Hello World"
```

When you press ENTER, nothing happens—that's because a script itself is not interactive. A script needs to be *run* as a program for the command to be executed. Before you can run a script, however, it needs to be saved. On the menu bar, click File > Save As and save your script as **hello.py**. The file extension .py indicates it is a Python script. Now it is time to run the

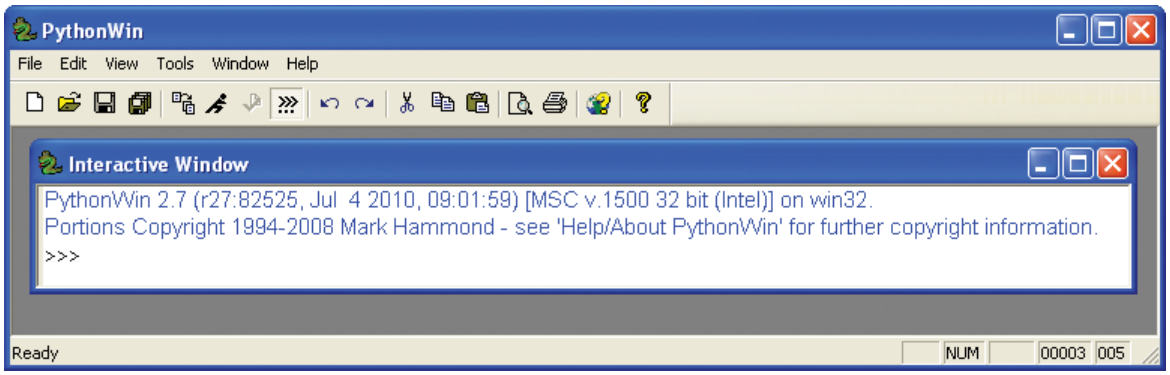
script. On the menu bar, click Run > Run Module. The string "Hello World" is printed to the interactive Python interpreter.

When you are coding, it is useful to have both the interactive Python interpreter and your script(s) open at the same time. If you want to try out something very quickly or check the syntax of a particular line of code, you can use the interactive Python interpreter. The script window contains the actual lines of code you want to save and keep working on. Occasionally, you can test your script by producing results that are printed to the interactive Python interpreter.

One widely used Python editor on the Windows platform is PythonWin. For the remainder of this book, it is assumed that you have access to PythonWin. The syntax, however, is the same, independent of the editor. The major differences between editors lie in how Python scripts are created, organized, and tested, although the syntax is the same.

> ***Note:*** *PythonWin is not installed by default when Python is installed during the ArcGIS installation. The corresponding exercise chapter, Exercise01, on the disk that comes with this book, includes instructions on how to install PythonWin.*

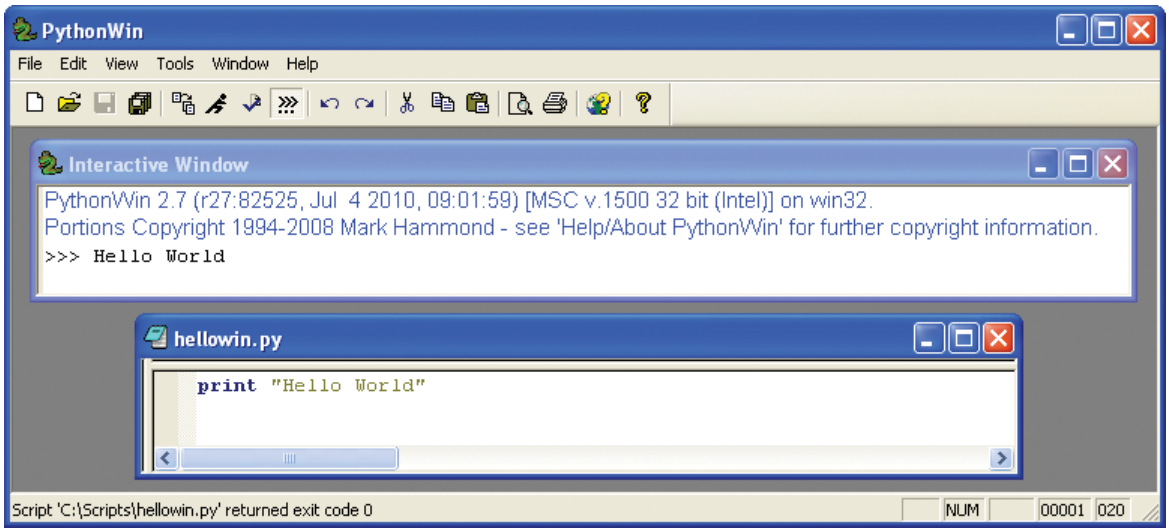
The basic PythonWin interface is shown in the figure. By default, it opens to an interactive Python interpreter called the Interactive Window, just like Python IDLE.



To create a new script, on the PythonWin menu bar, click File > New > Python Script. A new script window appears that can be resized so that both your interactive window and the script window are visible. To save the script, on the script window menu bar, click File > Save As. In the example that follows, the script is saved as `hellowin.py`. In the script window, enter the following line of code:

```
print "Hello World"
```

To run the script, on the script window menu bar, click File > Run. The result is printed to the Interactive Window. Notice that the syntax highlighting in PythonWin is slightly different.

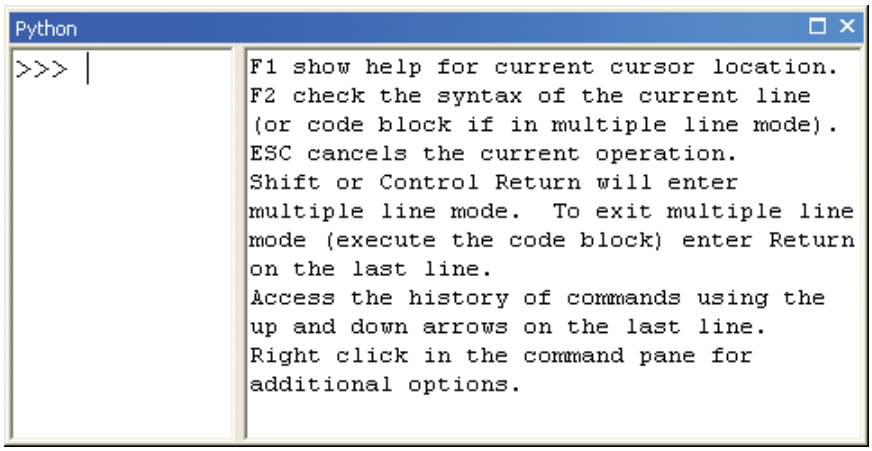


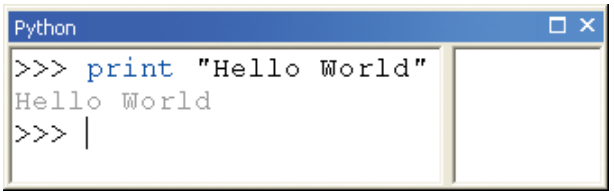
Note: *From now on, the examples in the book use PythonWin as the Python editor. Choosing a Python editor is largely up to you. PythonWin was selected for this book because it provides a relatively easy-to-use yet robust and proven environment for working with Python on the Windows platform. Experienced coders can use the code examples in this book in their preferred code editor, as long as it is Python compatible.*

In addition to dedicated Python editors such as IDLE and PythonWin, general-purpose code editors can also be used for writing Python scripts. These editors usually offer syntax highlighting, formatting options, and other types of options for working with code. Examples of widely used code editors include Bluefish, Context, and Notepad++. Typically, these code editors work with multiple programming languages, and as a result, experienced code developers may use a single editor to work with multiple languages. In contrast, PythonWin is designed to work exclusively with Python code. Dedicated Python editors have some advantages over text editors, including code autocompletion prompts and debugging procedures, which is one of the reasons why the PythonWin editor is used in this book rather than a more generic code editor.

Note: *Be careful using basic text editors, such as Notepad, that are not designed specifically for writing code. These editors do not preserve the proper formatting for a script and do not provide syntax highlighting or any other tools that help to write properly formatted scripts. Make sure a text editor is specifically designed for working with code such as the examples listed here.*

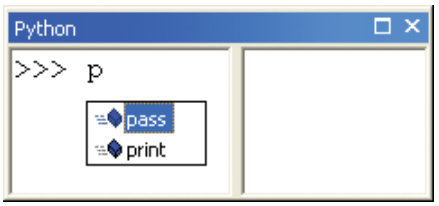
Finally, starting with ArcGIS 10, there is a way to use Python directly from within ArcGIS for Desktop applications that is both convenient and effective. The Command Line window in ArcGIS 9 has been replaced in ArcGIS 10 by the new Python window. When you open the Python window, you have an interactive interpreter for Python on the left and a Help section on the right.





You can enter code in the Python window and each line is executed immediately, as in any other interactive interpreter. ➔

The Python window allows for rapid testing of simple lines of code. One advantage is that it provides syntax suggestions, known as code auto-completion "prompts," as you type. For example, when you type the letter *p*, the Python window suggests the statements `pass` and `print` to complete the code. This allows for faster coding and fewer typos when the code is completed for you. ➔



The most effective use of the Python window is covered in greater detail in later chapters.

To review, Python scripts, by definition, are modules, which is Python's highest level of organizing code. The actual file that contains the Python script is a simple text file, and therefore any text editor can be used to write a script. The file extension for a Python script is .py, which is automatically associated with a Python editor.

> ***Note:*** *Python script names must start with a letter and can be followed by any number of letters, digits, or underscores (_). Script names should not use Python keywords—you will see later what those keywords are.*

Convention on sample code: Whenever sample code in the book is preceded by the prompt (>>>), the code is being written to an interactive interpreter (for example, the Python window in ArcGIS for Desktop applications or the Interactive Window in PythonWin). When you press ENTER,

the line of code is executed immediately. Whenever sample code is not preceded by the prompt, the code is being written in a script window and the script needs to be run for the code to be executed. Much of the sample code, however, can be run in both an interactive interpreter and a script.

Note: This book uses the PythonWin editor and the Python window in ArcGIS for Desktop applications to write and run code in Python. However, all the code can be used in any editor.

Points to remember

- ArcGIS for Desktop supports the use of scripting languages to automate workflows. Python is the preferred scripting language for working with ArcGIS.
- Python is not created by Esri. It is an open source programming language and therefore can be distributed by third parties, including Esri.
- Python is relatively easy to use. There is a large user community and there are many resources for learning Python. There is also a growing set of libraries for use in Python that provide additional functionality.
- One of the strengths of Python is that it is both a scripting language and a programming language. So you can use it for relatively simple scripts as well as for more advanced programming tasks. Using Python to develop scripts for ArcGIS is the focus of this book.
- Python is an interpreted language, which means it does not need to be compiled. Python scripts are run directly from the source code, making Python easier to work with and more portable than code in compiled languages such as C++ and .NET languages.
- Python scripts can be integrated into ArcGIS as script tools, which work just like the familiar geoprocessing tools.
- Working with Python requires the use of an editor, such as a general-purpose code editor or a specific Python editor. The default Python editor that installs with Python is called IDLE. This book employs PythonWin as the Python editor because it is relatively easy to use on the Windows platform. ArcGIS for Desktop applications also contain the Python window, which works like an interactive interpreter for Python.
- Python is installed as part of a typical ArcGIS for Desktop installation. This includes the IDLE editor by default, but not PythonWin.