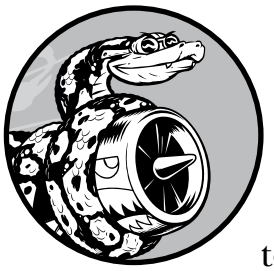
1

GETTING STARTED



In this chapter you'll run your first Python program, *hello_world.py*. First, you'll need to check whether Python is installed on your computer; if it isn't, you'll install it. You'll also install a text editor to work with your Python programs. Text editors recognize Python code and highlight sections as you write, making it easy to understand the structure of your code.

Setting Up Your Programming Environment

Python differs slightly on different operating systems, so you'll need to keep a few considerations in mind. Here, we'll look at the two major versions of Python currently in use and outline the steps to set up Python on your system.

Python 2 and Python 3

Today, two versions of Python are available: Python 2 and the newer Python 3. Every programming language evolves as new ideas and technologies emerge, and the developers of Python have continually made the language more versatile and powerful. Most changes are incremental and hardly noticeable, but in some cases code written for Python 2 may not run properly on systems with Python 3 installed. Throughout this book I'll point out areas of significant difference between Python 2 and Python 3, so whichever version you use, you'll be able to follow the instructions.

If both versions are installed on your system or if you need to install Python, use Python 3. If Python 2 is the only version on your system and you'd rather jump into writing code instead of installing Python, you can start with Python 2. But the sooner you upgrade to using Python 3 the better, so you'll be working with the most recent version.

Running Snippets of Python Code

Python comes with an interpreter that runs in a terminal window, allowing you to try bits of Python without having to save and run an entire program.

Throughout this book, you'll see snippets that look like this:

```
❶ >>> print("Hello Python interpreter!")
Hello Python interpreter!
```

The text in bold is what you'll type in and then execute by pressing ENTER. Most of the examples in the book are small, self-contained programs that you'll run from your editor, because that's how you'll write most of your code. But sometimes basic concepts will be shown in a series of snippets run through a Python terminal session to demonstrate isolated concepts more efficiently. Any time you see the three angle brackets in a code listing ❶, you're looking at the output of a terminal session. We'll try coding in the interpreter for your system in a moment.

Hello World!

A long-held belief in the programming world has been that printing a *Hello world!* message to the screen as your first program in a new language will bring you luck.

In Python, you can write the *Hello World* program in one line:

```
print("Hello world!")
```

Such a simple program serves a very real purpose. If it runs correctly on your system, any Python program you write should work as well. We'll look at writing this program on your particular system in just a moment.

Python on Different Operating Systems

Python is a cross-platform programming language, which means it runs on all the major operating systems. Any Python program you write should run on any modern computer that has Python installed. However, the methods for setting up Python on different operating systems vary slightly.

In this section you'll learn how to set up Python and run the *Hello World* program on your own system. You'll first check whether Python is installed on your system and install it if it's not. Then you'll install a simple text editor and save an empty Python file called *hello_world.py*. Finally, you'll run the *Hello World* program and troubleshoot anything that didn't work. I'll walk you through this process for each operating system, so you'll have a beginner-friendly Python programming environment.

Python on Linux

Linux systems are designed for programming, so Python is already installed on most Linux computers. The people who write and maintain Linux expect you to do your own programming at some point and encourage you to do so. For this reason there's very little you have to install and very few settings you have to change to start programming.

Checking Your Version of Python

Open a terminal window by running the Terminal application on your system (in Ubuntu, you can press CTRL-ALT-T). To find out whether Python is installed, enter **`python`** with a lowercase *p*. You should see output telling you which version of Python is installed and a >>> prompt where you can start entering Python commands, like this:

```
$ python
Python 2.7.6 (default, Mar 22 2014, 22:59:38)
[GCC 4.8.2] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

This output tells you that Python 2.7.6 is currently the default version of Python installed on this computer. When you've seen this output, press CTRL-D or enter `exit()` to leave the Python prompt and return to a terminal prompt.

To check for Python 3, you might have to specify that version; so even if the output displayed Python 2.7 as the default version, try the command `python3`:

```
$ python3
Python 3.5.0 (default, Sep 17 2015, 13:05:18)
[GCC 4.8.4] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

This output means you also have Python 3 installed, so you'll be able to use either version. Whenever you see the `python` command in this book, enter `python3` instead. Most Linux distributions have Python already installed, but if for some reason yours didn't or if your system came with Python 2 and you want to install Python 3, refer to Appendix A.

Installing a Text Editor

Geany is a simple text editor: it's easy to install, will let you run almost all your programs directly from the editor instead of through a terminal, uses syntax highlighting to color your code, and runs your code in a terminal window so you'll get used to using terminals. Appendix B provides information on other text editors, but I recommend using Geany unless you have a good reason to use a different editor.

You can install Geany in one line on most Linux systems:

```
$ sudo apt-get install geany
```

If this doesn't work, see the instructions at *http://geany.org/Download/ThirdPartyPackages/*.

Running the Hello World Program

To start your first program, open Geany. Press the Super key (often called the Windows key) and search for Geany on your system. Make a shortcut by dragging the icon to your taskbar or desktop. Then make a folder somewhere on your system for your projects and call it *python_work*. (It's best to use lowercase letters and underscores for spaces in file and folder names because these are Python naming conventions.) Go back to Geany and save an empty Python file (**File ▸ Save As**) called *hello_world.py* in your *python_work* folder. The extension *.py* tells Geany your file will contain a Python program. It also tells Geany how to run your program and highlight the text in a helpful way.

After you've saved your file, enter the following line:

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

If multiple versions of Python are installed on your system, you need to make sure Geany is configured to use the correct version. Go to **Build ▸ Set Build Commands**. You should see the words *Compile* and *Execute* with a command next to each. Geany assumes the correct command for each is `python`, but if your system uses the `python3` command, you'll need to change this.

If the command `python3` worked in a terminal session, change the Compile and Execute commands so Geany will use the Python 3 interpreter. Your Compile command should look like this:

```
python3 -m py_compile "%f"
```

You need to type this command exactly as it's shown. Make sure the spaces and capitalization match what is shown here.

Your Execute command should look like this:

```
python3 "%f"
```

Again, make sure the spacing and capitalization match what is shown here. Figure 1-1 shows how these commands should look in Geany's configuration menu.

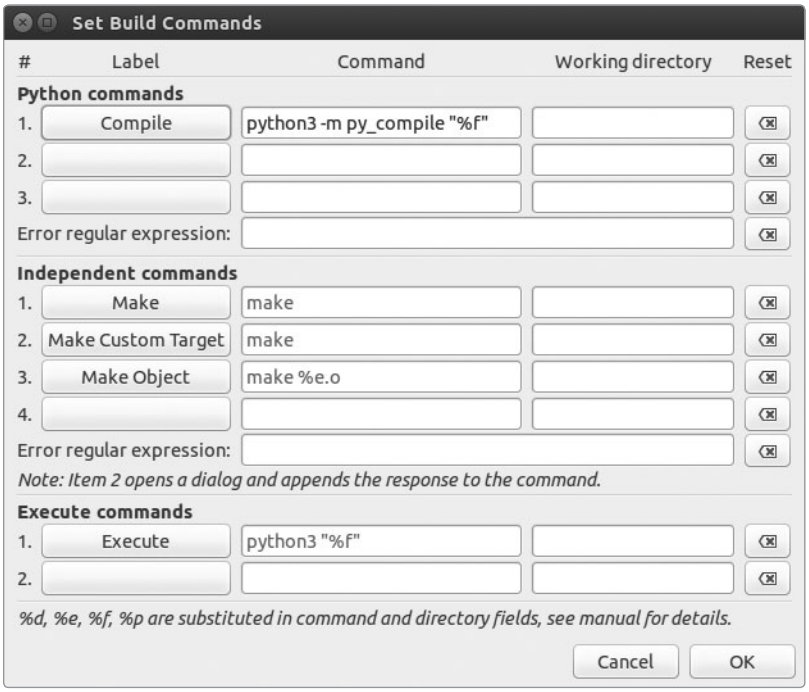


Figure 1-1: Here, Geany is configured to use Python 3 on Linux.

Now run *hello_world.py* by selecting **Build ▸ Execute** in the menu, by clicking the Execute icon (which shows a set of gears), or by pressing F5. A terminal window should pop up with the following output:

```
Hello Python world!

------------------
(program exited with code: 0)
Press return to continue
```

If you don't see this, check every character on the line you entered. Did you accidentally capitalize `print`? Did you forget one or both of the quotation marks or parentheses? Programming languages expect very specific syntax, and if you don't provide that, you'll get errors. If you can't get the program to run, see "Troubleshooting Installation Issues" on page 15.

Running Python in a Terminal Session

You can try running snippets of Python code by opening a terminal and typing `python` or `python3`, as you did when checking your version. Do this again, but this time enter the following line in the terminal session:

```
>>> print("Hello Python interpreter!")
Hello Python interpreter!
>>>
```

You should see your message printed directly in the current terminal window. Remember that you can close the Python interpreter by pressing CTRL-D or by typing the command `exit()`.

Python on OS X

Python is already installed on most OS X systems. Once you know Python is installed, you'll need to install a text editor and make sure it's configured correctly.

Checking Whether Python Is Installed

Open a terminal window by going to **Applications ▸ Utilities ▸ Terminal**. You can also press COMMAND-spacebar, type **`terminal`**, and then press ENTER. To find out whether Python is installed, enter **`python`** with a lowercase *p*. You should see output telling you which version of Python is installed on your system and a >>> prompt where you can start entering Python commands, like this:

```
$ python
Python 2.7.5 (default, Mar  9 2014, 22:15:05)
[GCC  4.2.1 Compatible Apple LLVM 5.0 (clang-500.0.68)] on darwin
Type "help", "copyright", "credits", or "license" for more information.
>>>
```

This output tells you that Python 2.7.5 is currently the default version installed on this computer. When you've seen this output, press CTRL-D or enter `exit()` to leave the Python prompt and return to a terminal prompt.

To check for Python 3, try the command `python3`. You might get an error message, but if the output shows you have Python 3 installed, you'll be able to use Python 3 without having to install it. If `python3` works on your system, whenever you see the `python` command in this book, make sure you use `python3` instead. If for some reason your system didn't come with Python or if you only have Python 2 and you want to install Python 3 now, see Appendix A.

Running Python in a Terminal Session

You can try running snippets of Python code by opening a terminal and typing `python` or `python3`, as you did when checking your version. Do this again, but this time enter the following line in the terminal session:

```
>>> print("Hello Python interpreter!")
Hello Python interpreter!
>>>
```

You should see your message printed directly in the current terminal window. Remember that you can close the Python interpreter by pressing CTRL-D or by typing the command `exit()`.

Installing a Text Editor

Sublime Text is a simple text editor: it's easy to install on OS X, will let you run almost all of your programs directly from the editor instead of through a terminal, uses syntax highlighting to color your code, and runs your code in a terminal session embedded in the Sublime Text window to make it easy to see the output. Appendix B provides information on other text editors, but I recommend using Sublime Text unless you have a good reason to use a different editor.

You can download an installer for Sublime Text from *http://sublimetext.com/3*. Click the download link and look for an installer for OS X. Sublime Text has a very liberal licensing policy: you can use the editor for free as long as you want, but the author requests that you purchase a license if you like it and want continual use. After the installer has been downloaded, open it and then drag the Sublime Text icon into your *Applications* folder.

Configuring Sublime Text for Python 3

If you use a command other than `python` to start a Python terminal session, you'll need to configure Sublime Text so it knows where to find the correct version of Python on your system. Issue the following command to find out the full path to your Python interpreter:

```
$ type -a python3
python3 is /usr/local/bin/python3
```

Now open Sublime Text, and go to **Tools ▸ Build System ▸ New Build System**, which will open a new configuration file for you. Delete what you see and enter the following:

Python3.sublime-build

```
{
    "cmd": ["/usr/local/bin/python3", "-u", "$file"],
}
```

This code tells Sublime Text to use your system's `python3` command when running the currently open file. Make sure you use the path you found when issuing the command `type -a python3` in the previous step. Save the file as *Python3.sublime-build* in the default directory that Sublime Text opens when you choose Save.

Running the Hello World Program

To start your first program, launch Sublime Text by opening the *Applications* folder and double-clicking the Sublime Text icon. You can also press COMMAND-spacebar and enter **`sublime text`** in the search bar that pops up.

Make a folder called *python_work* somewhere on your system for your projects. (It's best to use lowercase letters and underscores for spaces in file and folder names, because these are Python naming conventions.) Save an empty Python file (**File ▸ Save As**) called *hello_world.py* in your *python_work* folder. The extension *.py* tells Sublime Text that your file will contain a Python program and tells it how to run your program and highlight the text in a helpful way.

After you've saved your file, enter the following line:

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

If the command `python` works on your system, you can run your program by selecting **Tools ▸ Build** in the menu or by pressing CTRL-B. If you configured Sublime Text to use a command other than `python`, select **Tools ▸ Build System** and then select **Python 3**. This sets Python 3 as the default version of Python, and you'll be able to select **Tools ▸ Build** or just press COMMAND-B to run your programs from now on.

A terminal screen should appear at the bottom of the Sublime Text window, showing the following output:

```
Hello Python world!
[Finished in 0.1s]
```

If you don't see this, check every character on the line you entered. Did you accidentally capitalize `print`? Did you forget one or both of the quotation marks or parentheses? Programming languages expect very specific syntax, and if you don't provide that, you'll get errors. If you can't get the program to run, see "Troubleshooting Installation Issues" on page 15.

Python on Windows

Windows doesn't always come with Python, so you'll probably need to download and install it, and then download and install a text editor.

Installing Python

First, check whether Python is installed on your system. Open a command window by entering **command** into the Start menu or by holding down the SHIFT key while right-clicking on your desktop and selecting **Open command window here**. In the terminal window, enter **python** in lowercase. If you get a Python prompt (>>>), Python is installed on your system. However, you'll probably see an error message telling you that `python` is not a recognized command.

In that case, download a Python installer for Windows. Go to *http://python.org/downloads/*. You should see two buttons, one for downloading Python 3 and one for downloading Python 2. Click the Python 3 button, which should automatically start downloading the correct installer for your system. After you've downloaded the file, run the installer. Make sure you check the option Add Python to PATH, which will make it easier to configure your system correctly. Figure 1-2 shows this option checked.

Figure 1-2: Make sure you check the box labeled Add Python to PATH.

Starting a Python Terminal Session

Setting up your text editor will be straightforward if you first set up your system to run Python in a terminal session. Open a command window and enter **python** in lowercase. If you get a Python prompt (>>>), Windows has found the version of Python you just installed:

```
C:\> python
Python 3.5.0 (v3.5.0:374f501f4567, Sep 13 2015, 22:15:05) [MSC v.1900 32 bit
(Intel)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

If this worked, you can move on to the next section, "Running Python in a Terminal Session."

However, you may see output that looks more like this:

```
C:\> python
'python' is not recognized as an internal or external command, operable
program or batch file.
```

In this case you need to tell Windows how to find the Python version you just installed. Your system's `python` command is usually saved in your *C* drive, so open Windows Explorer and open your *C* drive. Look for a folder starting with the name *Python*, open that folder, and find the *python* file (in lowercase). For example, I have a *Python35* folder with a file named *python* inside it, so the path to the `python` command on my system is *C:\Python35\python*. Otherwise, enter **`python`** into the search box in Windows Explorer to show you exactly where the `python` command is stored on your system.

When you think you know the path, test it by entering that path into a terminal window. Open a command window and enter the full path you just found:

```
C:\> C:\Python35\python
Python 3.5.0 (v3.5.0:374f501f4567, Sep 13 2015, 22:15:05) [MSC v.1900 32 bit
(Intel)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

If this worked, you know how to access Python on your system.

Running Python in a Terminal Session

Enter the following line in your Python session, and make sure you see the output *Hello Python world!*

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

Any time you want to run a snippet of Python code, open a command window and start a Python terminal session. To close the terminal session, press CTRL-Z and then press ENTER, or enter the command `exit()`.

Installing a Text Editor

Geany is a simple text editor: it's easy to install, will let you run almost all of your programs directly from the editor instead of through a terminal, uses syntax highlighting to color your code, and runs your code in a terminal window so you'll get used to using terminals. Appendix B provides information on other text editors, but I recommend using Geany unless you have a good reason to use a different editor.

You can download a Windows installer for Geany from *http://geany.org/*. Click **Releases** under the Download menu, and look for the *geany-1.25_setup.exe* installer or something similar. Run the installer and accept all the defaults.

To start your first program, open Geany: press the Windows key and search for Geany on your system. You should make a shortcut by dragging the icon to your taskbar or desktop. Make a folder called *python_work* somewhere on your system for your projects. (It's best to use lowercase letters and underscores for spaces in file and folder names, because these are Python naming conventions.) Go back to Geany and save an empty Python file (**File ▸ Save As**) called *hello_world.py* in your *python_work* folder. The extension *.py* tells Geany that your file will contain a Python program. It also tells Geany how to run your program and to highlight the text in a helpful way.

After you've saved your file, type the following line:

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

If the command `python` worked on your system, you won't have to configure Geany; skip the next section and move on to "Running the Hello World Program" on page 14. If you needed to enter a path like *C:\Python35\python* to start a Python interpreter, follow the directions in the next section to configure Geany for your system.

Configuring Geany

To configure Geany, go to **Build ▸ Set Build Commands**. You should see the words *Compile* and *Execute* with a command next to each. The Compile and Execute commands start with `python` in lowercase, but Geany doesn't know where your system stored the `python` command. You need to add the path you used in the terminal session.

In the Compile and Execute commands, add the drive your `python` command is on and the folder where the `python` command is stored. Your Compile command should look something like this:

```
C:\Python35\python -m py_compile "%f"
```

Your path might be a little different, but make sure the spaces and capitalization match what is shown here.

Your Execute command should look something like this:

```
C:\Python35\python "%f"
```

Again, make sure the spacing and capitalization in your Execute command matches what is shown here. Figure 1-3 shows how these commands should look in Geany's configuration menu.

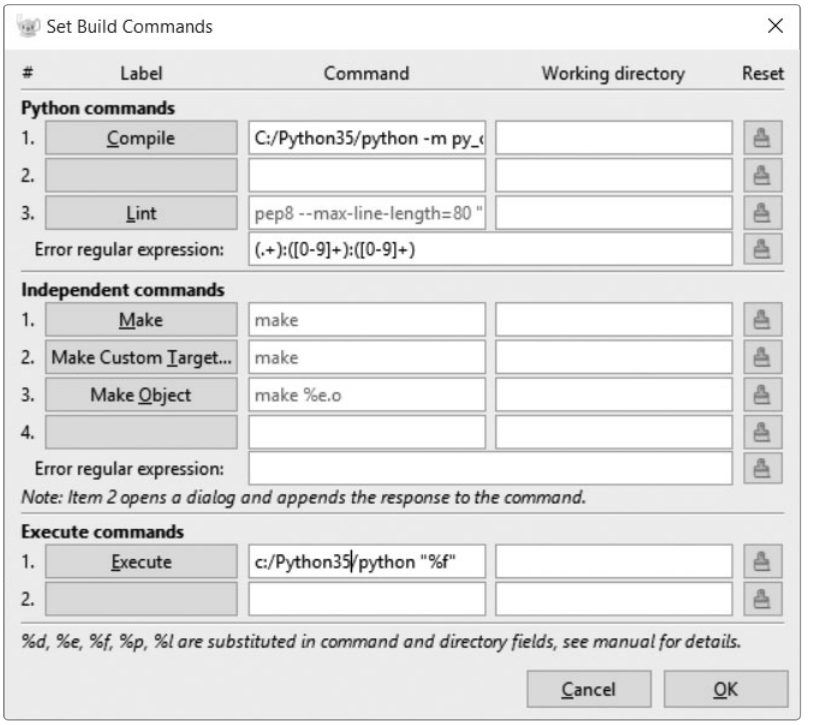


Figure 1-3: Here, Geany is configured to use Python 3 on Windows.

After you've set these commands correctly, click **OK**.

Running the Hello World Program

You should now be able to run your program successfully. Run *hello_world.py* by selecting **Build ▸ Execute** in the menu, by clicking the Execute icon (which shows a set of gears), or by pressing F5. A terminal window should pop up with the following output:

```
Hello Python world!

------------------
(program exited with code: 0)
Press return to continue
```

If you don't see this, check every character on the line you entered. Did you accidentally capitalize `print`? Did you forget one or both of the quotation marks or parentheses? Programming languages expect very specific syntax, and if you don't provide that, you'll get errors. If you can't get the program to run, see the next section for help.

Troubleshooting Installation Issues

Hopefully, setting up your programming environment was successful, but if you've been unable to run *hello_world.py*, here are a few remedies you can try:

- When a program contains a significant error, Python displays a *traceback*. Python looks through the file and tries to report the problem. The traceback might give you a clue as to what issue is preventing the program from running.
- Step away from your computer, take a short break, and then try again. Remember that syntax is very important in programming, so even a missing colon, a mismatched quotation mark, or mismatched parentheses can prevent a program from running properly. Reread the relevant parts of this chapter, look over what you've done, and see if you can find the mistake.
- Start over again. You probably don't need to uninstall anything, but it might make sense to delete your *hello_world.py* file and create it again from scratch.
- Ask someone else to follow the steps in this chapter, on your computer or a different one, and watch what they do carefully. You might have missed one small step that someone else happens to catch.
- Find someone who knows Python and ask them to help you get set up. If you ask around, you might find that you know someone who uses Python.
- The setup instructions in this chapter are also available online, through *https://www.nostarch.com/pythoncrashcourse/*. The online version of these instructions may work better for you.
- Ask for help online. Appendix C provides a number of resources and areas online, like forums and live chat sites, where you can ask for solutions from people who've already worked through the issue you're currently facing.

Don't worry about bothering experienced programmers. Every programmer has been stuck at some point, and most programmers are happy to help you set up your system correctly. As long as you can state clearly what you're trying to do, what you've already tried, and the results you're getting, there's a good chance someone will be able to help you. As mentioned in the Introduction, the Python community is very beginner friendly.

Python should run well on any modern computer, so find a way to ask for help if you're having trouble so far. Early issues can be frustrating, but they're well worth sorting out. Once you get *hello_world.py* running, you can start to learn Python, and your programming work will become more interesting and satisfying.

Running Python Programs from a Terminal

Most of the programs you write in your text editor you'll run directly from the editor, but sometimes it's useful to run programs from a terminal instead. For example, you might want to run an existing program without opening it for editing.

You can do this on any system with Python installed if you know how to access the directory where you've stored your program file. To try this, make sure you've saved the *hello_world.py* file in the *python_work* folder on your desktop.

On Linux and OS X

Running a Python program from a terminal session is the same on Linux and OS X. The terminal command `cd`, for *change directory*, is used to navigate through your file system in a terminal session. The command `ls`, for *list*, shows you all the nonhidden files that exist in the current directory.

Open a new terminal window and issue the following commands to run *hello_world.py*:

```
❶ ~$ cd Desktop/python_work/
❷ ~/Desktop/python_work$ ls
  hello_world.py
❸ ~/Desktop/python_work$ python hello_world.py
  Hello Python world!
```

At ❶ we use the `cd` command to navigate to the *python_work* folder, which is in the *Desktop* folder. Next, we use the `ls` command to make sure *hello_world.py* is in this folder ❷. Then, we run the file using the command `python hello_world.py` ❸.

It's that simple. You just use the `python` (or `python3`) command to run Python programs.

On Windows

The terminal command `cd`, for *change directory*, is used to navigate through your file system in a command window. The command `dir`, for *directory*, shows you all the files that exist in the current directory.

Open a new terminal window and issue the following commands to run *hello_world.py*:

```
❶ C:\> cd Desktop\python_work
❷ C:\Desktop\python_work> dir
  hello_world.py
❸ C:\Desktop\python_work> python hello_world.py
  Hello Python world!
```

At ❶ we use the `cd` command to navigate to the *python_work* folder, which is in the *Desktop* folder. Next, we use the `dir` command to make sure *hello_world.py* is in this folder ❷. Then, we run the file using the command `python hello_world.py` ❸.

If you haven't configured your system to use the simple command `python`, you may need to use the longer version of this command:

```
C:\$ cd Desktop\python_work
C:\Desktop\python_work$ dir
hello_world.py
C:\Desktop\python_work$ C:\Python35\python hello_world.py
Hello Python world!
```

Most of your programs will run fine directly from your editor, but as your work becomes more complex, you might write programs that you'll need to run from a terminal.

TRY IT YOURSELF

The exercises in this chapter are exploratory in nature. Starting in Chapter 2, the challenges you'll solve will be based on what you've learned.

1-1. python.org: Explore the Python home page (*http://python.org/*) to find topics that interest you. As you become familiar with Python, different parts of the site will be more useful to you.

1-2. Hello World Typos: Open the *hello_world.py* file you just created. Make a typo somewhere in the line and run the program again. Can you make a typo that generates an error? Can you make sense of the error message? Can you make a typo that doesn't generate an error? Why do you think it didn't make an error?

1-3. Infinite Skills: If you had infinite programming skills, what would you build? You're about to learn how to program. If you have an end goal in mind, you'll have an immediate use for your new skills; now is a great time to draft descriptions of what you'd like to create. It's a good habit to keep an "ideas" notebook that you can refer to whenever you want to start a new project. Take a few minutes now to describe three programs you'd like to create.

Summary

In this chapter you learned a bit about Python in general, and you installed Python to your system if it wasn't already there. You also installed a text editor to make it easier to write Python code. You learned to run snippets of Python code in a terminal session, and you ran your first actual program, *hello_world.py*. You probably learned a bit about troubleshooting as well.

In the next chapter you'll learn about the different kinds of data you can work with in your Python programs, and you'll learn to use variables as well.