10 Fun Python Coding Projects

Python® FOR KIDS

- Learn a Pro Coding Language
- Create Games and Secret Messages
- Customize Your Projects

Brendan Scott
Creator of Python4Kids blog

FOR DUMMIES®
A Wiley Brand

Introduction

Hi! Welcome to the book. You're going on a tour of all things Python. If you join me and code along with the projects, you'll have your basic Python programming wings by the end of the book.

Everything in this book you need to know by *doing* — typing in the code or, better yet, thinking up the code before reading what I've done.

About This Book

This book walks you through all the parts you have to know about Python programming. You get examples. I talk about planning programs. And I help link you with the broader Python community so that you can head out there after mastering the projects in this book.

Conventions Used

Keep these things in mind while you read:

- Sometimes words are in *italics* and then I explain the words. Here is an example: "The objects in the list are called *elements*." When you see this sentence, you know to keep your eyes peeled for a definition. (Elements are the objects in a list.)
- Python code is written in a different font from the other text. Sometimes it's inline with the text and looks like this: `print('Hello World!')`.
- Sometimes it's a separate block of text, like this:

```
print('Hello World!')
```

- Some code blocks have a >>> at the front of some lines. I'm showing you what happens when you're using an interactive Python prompt. You need to type the code that follows the >>> in this book into the Python console that's running on your computer to see what happens:

```
>>> my_message = "Hello World!"
>>> print(my_message)
```

- The number of spaces at the front of each line of code is important. The length of your lines isn't (technically) significant, but Python style guidelines suggest lines with no more than 79 *characters* (letters, numbers, spaces, or punctuation marks). This book isn't as wide as your screen; it only lets me show 69 characters in a line. I've *broken* (split) some lines of code in the book. I split them to make sure that the code works and prints the right way. Be careful when you type them in! It's not always clear how many spaces are at the front of the broken line.
- I split lines two ways.
    - The first is *implicitly.* Basically, you can split the code in between any pair of parentheses at a comma. Python still sees it as a single line. The second and later parts of a split line should be indented to where the parentheses opened. Here's an example from the code in Project 9:

```
values = (e.first_name, e.family_name,
          e.date_of_birth, e.email_address)
```

      Even though you type this as two separate lines, Python sees it as a single line. (Think of it as one long line.) Type this code as you see it, pressing Enter at the end of each line and typing spaces at the start of each line so that the first character in the line is in the right place.

- The second way to split a line is *explicitly* with the backslash character: \ (not /). Here's an example from Project 9:

```
raw_input_prompt = "Press: 1 for training,"+\
                   " 2 for testing, 3 to quit.\n"
```

  You type these as two separate lines, with the \ at the end of the first line. However, Python sees it as a single line.

- When using the Python interpreter in Projects 2 and 3 only, each new line starts with either … or >>>. If you don't see these in the code in the book, then the previous line is meant to be typed in as one long line. For example, the following code is from Project 2:

```
>>> my_second_message = 'This name is a little long. Ideally, try to
  keep the name short, but not too short.'
```

  This code doesn't have … or >>> at the start of the second or third line. This means you're supposed to type it all in before pressing Enter. Only press Enter after typing `too short.'` at the end, *not* after typing `little long.` and `but not`.

- Sometimes the output on your computer may look a little different from what you see in this book. For example, when you run a program in later projects, you might see a restart line. On my screen, all the following text is on a single line:

```
>>> ================================ RESTART
  ================================
```

- In Project 4 you see how to automatically indent your code. Until then, each time you need to indent code, do it by pressing the spacebar four times before you add your code. If you have to indent code two levels, press the spacebar eight times (that's two levels of indent by four spaces per level) before typing your code, and so on. You need to do this for each line of the indented code.

- When I'm explaining how code works, I often provide a *code template* — an outline of how to use the code. A sample template is: `help([object name])`. In this template, the keyword is `help`, and it needs to be followed by a pair of parentheses. The square brackets indicate something which is optional. The italics mean you need to fill in. Everything not in italics, type it just like it looks. Using this template, the code `help(help)` works (it gets help on the `help` keyword), and so does `help()`, with nothing inside the parentheses (since [*object name*] was optional).
- Web addresses (URLs) and programming code are in `monofont`. If you're reading this book on a device connected to the Internet, you can click the address to visit that website. Try it: www.dummies.com.
- Sometimes you need to choose something from a menu. I'm not talking about a burger and fries. I mean actions. For example, I might ask you to choose File ⇒ New File. This means that you go to the File menu and choose New File from it.
- The word Ctrl means the Ctrl key on your keyboard. Ctrl+A means that you press the key marked Ctrl while you press the A key. All at the same time. Then release both keys. If you're using a Mac, your keyboard has a control key — use it. Ctrl-A means press the control key down and press the A key. Then release both keys. Don't use the option or command keys.
- If you're using a Mac, assume that when I say Enter key, it means the Return key on your keyboard.

Foolish Assumptions

I've tried not to make too many assumptions about you in this book. In order to use this book, you need to be able

to turn on your computer and navigate the Start Menu (on Windows). To install Python you will need administrator access for the computer you're installing it on.

Learning anything is slow going to start. You are going to need a bit of determination to make it through the book. Hang in there.

Icons Used in This Book

The Warning icon tells you to watch out! This information may save you headaches. In some cases, you could lose data if you don't heed the warnings.

You're gonna use this information for a long, long time. Commit it to long-term gray matter.

The Tip icon marks shortcuts that make programming easier.

Coding Connections icons mark information that applies not just to Python, but to coding in general.

Beyond the Book

You can find a bunch more information outside this book. Check out:

- **Cheat Sheet:** This book has an online cheat sheet at www.dummies.com/cheatsheet/pythonforkids. The Cheat

Sheet has a list of Python keywords, common built-ins, and selected functions from the standard library. Use it as a quick reference when you're coding.

- **Dummies.com online articles and bonus projects:** In addition to the projects in this book, there are some bonus projects online. You can get them from www.dummies.com/extras/pythonforkids.
- python4kids.brendanscott.com: Visit my Python for Kids blog. Many of the projects in this book started out there. The blog has a dedicated blog entry for each of the projects and has a heap of other things you can try, too. If you've got feedback, you can leave it on the blog page that applies.

Where to Go from Here

Right now you should go to Project 1 to read more about what this language can do and to install it. Before you move on to Project 2, make sure you know about Ctrl+C. Then you're ready to write your first Python program! Move in and out of the projects as you like. The code in each project stands on its own. Be careful though — even though they don't use *code* from earlier projects, they often use *concepts* introduced earlier.

Week 1

Slithering into Python

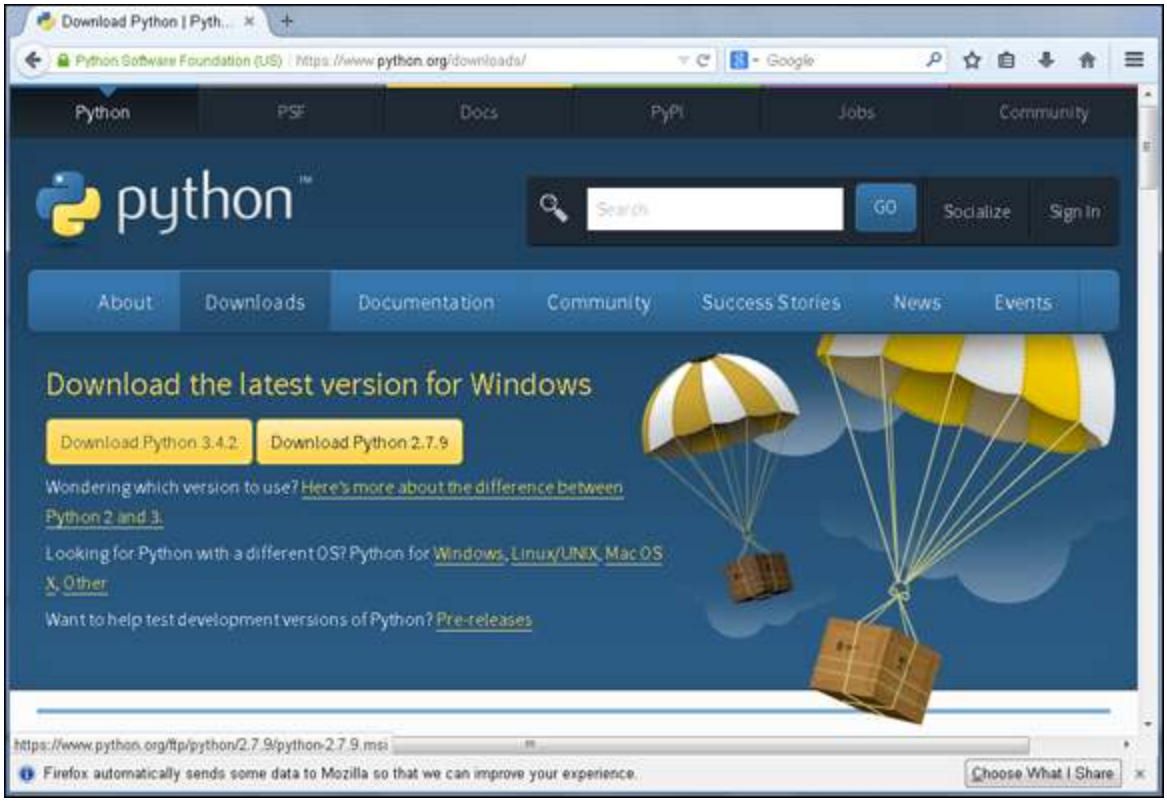
This week you're …

- [ ] Getting Started with Python
- [ ] Building Hello World!

PROJECT 1

Getting Started with Python

In this project, you're introduced to Python: where it's used and what it's used for. I explain the two current versions of Python. This book is focused on Python 2.7 and I explain why. With my help during this chapter, you install a copy of Python 2.7 (if you don't already have it installed) and fire it up. I also tell you how to stop once you've started.



This project also shows you how to get Python's documentation, both built in to Python and online. I give you ways to search online for answers to your Python

problems, just in case you've never searched the Internet before. You also read about the Python community, which is one place you can go for help or new ideas. All that, but no actual programming? No worries. Actual programming starts in Project 2.

TL;DR: If you've already installed Python, and you can start and stop it, then skip to Project 2.

Python and Why It's Wonderful

Python is a programming language written by a person called Guido van Rossum in the 1990s. *Programming* languages allow you to control what a computer does and the way it does it.

Some of the things that make Python totes awesome (also known as "really helpful and lots of fun") are:

- **Python code is easy to read and understand.** In fact, I think Python's code is sublime and beautiful. (Hey, that's just my opinion.) Its beauty means you don't even notice the way Python makes complex things simple. This makes Python easy to learn, which makes it perfect for kids.
- **Python is productive.** It makes tough tasks simple. Almost any programming task is easier with Python than it is with other programming languages. Computer types call this *RAD* (for Rapid Application Development).
- **Python is dangerous.** It has a lot of power. But with great power comes great responsibility. (Remember Spider-Man?) And you'll have to use your powers for

good, not evil. (If you want to use them for evil, you have to stop reading now.)

- **Python is a scripting language.** The programs are fed into Python's interpreter, which runs them directly, so there's no compiling (which is the case for some other languages). It is faster and easier to get feedback on your Python code (finding errors, for example). Python means you complete and *execute* (run) your programs faster and that makes programming fun!
- **Python is cross platform.** Almost anyone can use it, no matter what computer operating system they have. You can run pretty much any Python program on Windows, Mac, and Linux personal computers and from large servers through to tiny computers like the Raspberry Pi. (A Pi-specific project is waiting at `dummies.com/go/pythonforkids` for you.) You can even run Python programs on Android and iOS tablets. I even used my Android tablet to code some of the early projects in this book.
- **Python uses dynamic typing for its variables.** This may not mean much to you if you've never done programming before. Dynamically typed variables make programming easier because they let you just start using a variable, rather than first explaining to the computer what the variable is supposed to be.
- **Python gets lots of help from *third-party modules*.** This means that a lot of other people (third parties) have written libraries. A *library* is a bunch of code for doing something specific. This makes your work easier because you don't have to start from scratch every time you write a new program; sometimes you can use the libraries already written.

The Minecraft project online uses a third-party library to change a Minecraft game on a Raspberry Pi.

- **Python is free software.** This means that the license terms for Python respect your freedom. I think this is pretty important. You can download and run Python without paying any money, and any program that you write with it is yours to use and share any way you want. It also means that the Python *source code* (the human-readable form of what the computer runs) is available so, when you're feeling brave enough, you can look at how the Python developers wrote their code. (It's written in a different programming language, though, d'oh!)

Pythons aren't just snakes

The Python programming language is named after a comedy group called Monty Python, not the reptile. Monty Python was active mainly in the 1970s. (40 years ago! Forever and ever, right?) They had a British television show called *Monty Python's Flying Circus* and have made lots of movies, the most notable of which is *Monty Python and the Holy Grail*.

Who's Using Python

Python is used just about everywhere.

- **In space:** The International Space Station's Robonaut 2 robot uses Python for its central command system. Python is planned for use in a European mission to Mars in 2020 to collect soil samples.
- **In particle physics laboratories:** Python helps understand the data analysis from some atom smashing experiments at the CERN Large Hadron Collider.

- **In astronomy:** The MeerKat Radio telescope array (the largest radio telescope in the Southern Hemisphere) uses Python for its control and monitoring systems.
- **In movie studios:** Industrial Light and Magic (*Star Wars* geniuses) uses Python to automate its movie production processes. Side Effects Software's computer-generated imagery program Houdini uses Python for its programming interface and to script the engine.
- **In games:** Activision uses Python for building games, testing, and analyzing stuff. They even use Python to find people cheating by boosting each other.
- **In the music industry:** Spotify music streaming service uses Python to send you music.
- **In the video industry:** Netflix uses Python to make sure movies play *(stream)* without stopping. Python is used a lot for YouTube.
- **In Internet search:** Google used Python all over in its early development phase.
- **In medicine:** The Nodality company uses Python to handle information that they use to search for a cure to cancer.
- **In your OS (admin-ing your datas):** Operating systems like Linux and Mac OSX use Python for some of their administrative functions.
- **In your doorbell:** Rupa Dachere and Akkana Peck say that you can automate your home with Python, hooking up sensors to your house. With it you can, for example, open and close the curtains or automatically turn on lights when you come in the room.

I could go on. The point is that Python will apply to whatever you're interested in, no matter what it is.

Making Things with Python

You do these things while you work through this book:

- Make a math trainer for practicing your times tables.
- Make a simple *encryption* (a secret code) program.
- Use Python on a Raspberry Pi to work with and modify your Minecraft world. (See `www.dummies.com/go/pythonforkids` for that project.)

When you've honed your mad skills and are ready to move on, there'll be other things you can do:

- Using `Tkinter` (or other widget sets), you can write user applications that use graphics rather than just text to interact with the user.
- You can extend other programs like Blender (a 3D modeling program), GIMP (a 2D photo-retouching program), and LibreOffice (office programs), among many others by writing custom scripts. I had to fix some 3D models I was making in Blender. It would've taken forever to do by hand, so I wrote a Python script to do it quickly.
- You can write games with graphics using `Tkinter` or the Pygame or Kivy libraries. The games in this book are text only.
- You can use the matplotlib library to draw complex graphs for your math or science courses.

- Using the openCV library, you can experiment with computer vision. People who are into robotics use it to help their robots see and grab things and to avoid obstacles when moving.

Whatever you want it to do, there's a good chance someone has already written code to do it or to help you do it yourself.

Understanding This Book's Pedagogical Approach

That title is just to impress your parents. (I hope they're not reading this part. But look: If they don't see that title, tell them that this book *has* a pedagogical approach — *ped-uh-goj-i-cul.* It means education or teaching.)

The point of this book is to give you a chunk of information about the programming concepts that you need to program in Python. The book is for you — a kid who can learn Python.

I am thorough

Thorough, yes. Will I include everything? No way. Many aspects of Python have lots of options. If I took you through all the possibilities of each option, you'd fall asleep (or throw this book out the window). If you do either of those things, then you won't be learning.

As you read, remember that I've tried to introduce you to enough information so you can be a Python programmer, but not so much that you'd need superhuman powers to get through it. Expand on your own using the documentation and help.

You start pretty slowly with *core* (main) principles. If you think things aren't going fast enough, skip ahead! The examples are generally self contained. This means that you end up with many smaller projects rather than a few larger projects. I did that on purpose so you can do the projects in any order you like. You've got enough people telling you what to do. You can go where you want to in this book.

The earlier projects use plain English, rather than technical words. As you go through the book, you'll see more jargon. You'll also get less hand holding. You'll have to work harder the further you get through the book.

If you're dying to know more

If you really want to know everything about one particular part of Python, first try Python's help function, its introspection features, and its online documentation. Each of these is introduced later in this project. You can also try some of the reference manuals. They're different from instructional books (which is what this is). This book: filled with wildly interesting information and humor. Reference manuals: filled with boring (but useful and sometimes important) details. You can also try one of the Python *cookbooks* (a book with coding recipes that solve specific problems).

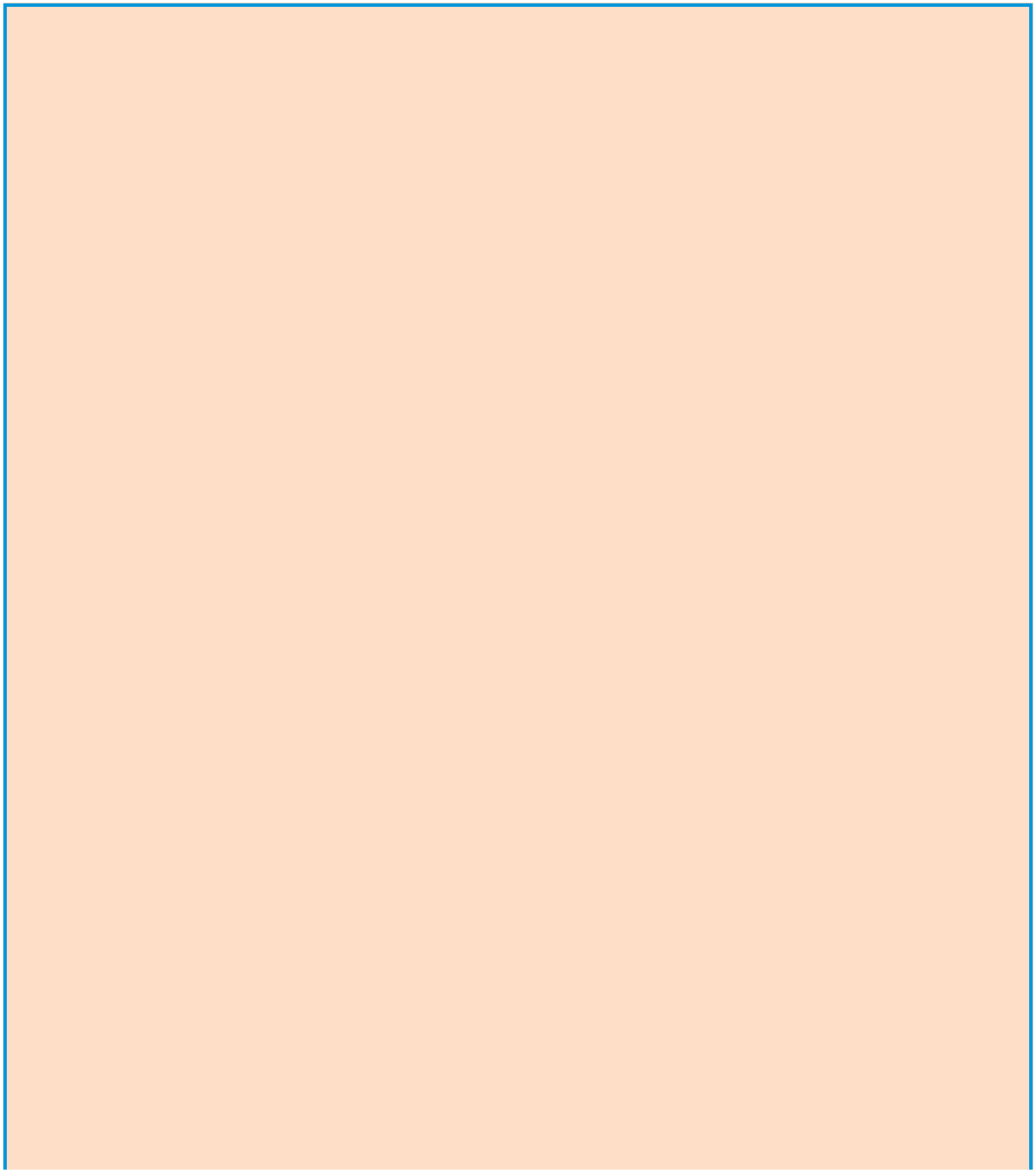
I walk you through programming

The projects try to show you realistic programming without boring you to tears. When you write your own code to solve your own problems, you'll need methods and approaches *(tools)* that get the job done. I teach you these tools by walking you through each project, step by step. Try every step — don't skip any.

If you want to *run* the working program, skip to the end of the project and cut and paste. If you want to *learn* Python, then consider each project a journey, not a destination. Work through the projects with me and type them in yourself.

I am practical

Hopefully, you can use these projects for something in your everyday life. Maybe they'll help with your homework or let you store private notes. I start small and dream big. Please dream big with me while you're getting the concepts in the first projects.



Why this book uses Python 2.7

The Python language is changing from Python 2.7 (sometimes called Python 2) to a new version, called Python 3. This change is taking years to complete. Python 3 is very similar to Python 2, but the two are incompatible — they don't work together. If you write a script that works in Python 2, it isn't guaranteed to work in Python 3 (and vice versa).

Deciding whether to use Python 2 or Python 3 in this book was difficult. I focus on Python 2 (specifically Python 2.7) mainly because I think it still has the best third-party libraries. For example, the Minecraft Pi project at `www.dummies.com/go/pythonforkids` needs Python 2. If you want to do something productive using Python, and you need to use a third-party module to do it, you'll probably need Python 2. Third-party modules that work with Python 3 often have a version that works with Python 2 support, but the reverse isn't true. This will change over the next couple of years.

The way Python 3 is different from Python 2 is mainly in advanced features. Because of that, even if I based the book on Python 3, I wouldn't cover most of what's new about Python 3.

Finally, Python 2 is part of the *standard install* on Mac OSX computers, which means that this book should be usable by Mac owners without your having to download and install anything. Most Linux comes with Python installed (but make sure you have Python version 2.7).

Install Python on Mac OSX

To find and start Python on Mac OSX computers, follow these steps:

1. Press Cmd+spacebar to open Spotlight.
2. Type the word `terminal`.

   Or, from the Finder, select Finder ⇒ Go ⇒ Utilities ⇒ Terminal.

   The Terminal window opens.
3. In the terminal, type `python`.

   The Python interpreter that's built in to Mac OSX opens.

Install Python on Windows

Unfortunately, Python doesn't come on Windows. If you're running Windows, then you need to download and install Python by following the instructions here. Installing Python on Windows isn't difficult. If you can download a file from a website, you have the skills to install Python.

Fortunately, the Python Foundation (the peeps who guide the development of Python) makes installable files available from its website.

When I did the installation, I found that Firefox and Internet Explorer responded differently to the Python download website, so the instructions are based on which of these browsers to use. If you use a whole other browser altogether, try the Internet Explorer instructions.

With Firefox

To install Python on a Windows machine with Firefox, follow these steps:

1. Visit `www.python.org/downloads`.
2. Click the button that says Download Python 2.7.9.

   Or, if it's there, click a more recent version number that starts with 2.7.

   Clicking this button automatically downloads and saves an `msi` file for you. If not, try the instructions for Internet Explorer. See Figure 1-1.
3. When the download's complete, click the icon for Firefox's download tool.
4. Click the file called `python-2.7.9.msi` (or the more recent version, if you downloaded one).

Python 2.7.9 installs on your computer.

Figure 1-1: Download Python with Firefox.

With Internet Explorer

To install Python on a Windows machine with Internet Explorer, follow these steps:

1. Visit `www.python.org/downloads`.
2. From the menu bar, select Downloads ⇒ Windows.
   You can see the menu options in Figure 1-2.
3. Scroll down to the heading Python 2.7.9-2014-12-10.
   Or scroll to a more recent version, which starts with Python 2.7, if one is available.

4. Under this heading, click the link titled `Download Windows x86 MSI Installer`.

   See Figure 1-3. This is a link for a 32-bit installation, which makes things work better with third-party libraries. Use the 32-bit installer even if you have a 64-bit machine and even if you have no idea what this paragraph is talking about.

5. If you're asked to choose whether to run or save the file, choose Run.

   This downloads `python2.7.9.msi` and starts running the installer.

6. If you get a security warning when the installer begins (or at random times during the installation), choose Run.

7. Accept the default installation options that the installer provides.

Figure 1-2: Download Python with Internet Explorer.

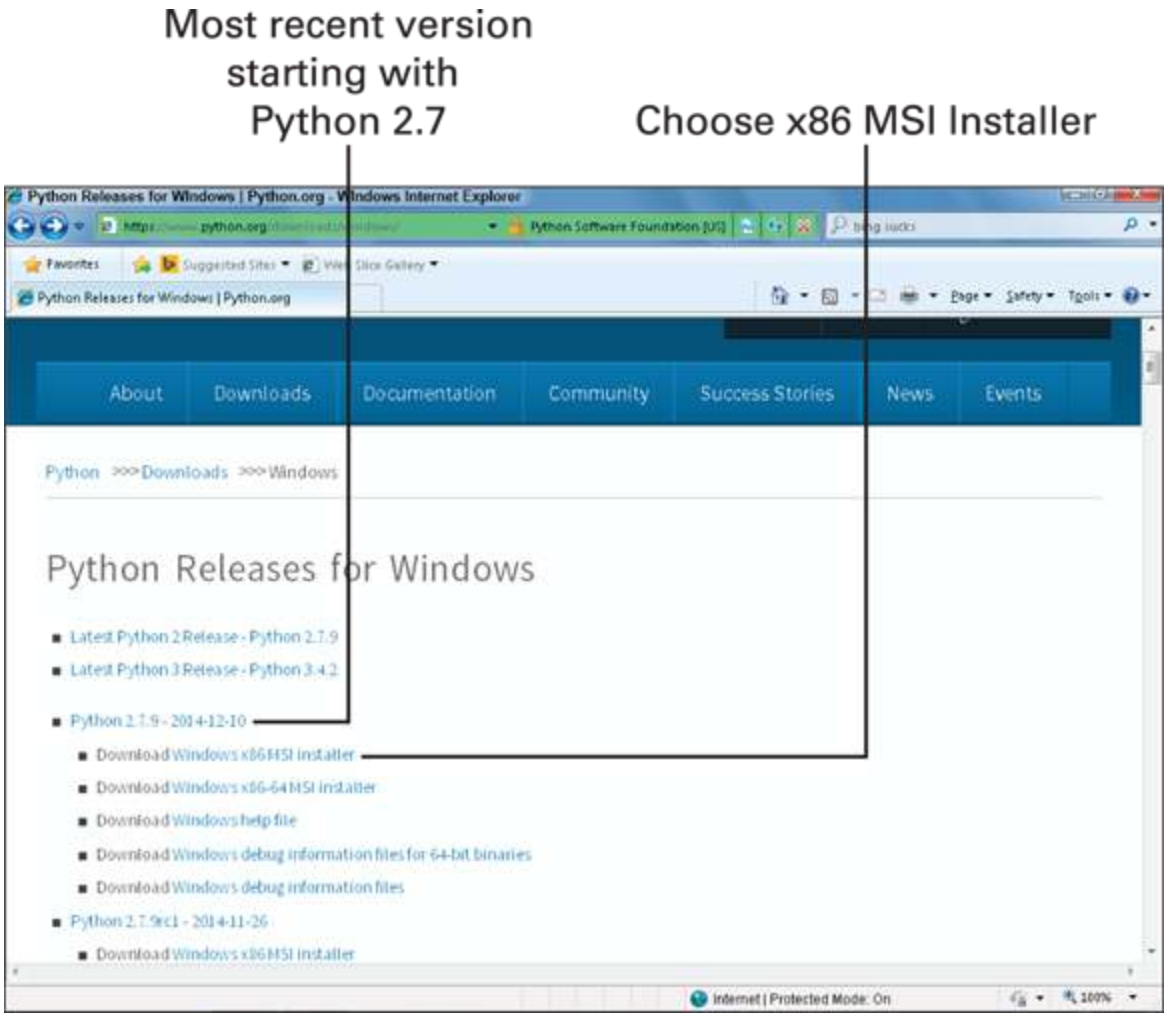


Figure 1-3: Python x86 MSI Installer.

Install Python for Linux

If you're running Linux, confirm that you have version 2.7.9 of Python installed, rather than version 3. This shouldn't be a problem because Python 2.7 is installed by default in recent versions of OpenSuSE, Ubuntu, and Red Hat Fedora.

In the nutty odd case when someone has Python 3 but not Python 2.7, read your distribution's documentation for how to use the package manager and get Python 2.7 and IDLE.

Pin Python to Your Start Menu

After you've downloaded Python, it's a good idea to pin it to your Start menu. That way you can find it more easily for the rest of this book.

Type `Python` in the Start menu's search bar, or click All Programs. In the folder `Python 2.7`, you should find the following entries (see Figure 1-4):

- IDLE (Python GUI)
- Module Docs
- Python (command line)
- Python Manuals
- Uninstall Python

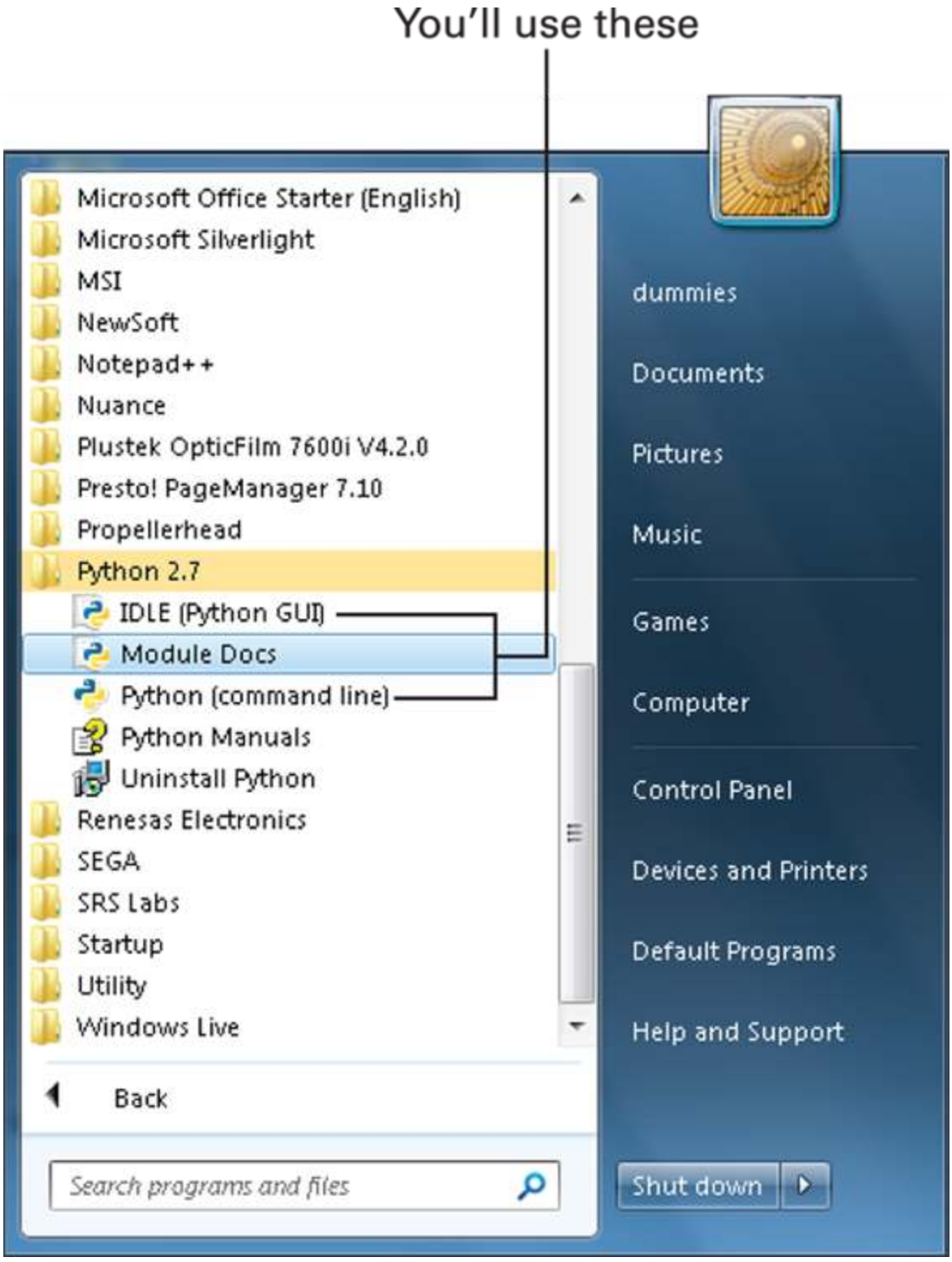


Figure 1-4: Python entries in Start menu.

Of these, you'll use:

- IDLE (Python GUI)
- Python (command line)

To make IDLE and the command line easier to find, pin them to your Start menu:

1. Open your Start menu.
2. Choose All Programs ⇒ Python 2.7.
3. Right-click IDLE (Python GUI). See Figure 1-5.
4. Select Pin to Start Menu.
5. Right-click Python (command line).
6. Select Pin to Start Menu.

   You should see the entries at the top of your Start menu. If you prefer, you can pin them to your taskbar.

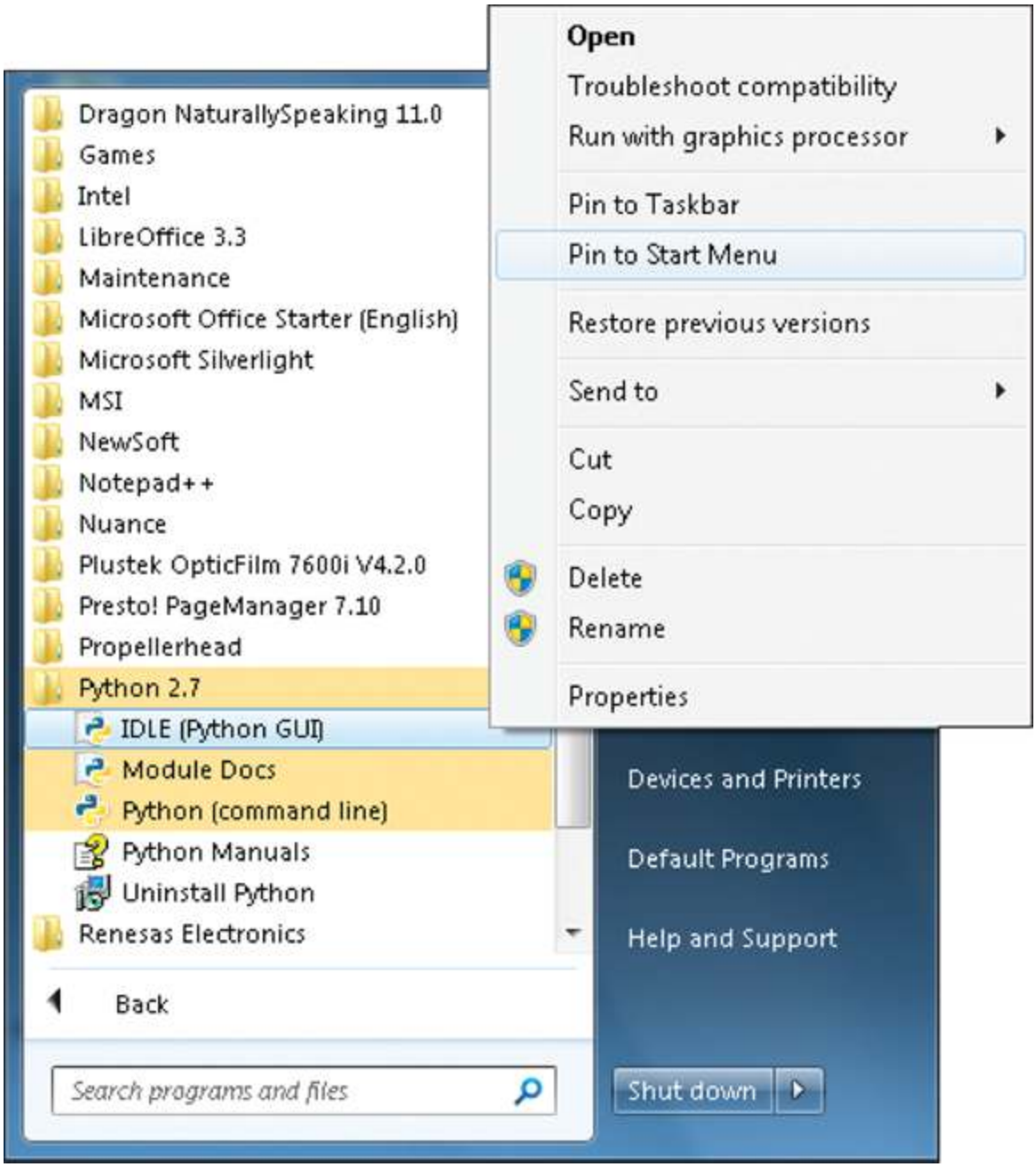


Figure 1-5: Right-click to pin Python to your Start menu.

Python on tablets

Are you interested in running or programming Python using your tablet? If you're interested in writing a program for a tablet, take a look at the Kivy library. I installed SL4A, the Scripting Layer for Android, on my tablet, along with its Python interpreter and drafted some early chapters using SL4A. Check your tablet's app store for Python interpreters and see whether any fit your ability.

Because different tablets display graphics different ways, Python has to use special libraries to write programs other than plain text. Tablets are only useful for the non-graphical projects in this book — unless you're prepared to research those libraries and rewrite the projects to use them. In that case, you're beyond this book.

Also, it's better to have a hardware keyboard. *Soft* (onscreen) keyboards usually don't give easy access to the punctuation that Python needs (or, in my experience, for everyday English — but that's another matter ...).

Start the Python Interpreter

You can't use this book unless you can start Python. Click Python (command line) that you pinned to your Start menu.

If you haven't pinned it yet, do it. No, seriously. You can also search for Python in the search bar and click Python (command line) in the Programs section of the results.

You get a window that looks like Figure 1-6.

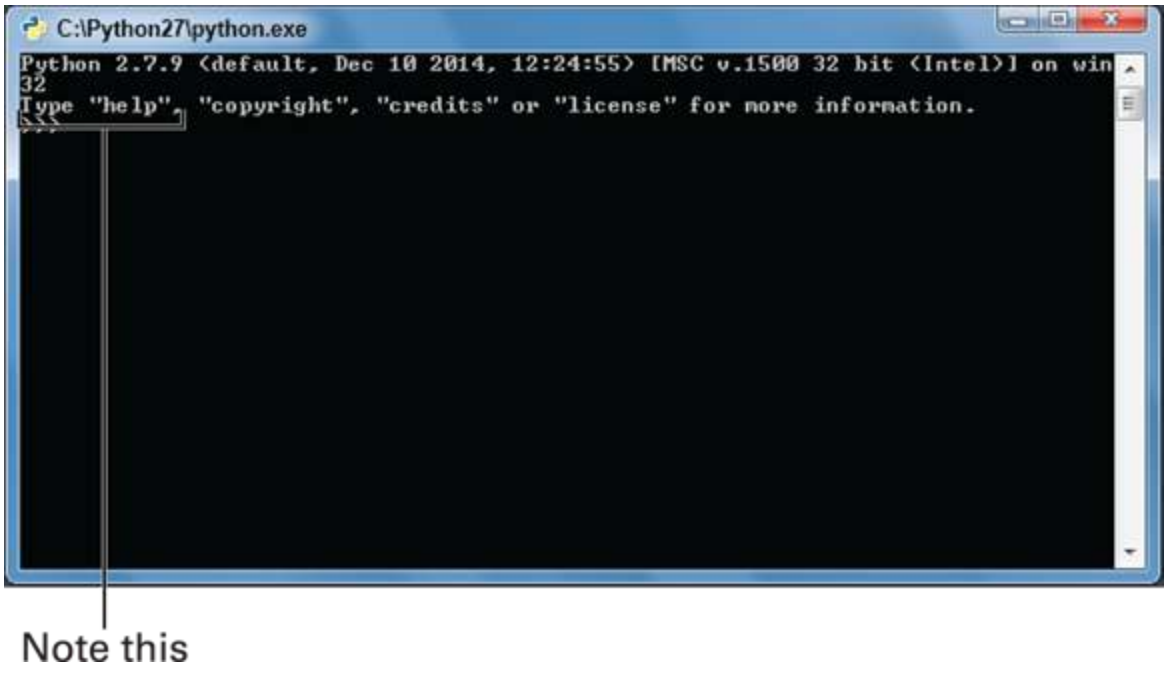


Figure 1-6: The Python command line suggests you type `help` if you want help.

Use Python's Built-In Documentation

Python comes with its own help. In fact, in Figure 1-6 it even tells you about it in the welcome message. If you type `help` and press Enter, you get more help options. Type `help()` (including the parentheses) to get interactive help like what you see in Figure 1-7.

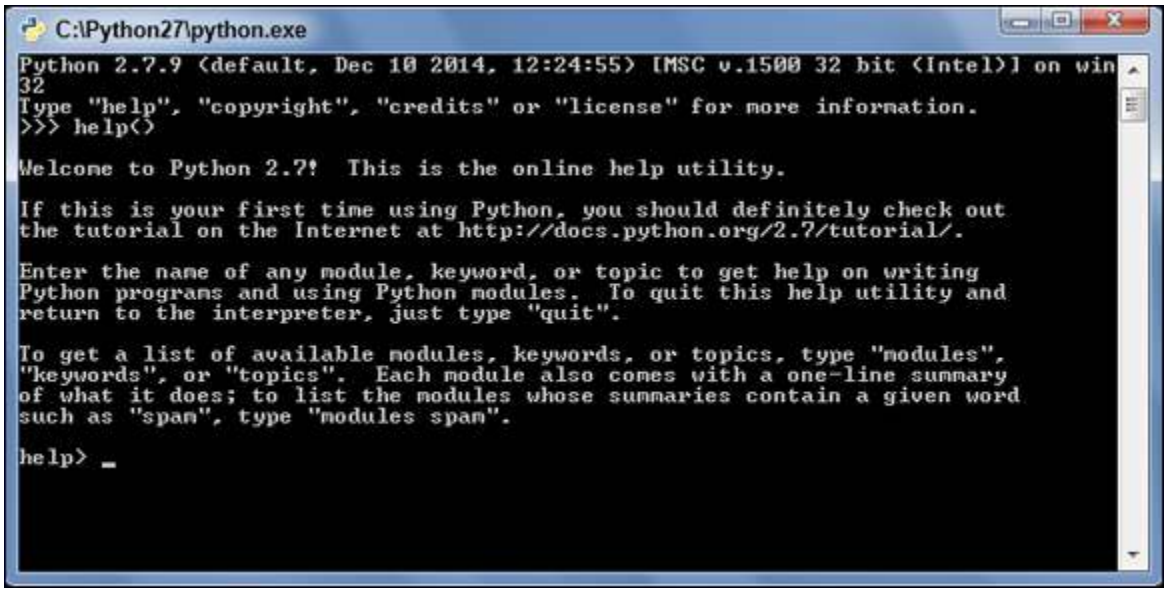


Figure 1-7: Python's interactive help is ready for you to type.

Once you're inside the help service, you can get information on any topic. Just type the topic. The help service can't handle complex queries (like the ones you use to search the Internet), so keep it simple. For example, instead of typing `how do I get a range of numbers`, type `range` (and read what it has to say). To quit interactive help, type `quit`.

You can write help text for your own programs. You do it in Project 5.

Put the Kibosh on the Python Interpreter

You can't use this book unless you can start Python, but you'll want to stop Python after you've started it. If you can't wait to start programming, skip ahead to Project 2.

You can put the kibosh on Python from the command line by doing one of the following: