IN THIS CHAPTER

- Deciding on a programming language
- Mastering Python language basics
- Getting up and running with Python

Chapter 1
Automating with Python

Welcome to Python automation! In this chapter, you explore why Python is the ideal language for automating mundane, time-consuming computer tasks. If you've ever found yourself stuck doing the same boring computer chores over and over — like renaming a bazillion files, sorting through spreadsheets, or downloading stuff from the web — Python may just become your new best friend. It's a programming language that's easy to pick up, even if you're not a tech wizard, and it's perfect for automating those mind-numbing tasks that eat up your time. Think of Python as a trusty robot assistant: You tell it what to do in plain, simple words (well, code), and it does your work in no time at all.

What makes Python so great for automation is the fact that it has a little something for everyone. Python has built-in tools to handle all sorts of everyday tasks — like managing files, crunching data, bossing around your computer — and a huge pile of free add-ons (called *libraries*) can do even fancier things, like scraping websites or sending emails. You don't need to be a coding genius to get started — just a few lines of Python can save you hours of clicking and typing. So, whether you're organizing your music collection or taming a messy inbox, Python's got your back, making life a whole lot easier with a few friendly commands.

THINKING LIKE A SOFTWARE ENGINEER: PUTTING ARTIFICIAL INTELLIGENCE TO WORK FOR YOU

Writing Python automation requires writing computer code. This book isn't a replacement for a beginner's tutorial on the entire Python language. Instead, it's a collection of Python automation scripts, designed to automate and simplify mundane, time-consuming computer tasks.

These days, most software engineers (people who write computer code for a living) use artificial intelligence (AI) to help with writing code. Sure, you'll hear many software engineers complain that AI can't write code as well as they can, but that sentiment may be rooted in feeling threatened.

In addition to writing code, AI can easily answer any questions that come up along the way as you're using this book. If I throw some terminology at you that leaves you scratching your head, ask AI to explain things. If some code leaves you stymied, show the code to AI and ask it to explain the code to you.

You can even tell AI to write an entire script for you, doing exactly what a script in this book does. But don't be surprised if the code you get from AI looks different from what's in this book. Python offers many tools and techniques for accomplishing any task. There's no telling exactly how AI will generate code to perform some feat. If AI gives you a script that looks nothing like the script in this book, that doesn't mean one is right and the other is wrong — you probably just have two scripts that do the same thing in different ways. That's not unusual. They say there's more than one way to bake a cake. Likewise, there's more than one way to write a script to accomplish some task.

Choosing a Programming Language

There are many programming languages in the world. They have names like C#, Go, Java, JavaScript, Python, and TypeScript, to name a few. The TIOBE Index (`www.tiobe.com/tiobe-index`) consistently ranks Python as the most popular language of our time.

JavaScript is great for creating web apps, but it's rarely used for anything else. Python excels at AI and automation. In fact, Python has so many ready-to-use modules designed for automation that it would probably be crazy to use any language *other* than Python for the kinds of automation scripts you'll see throughout this book.

There's a lot to like about Python — and many reasons to learn Python beyond automation. For one, many people regard Python as the easiest language for many beginners to learn. Python's syntax is clean and simple — it reads almost like English.


TIP

You're never stuck without information with Python. There are endless tutorials, forums, and free *libraries* (premade code you can borrow) to help you out. Virtually every modern AI chatbot is perfectly capable of writing Python code for you and answering any questions about Python that pop into your head.

Python lets you write short, powerful code. What may take 20 lines in another language often takes just a few lines in Python. That means less typing and fewer mistakes to try to ferret out. Plus, modern AI can debug your existing code as easily as it can write code for you.

Let's zoom in on automation — the topic of this book. When it comes to automation, Python is a superstar. Whether you're on Linux, macOS, or Windows, Python works like a charm. Write your automation script once, and it'll run anywhere. No need to reinvent the wheel for different systems.

With Python, you can write a quick script to handle many tasks in minutes. Although the following code below may not mean much to you right now, it illustrates how you can take a daunting task, like renaming hundreds of files in a folder, with just a few lines of code:

```
import os
for filename in os.listdir("."):
    os.rename(filename, filename.replace("old", "new"))
```

Tiny bits of code like that can handle big automation tasks.

Beyond file tasks, Python plays nice with application programming interface (APIs; define here), databases, Microsoft Excel files, and AI. If you're automating something like "Check my email, grab attachments, and update a spreadsheet," Python can tie it all together smoothly.


TECHNICAL STUFF

APIs allow Python to interact with AI and other powerful online capabilities, without your having to reinvent the wheel or host huge files on your own computer. APIs are a hallmark of modern computing, and you definitely want to use a programming language that makes API access easy.

REMEMBER

Learning Python is like giving yourself a superpower. Python is easy to start, endlessly useful, and when it comes to automation, unbeatable. You'll save time, impress your friends (or boss), and maybe even have some fun along the way. Perhaps best of all, Python is completely free.

Have I convinced you to choose Python yet?

Understanding Python Syntax

Every language has certain rules of *syntax* that outline how you must arrange words in order for them to make sense. Like, "Teddy, jump three times!" If you say it all jumbled up, or leave out words, like "Jump Teddy three," Teddy may get confused and not know what to do. In programming, syntax is the same thing — you need to order the words so the computer understands what you want. Syntax is just the rules for putting words and symbols in the right order.

Some programming languages require lots of punctuation, in addition to words, as part of their syntax. That gets tiresome and makes learning more difficult. I'll give you a simple example — a piece of code that checks whether a number is even or odd and prints a message — in both JavaScript and Python.

JavaScript seems very "busy" with parentheses, curly brackets, and semicolons:

```
function checkEvenOrOdd(number) {
    if (number % 2 === 0) {
        console.log("The number " + number + " is even!");
    } else {
        console.log("The number " + number + " is odd!");
    }
}

checkEvenOrOdd(7);
```

That code looks like something written by aliens. But that's what a JavaScript requires. You've got:

- Curly brackets { } to wrap the function and the `if...else` blocks.
- Parentheses ( ) for the function definition and the `if` condition.
- A semicolon (;) at the end of each line (JavaScript loves semicolons).

Now here's the same thing in Python:

```
def check_even_or_odd(number):
    if number % 2 == 0:
        print(f"The number {number} is even!")
    else:
        print(f"The number {number} is odd!")

check_even_or_odd(7)
```

Granted, it's still not plain English. But it's much, much cleaner and simpler. Here's what's special about Python:

- No curly brackets! Python uses *indentation* (those spaces at the start of lines) to know what's inside the function or `if...else`. It's like the code is breathing — it looks airy and neat.
- Fewer parentheses — only needed for the function definition, not the `if` condition.
- No semicolons — Python doesn't need them, so the code is less cluttered.


TIP

As an experienced instructor who has taught thousands of software developers, I can assure that all the curly brackets and semicolons are the toughest things for beginners to get used to — they're among the main things that drive people away from learning to code. Learning Python first lets you dodge that bullet.

Getting Python

Python is super lightweight and doesn't demand much from your hardware, which is one reason it's so popular. Think of this as the "minimum stuff" your computer needs to run Python and get started with coding or automation.

Identifying the hardware requirements

You can run Python on almost any modern computer. That doesn't include mobile devices like phones and tablets, but it does include most desktops and laptops. Here's what you'll need at the bare minimum:

- Python works on Linux, macOS, Windows (7, 8, 10, or 11), and even some mobile systems. Basically, if it's a computer from the last 10 to 15 years, you're good!

- Any modern processor, including Apple M series, or even just a basic processor like an Intel or AMD processor works fine. Even a 1 gigahertz (GHz) single-core central processing unit (CPU) can handle it, but it may feel slow for big projects.
- In terms of random access memory (RAM), 512 megabytes (MB) is enough to run Python itself, but 2GB or more is better if you're doing anything practical (like automation or running other programs at the same time). Most modern computers have 8GB of RAM or more, so you're probably covered.
- Python's installer is tiny — about 30MB to 50MB to download and install. You'll want at least 100MB to 200MB of free space for Python, its libraries, and your own code files. If you're adding big libraries (like for data science), a few gigabytes of free space is smart.
- No special graphics card or graphics processing unit (GPU) is needed. Python runs in a text window, so any basic screen works.

To get your system specs in Windows, press Windows+I to open Settings and choose System➪About. On a Mac, click the Apple menu in the upper-left corner of your screen, and choose About This Mac.

If you're automating something heavy — like controlling a web browser with selenium or processing tons of files — you'll want more RAM (maybe 8GB) and a faster CPU. But for most people learning Python, and for everyday automation (like renaming files or sending emails), even a cheap laptop is probably sufficient, as long it's not a Chromebook or a similar device with a mobile operating system.

Installing Python

To use Python, you may first have to install it on your computer. Some Mac computers come with Python version 2 preinstalled. But these days, you really need to use Python 3, so plan on installing Python yourself. It's free, it's easy, and I can give you the steps. However, I *can't* tell you exactly what you'll see when you browse to the Python website, because websites change often. If you're using a Mac or Windows PC, follow these steps (if you're using Linux, see the nearby sidebar):

1. **Go to `www.python.org`.**
2. **Click Downloads and click either Mac or Windows.**

   You don't technically need to click Mac or Windows — the website will detect which operating system you're using and when you hover your mouse over Downloads (as I did in Figure 1-1), you'll see the option to download the correct

version. As you can see in the figure, I was using Windows, so the website offered that automatically.

3. **Click the button that shows the current version number.**

   In Figure 1-1, the version is 3.13.2, but the number you see may be different.

4. **Open the folder to which you downloaded Python.**

   This is usually your `Downloads` folder.

5. **Double-click the icon of the downloaded file.**

6. **Follow the onscreen instructions to Install (or Upgrade Now if you're given that option).**

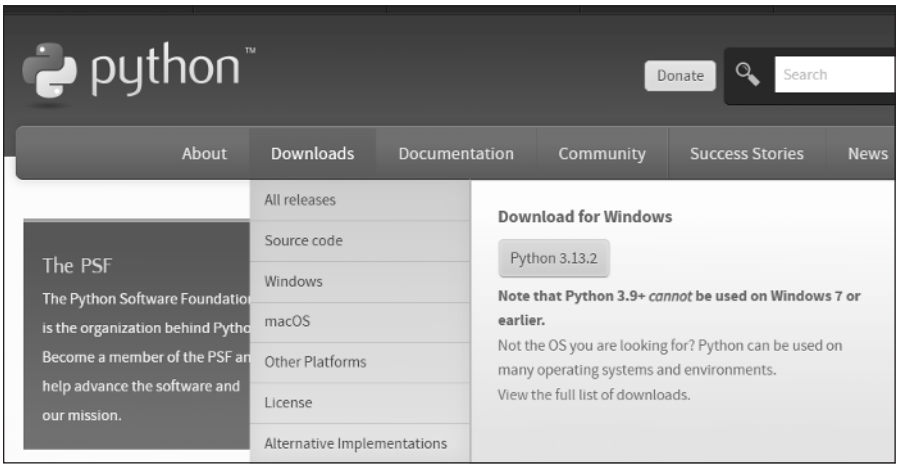

FIGURE 1-1: Download options from the Python website.

INSTALLING PYTHON ON LINUX

Most Linux distributions come with Python preinstalled (usually Python 3). There are many different Linux distributions (or distros), including Arch, Debian, Fedora, and Ubuntu. I can't give step-by-step instructions for each. But a good starting point may be to determine whether you already have Linux installed and, if so, which version. You should be able to do so following these steps:

1. **Press Ctrl+Alt+T to open the Terminal.**
2. **Type the following and press Enter:**

```
python3 --version
```

(continued)

(continued)

If the command returns something like Python 3.12.2, Python is installed. If you get an error, try the following command:

```
python --version
```

If you get an error message on both tries, or you want a newer version of Python, you can still install Python from the Python website. Download and install the `Gzipped source tarball` or `XZ compressed source tarball` file. Or check the documentation for your specific Linux distribution for recommendations. Optionally, you can also ask any AI for recommendations related to your specific Linux distro.