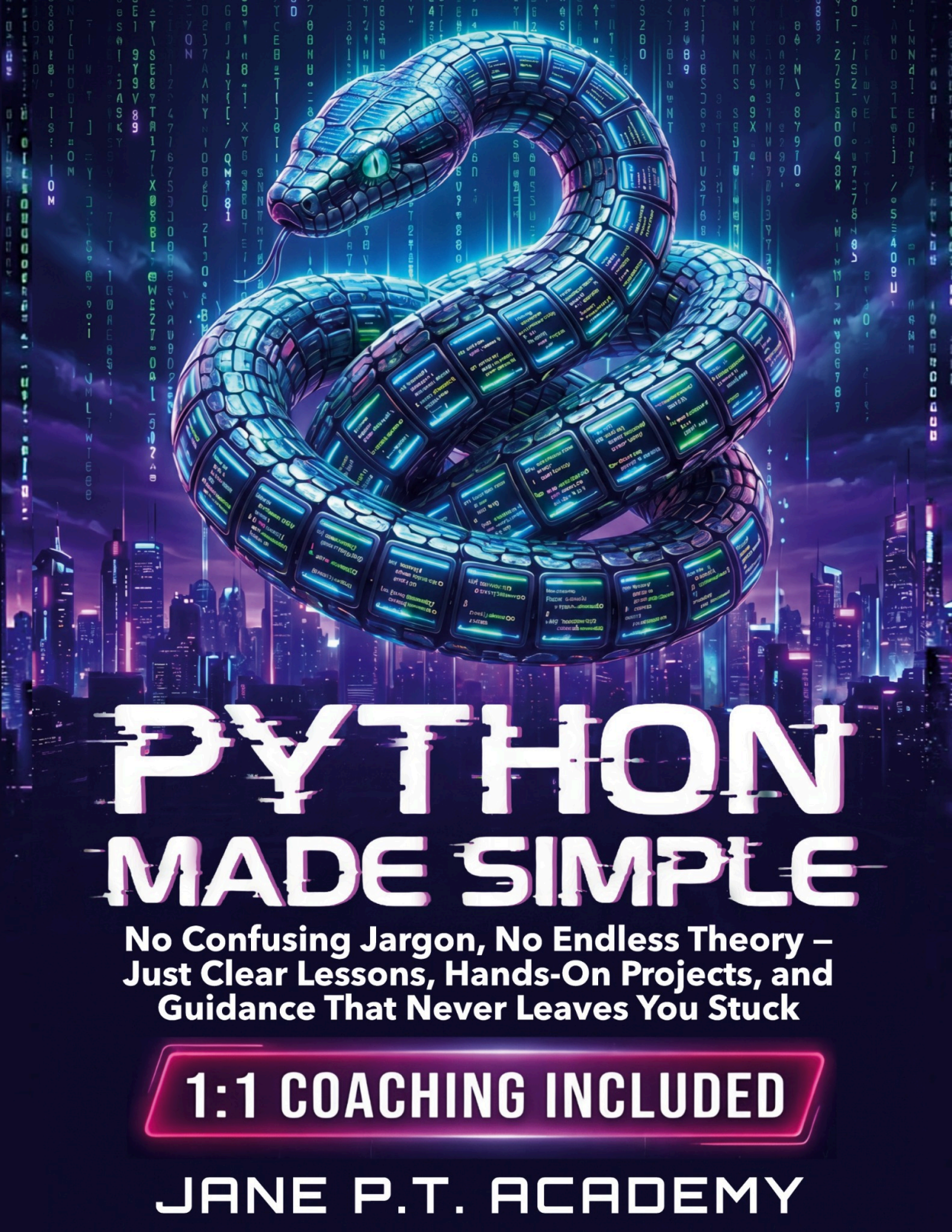

PYTHON
MADE SIMPLE
No Confusing Jargon, No Endless Theory –
Just Clear Lessons, Hands-On Projects, and
Guidance That Never Leaves You Stuck
1:1 COACHING INCLUDED
JANE P.T. ACADEMY

PYTHON MADE SIMPLE

**No Confusing Jargon, No Endless Theory –
Just Clear Lessons, Hands-On Projects, and
Guidance That Never Leaves You Stuck**

JANE P.T.ACADEMY

Herstellung und Verlag:

BoD · Books on Demand GmbH, Überseering 33, 22297 Hamburg
bod@bod.de

ISBN: 9783696765101

TABLE OF CONTENTS

BEFORE YOU START: A Letter to the Reader

"This Book Is Different: Here's Why"

Let me guess.

You've tried to learn programming before. Maybe you watched a few YouTube tutorials, maybe you bought another book, maybe you signed up for a free course online and made it through the first two lessons before life got in the way, or before the material got confusing enough that you quietly closed the tab and told yourself *"maybe later."*

Maybe "later" has been going on for a while now.

If that's you, I want you to know something before you read a single line of code in this book: **there is nothing wrong with you.** You didn't fail because you're not smart enough. You didn't fail because programming is too hard. You failed because the material you used was designed for the wrong person, someone who already thinks like a programmer, someone who enjoys reading 400-page manuals, someone who finds dense technical jargon *interesting* rather than exhausting.

That person is not most people. And this book was not written for that person.

This book was written for you, the person who has a real reason to learn Python. Maybe you want a better job. Maybe you want more money, more freedom, the ability to work from anywhere. Maybe you're tired of feeling left behind while the world keeps talking about AI, automation, and tech careers like they're a club you're not allowed into. Maybe someone told you that Python is the easiest way in, and you thought: *okay, one more try.*

This is the try that works. Here's why.

First: this book respects your time. There is no unnecessary theory in these pages. Every concept you'll learn has been chosen because you will actually use it. Everything that exists only to impress other programmers has been cut. What remains is the essential core, the 20% of Python that gets you 80% of the results.

Second: this book is built around doing, not reading. You will write code starting in Chapter 1. Not Chapter 5. Not after a long theoretical introduction. Chapter 1. Because the only way to learn programming is to program, and the sooner you experience the satisfaction of making something work, the sooner your brain accepts that this is something you can actually do.

Third: this book has a safety net. Getting stuck is not a sign of failure, it's a normal part of learning to code. Every developer in the world, at every level, gets stuck. The difference between beginners who quit and beginners who make it is not intelligence. It's having somewhere to turn when things get confusing. That's why this book comes with a built-in support system: a troubleshooting guide, a cheat sheet, project resources, and, for readers who need it most, access to a real 1:1 coaching session. You'll find the details in the Bonus Section at the end.

One last thing. Learning Python is not the goal. The goal is what comes *after* Python, the job, the income, the freedom, the confidence of knowing you can build something from nothing using only a computer and your own mind. Python is the door. This book is the key.

Let's open it.

PART I: QUICK START

Feel capable in the first 30 minutes.

Chapter 1
What Is Python
And Why It Can Change Your Career

Here's a number worth knowing: **Python is the most in-demand programming language in the world**, and has been for several years running. It tops the charts on every major developer survey, it's required in job postings at Google, NASA, Netflix, Spotify, and thousands of companies you've never heard of. It powers artificial intelligence, data analysis, web applications, automation tools, and scientific research.

But here's the number that probably matters more to you right now: the average salary for a Python developer in the United States is **over $110,000 per year.** Entry-level positions start around $70,000. Freelancers charge anywhere from $50 to $150 per hour for Python work. Remote positions are abundant. The demand is not slowing down, if anything, the rise of AI and data-driven business has made Python skills *more* valuable than they were five years ago, not less.

So the question isn't whether Python is worth learning. The question is: what exactly *is* it, and how does someone like you, with no technical background, actually get started?

Let's answer that right now.

What Python Actually Is

Python is a programming language. That means it's a way of giving instructions to a computer, but unlike the binary code that computers actually run on, Python is written in something close to plain English. Compare these two ways of telling a computer to display a "Hello, World!" message:
In binary machine code, that instruction looks like an incomprehensible string of ones and zeros. In Python, it looks like this:

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

That's it. One line. Plain, readable, almost like a sentence. This is the philosophy that makes Python different from most other languages: it was designed from the beginning to be *human-readable*. The creator of Python, Guido van Rossum, had a simple goal when he

built it in the late 1980s, he wanted a language that was powerful enough for serious work but clean enough that a human being could read the code and understand it without a manual.

He succeeded. And that single design decision is the reason Python became the default choice for beginners, scientists, data analysts, and AI researchers around the world.

What You Can Actually Do With Python

One of the biggest mistakes beginners make is thinking of Python as an abstract skill, something you "learn" in a vacuum and then figure out how to use later. Let's flip that. Before you write a single line of code, here is a concrete map of what Python can do for you in the real world:

Automation is one of the most immediately practical applications. Python can automate any repetitive task you do on a computer, renaming hundreds of files at once, sending scheduled emails, filling out web forms, organizing folders, scraping data from websites. If you do something on a computer more than twice a week, Python can probably do it for you. Businesses pay real money for this.

Data Analysis is the skill that opens the door to data analyst and business analyst roles, two of the fastest-growing job categories in the US market. With Python, you can take a spreadsheet with ten thousand rows, clean it, analyze it, and produce a visual report in under an hour. The companies hiring for these roles are not just tech companies, they're banks, hospitals, marketing agencies, logistics firms, and retailers.

Web Development with Python frameworks like Flask and Django allows you to build websites and web applications. Countless startups and agencies are built on Python backends. If you've ever thought about building your own product or working as a web developer, Python is one of the clearest paths there.

Artificial Intelligence and Machine Learning run largely on Python. The libraries that power AI tools, including many of the AI systems you interact with today, are built in Python. You don't need to become an AI researcher to benefit from this: even basic familiarity with Python's data and AI tools makes your resume significantly more attractive to modern employers.

Freelancing and Consulting is a path available to you faster than you might think. As you'll discover in Part 6 of this book, there is a consistent, real market for Python freelancers who can build small automation tools, scrape data, or clean spreadsheets. These are skills you will have before you finish this book.

The Truth About How Long It Takes

Let's have an honest conversation here, because a lot of books and courses are not honest with you about this.

You will not master Python in a day. Anyone who tells you otherwise is selling you something. The title of this book says "crash course", and it means it. You *will* be writing

working Python programs quickly. You *will* feel competent and capable by the end of Part 1. You *will* have built four real projects by the time you reach Part 4.
But mastery is a longer road, and that's fine. Here is a realistic timeline for someone who practices consistently for 30 to 60 minutes per day:
After **one week**, you will understand the fundamentals, variables, loops, functions, and be able to write simple programs from scratch. After **one month**, you will have completed multiple projects and have a basic portfolio. After **three months**, you will be ready to apply for entry-level jobs, take on small freelance projects, or confidently pursue a specialization like data analysis or web development. After **six months** of consistent practice, you will be a functional Python developer.

This book covers the first month of that journey, and it covers it better than anything else you'll find, because it's designed specifically around how beginners actually learn, not how experienced developers think beginners should learn.

Why You Don't Need Math, a CS Degree, or Any Prior Experience

This is the belief that stops more people from starting than anything else: *"I'm not a math person. I'm not technical. Programming is not for people like me."*
Here is the reality. The vast majority of practical Python programming, automation, data work, web development, scripting, requires nothing beyond basic arithmetic. Addition, subtraction, the occasional percentage. That's it. You will not be solving differential equations. You will not be writing complex algorithms from scratch. You will be telling a computer to do things, in a language that was explicitly designed to look like English.
The programmers who tell you that you need a strong math background are usually talking about a very specific subset of programming, competitive algorithms, certain areas of AI research, graphics rendering. That is not what this book is about. This book is about giving you practical, marketable Python skills as efficiently as possible. For that purpose, if you can do basic arithmetic and you can read English, you are fully qualified to begin.

Right now. Today. On the next page.

Your Transformation Starts Here

Before you move to Chapter 2, take 30 seconds and do something that might feel a little unusual for a programming book: write down, somewhere you'll see it, *why* you're learning Python. Be specific. Not "to get better at tech", but the real reason. The job you want. The income you want. The freedom you want. The version of yourself you're trying to become. Keep that somewhere visible. On days when a piece of code doesn't work and frustration creeps in, that note is your anchor. The code will eventually work. The reason you're here is bigger than any single error message.

Now let's get Python on your computer.

Chapter 2
Setup in 15 Minutes
Zero Confusion Guide

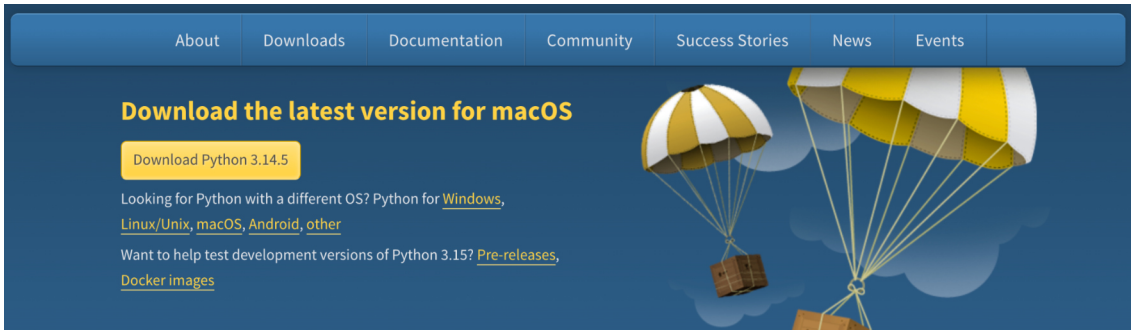
This is the chapter most beginners dread, and the chapter most books get wrong. The typical setup guide assumes you already know what a "PATH variable" is, or why your terminal is showing a cryptic error, or which of the twelve download options on the Python website is the right one.

We're not doing that here.
This chapter has one job: get Python running on your computer, correctly, in 15 minutes, with zero frustration. Every step is explicit. Every screenshot reference is described clearly. Every common error has a fix. By the end of this chapter, you will have a working Python environment and you'll know exactly how to use it.

Let's go.

Step 1: Download Python

Open your web browser and go to **python.org/downloads**. The website will automatically detect your operating system and present the correct download button at the top of the page. As of the time this book was written, the current stable version is Python 3. Click the large yellow button that says *"Download Python 3.x.x"*, the exact version number doesn't matter as long as it starts with 3 as you can see in the image below.



⚠ **Important:** Do *not* download Python 2. It is an older version that is no longer supported and is not compatible with modern code. Always use Python 3.

The download will take between 30 seconds and 2 minutes depending on your connection speed.

Step 2: Install Python

On Windows:

Run the installer file you just downloaded. Before you click anything else, look at the bottom of the installer window. You will see a checkbox that says **"Add Python to PATH."** This checkbox is *not checked by default*, and if you skip it, Python will not work from your terminal. **Check this box first.** Then click "Install Now." The installation takes about 2 minutes. When it finishes, click "Close."

On Mac:

Run the `.pkg` installer file and follow the prompts, it's a standard Mac installation. Click "Continue," agree to the license, and click "Install." You may be asked to enter your Mac password. After installation completes, you may see a window open asking you to install additional certificates. Run that step, it's needed for Python to connect to the internet securely in later projects.

On Linux:

If you're on Ubuntu or any Debian-based Linux distribution, Python 3 is likely already installed. Open your terminal and type:

```
python3 --version
```

If a version number appears (like `Python 3.11.2`), you're done. If not, type the following:

```
sudo apt-get install python3
```

and press Enter.

Step 3: Verify the Installation

This step takes 30 seconds and it's important, it confirms that everything is working before you go any further.

On Windows: Press the Windows key, type `cmd`, and press Enter to open the Command Prompt. Type the following and press Enter:

```
python --version
```

You should see something like `Python 3.11.4`. If you see that, you're done with this step.

On Mac or Linux: Open the Terminal application and type:

```
python3 --version
```

You should see the version number appear. Done.

🔧 **If you see an error instead of a version number:** On Windows, the most common cause is forgetting to check the box "Add Python to PATH" during installation. Uninstall Python from your Control Panel, re-run the installer, and make sure that checkbox is checked. On Mac, try typing in your terminal:

```
python3
```

instead of:

```
python
```

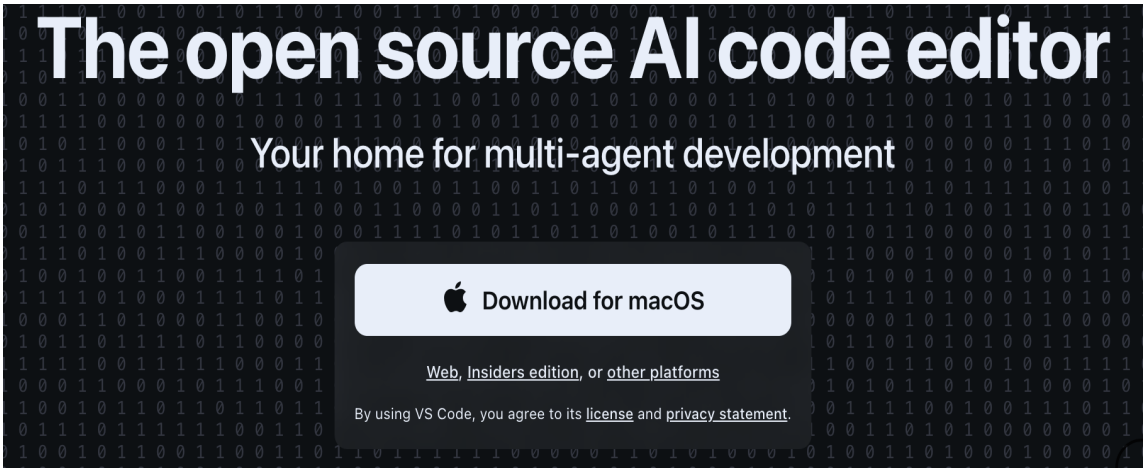
On Linux, try:

```
sudo apt-get install python3
```

as shown above.

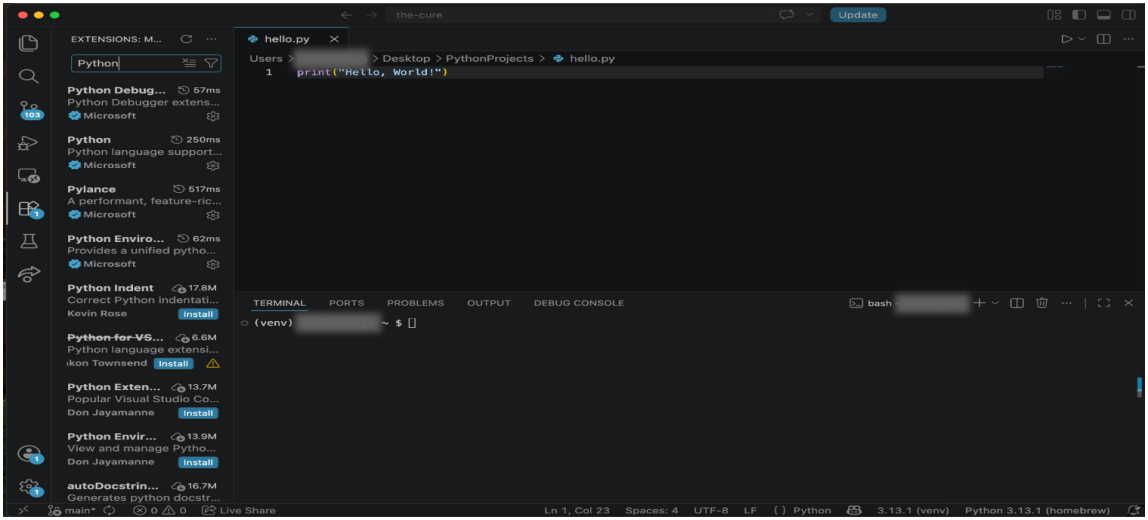
Step 4: Install VS Code, Your Code Editor

Python is now installed, but you need a place to write your code. You *could* write Python in a plain text file, but a proper code editor makes the experience dramatically better, it highlights your syntax with colors, catches common errors as you type, and gives you tools that will save you hours of debugging time.
The editor we recommend, and the one this book assumes you're using, is **Visual Studio Code**, commonly called VS Code. It's free, it's lightweight, it's the most popular code editor in the world among professional developers, and it works perfectly for beginners.
Go to **code.visualstudio.com** and click the large download button for your operating system like the one visible in the image below. Run the installer and follow the default prompts, the default settings are fine for everything.

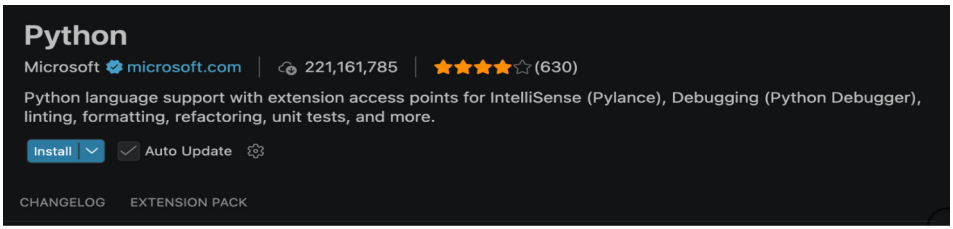


Once VS Code is open, you need to install one extension: the **Python extension by Microsoft**. Here's how:

On the left side of VS Code, you'll see a column of icons. Click the one that looks like four small squares, that's the Extensions panel. In the search bar at the top, type `Python`.



The first results should be “Python” by Microsoft, with several million downloads. Select the one with subtitle “Python language support” and click “Install” like shown in the picture below.



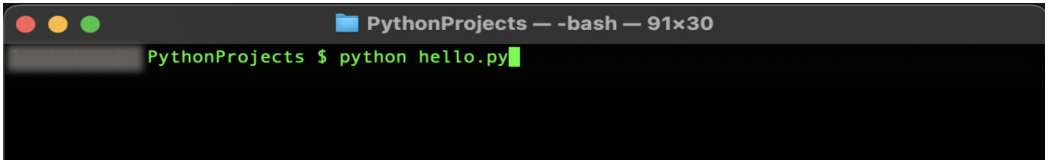
This adds Python language support, autocomplete, and error highlighting to your editor.

Step 5: Create Your First Python File

Now let’s make sure the whole system works together.
In VS Code, go to . Then go to , name your file `hello.py`, and save it somewhere easy to find, a new folder on your Desktop called `PythonProjects` works perfectly. The `.py` extension of the file tells VS Code that this is a Python file.
You now have a working Python development environment. In the next chapter, you’ll write your first real program in this file. But first, let’s look at one more tool you’ll be using constantly.

Your Terminal: Demystified in 2 Minutes

The terminal, also called the command line, command prompt, or console, is a text-based window where you type instructions directly to your computer. It looks intimidating at first, but you only need to know three commands to use it effectively for this book.



The first command to type ito the terminal is:

```
python hello.py
```

or:

```
python3 hello.py
```

on Mac/Linux), this runs the Python program you are going to write in the file you have just created. S of now, the file is empty, so running it in the terminal will return an empty output.
The second command is:

```
cd PythonProjects
```

this navigates into the project folder you have created when saving the file you created using VS Code.
The third command is on Mac/Linux:

```
ls
```

or on Windows:

```
dir
```

This shows you the files in your current location.
That's it. Those three commands will carry you through everything in this book. Everything else you'll learn as you need it.
VS Code also has a built-in terminal that you can open with the keyboard shortcut **Ctrl + `** (the backtick key, just below Escape). This is usually the most convenient way to run your programs, you write code in the top half of the screen and run it in the terminal at the bottom.

Your Setup Is Complete

Take a moment to appreciate what you just did. You have Python installed and verified. You have a professional code editor configured correctly. You know how to run a Python file. This is the exact same setup that professional Python developers use every day.
The hard part of getting started, the part that stops a surprising number of beginners before they write a single line, is done. In the next chapter, you'll write your first real program and experience for the first time what it feels like to make a computer do exactly what you tell it to.

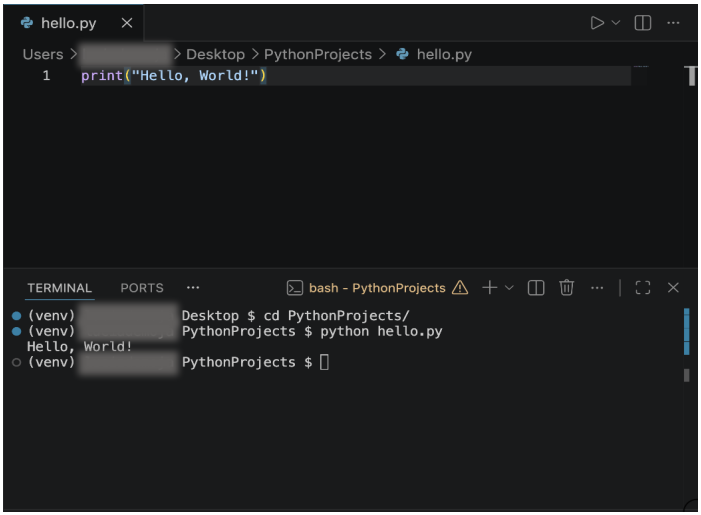
It's a better feeling than you might expect.

3 Chapter 3
Your First Python Program
Immediate Win

This is the moment everything becomes real.
Up until now, you've been reading about Python. You've learned what it is, why it matters, and how to install it. All of that is preparation. This chapter is where you actually become someone who writes code, and it happens faster than you might think.
By the end of this chapter, you will have written, run, and modified a real Python program. It won't be complicated. But it will work. And the moment you see your program produce output on the screen for the first time, something shifts. The abstract idea of "learning to code" becomes something concrete and real: *I just did that.*

Let's do it.

The Most Famous Line in Programming



Open VS Code. Open the `hello.py` file you created at the end of Chapter 2. If you haven't created it yet, go to , then , name it `hello.py`, and save it in your `PythonProjects` folder.
In the empty file, type exactly this:

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

Now open the terminal in VS Code with **Ctrl +** `, navigate to the folder where you saved your file (using `cd PythonProjects`), and type:

```
python hello.py
```

Press Enter. You should see this appear in the terminal:

```
Hello, world!
```

You just ran your first Python program.

You should have the code on top of the VS Code editor window, and the terminal below, as shown in the image on the right.

Explaining the text shown in the picture:

```
(venv) username Desktop $ cd PythonProjects/
```

`(venv)`: stands for Virtual Environment. We will not need to look into this as you will not use it for the scope of this book. It should not be visible in your terminal.
`username`: it is the username for the user account on the laptop, your terminal should show your username.
`Desktop`: the folder you are currently working in.
`$`: the dollar sign separates the path (username + folder name) from the commands.
`cd`: stands for Change Directory, it is the command used to move from the folder Desktop to the folder PythonProjects.

Your terminal should therefore show the same pattern:

```
your_username your_current_directory $ your_commands
```

What Just Happened, In Plain English

Let's break down that one line, because understanding what you just wrote is more important than it might seem.

`print` is a **function**, a built-in Python command that does something specific. In this case, it displays whatever you put inside the parentheses on the screen. You'll learn much more about functions in Chapter 9, but for now, think of `print` as Python's way of speaking out loud.
The parentheses `()` tell Python that you're calling, activating, the function. Everything inside the parentheses is what you're giving the function to work with.
`"Hello, world!"` is a **string**, which is programming language for "a piece of text." The quotation marks tell Python that what's inside is text to be treated literally, not as a command. You'll learn everything about strings in Chapter 5.
So in plain English, `print("Hello, world!")` means: *"Hey Python, display the text 'Hello, world!' on the screen."* And Python did exactly that. One line. One instruction. One result.

This is the fundamental pattern of all programming: you give the computer a precise instruction, and it executes it exactly as written. As programs get more complex, you'll be giving many instructions in sequence, with logic and conditions and loops, but it's always the same idea at the core.

Making It Your Own

Now let's do something more interesting. Modify your `hello.py` file so it looks like this:

```
print("Hello, world!")
print("My name is Python.")
print("And I'm going to change your career.")
```