Don't just learn Python—
build an intelligent AI companion.

Learn modern Python through hands-on AI projects and real-world applications while building your own intelligent AI companion system from scratch.

You will learn how to:

- write clean and modern Python code
- build conversational AI applications
- connect Python to modern AI models
- create memory and context-aware systems
- add voice and vision capabilities
- automate tasks with Python
- organize projects like a professional developer
- package and share your applications

About the Author

Dr. Shouke Wei is a researcher, scientist, and entrepreneur specializing in data analysis and modeling, wavelet-based signal processing, and AI-driven applications.

He earned his Ph.D. from Brandenburg University of Technology Cottbus–Senftenberg (Germany) and conducted postdoctoral research at Eawag (Switzerland). He has held research positions at the University of British Columbia (Canada) and served as a distinguished and adjunct professor at multiple universities in China.

His work focuses on bridging theoretical modeling with practical, real-world data science systems.

DEEPSIM PRESS

ISBN 978-1-0675592-9-8

Learn Python by Building Your Own AI Companion SHOUKE WEI

DEEPSIM PRESS

Learn Python by Building Your Own AI Companion

DEEPSIM PRESS

Master **Modern Python** by Building a **Smart AI** Companion System

AI Conversations · Memory & Context · Voice & Vision · Automation & APIs · Real-World Projects

SHOUKE WEI

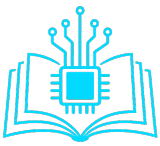
Learn Python by Building Your Own AI Companion

Master Modern Python by Building a Smart AI Companion System

Learn Python by Building Your Own AI Companion

Master Modern Python by Building a Smart AI Companion System

Shouke Wei



DEEPSIM
PRESS

Learn Python by Building Your Own AI Companion

First Edition, 2026

For permission requests, contact the publisher:
Email: shouke.wei@deepsim.ca

ISBN 978-1-0677475-1-0 (Hardcover)
ISBN 978-1-0677475-0-3 (Paperback)
ISBN 978-1-0675592-9-8 (eBook)
DOI 10.5281/zenodo.20102902

Published by
Deepsim Press
An independent imprint of Deepsim Intelligence Technology Inc.
Abbotsford, British Columbia, Canada
https://press.deepsim.ca

This book is intended for educational and professional readers.

For resources, updates, and companion code, visit:
https://press.deepsim.ca/ai-companion

About the Author

Shouke Wei, Ph.D., is a researcher, scientist, and entrepreneur specializing in intelligent IoT systems, robotics, big data analytics, modeling and forecasting, early-warning systems, and edge computing. With academic and industry experience across Europe, North America, and Asia, Dr. Wei is recognized for bridging advanced theory with real-world, production-ready systems.

Dr. Wei earned his Ph.D. in Environmental and Resource Management from the Department of Ecosystems and Environmental Informatics at Brandenburg University of Technology Cottbus–Senftenberg (Germany). He conducted postdoctoral research at the Swiss Federal Institute of Aquatic Science and Technology (Eawag, Switzerland) and held research positions at the University of British Columbia (Canada). He has also served as a distinguished and adjunct professor at multiple institutions in China.

Dr. Wei currently serves as CEO and Chief Scientist of Deepsim Intelligent Technology Inc. (Canada) and Shandong Deepsim Intelligent Technology Co., Ltd. (China). He is also a Postdoctoral Co-Supervisor at the Shandong Postdoctoral Innovation Practice Base and Director of Qilu Artificial Intelligence and Digital Manufacturing Innovation at Shandong Deepsim Intelligent Technology Co., Ltd., China.

Dr. Wei has led or contributed to 19 major international research projects and the development of numerous intelligent systems, including autonomous water-quality monitoring vessels, AI-based environmental early-warning platforms, medical image diagnosis systems, precision agriculture robots, and autonomous service robots.

His scholarly contributions include over 40 peer-reviewed publications, 8 books on practical wavelet transform applications, data analysis, and data modeling, more than 500 technical tutorial articles, six patents, and over 30 software copyrights. His work focuses on making advanced computational methods—particularly data analysis, modeling, and wavelet-based signal processing—accessible, practical, and impactful for researchers and practitioners worldwide.

For more information, visit: https://press.deepsim.ca/shouke/

Contents

Preface

Programming books often follow a familiar path: they start with syntax, move through isolated concepts, and end with small exercises.

This book takes a different approach.

From the very beginning, you will build something real.

Why This Book Exists

Python is one of the most popular programming languages in the world. There are countless tutorials, courses, and books.

Yet many learners still struggle:

- They learn syntax but can't build systems
- They complete exercises but lack real-world experience
- They feel overwhelmed when starting actual projects

This book is designed to solve that problem.

Instead of teaching Python as a collection of features, it teaches Python as a tool for **building intelligent systems**.

What You Will Build

Over fifteen chapters, you will build a complete, working AI companion — from a single `print("Hello, Python!")` to a voice-enabled, memory-persistent, emotion-aware application that connects to real AI models and runs asynchronously.

Along the way, you will:

- write Python programs that grow from a single script to a modular, multi-file system
- connect your companion to the OpenAI API and give it a custom personality
- build a persistent memory system that survives between sessions
- automate real file-system tasks with a few lines of code
- fetch live data from web APIs
- detect emotion in text and adapt your companion's responses accordingly
- capture audio from a microphone and play spoken replies through a speaker
- access a webcam and analyse what the camera sees
- structure, package, and share a professional-quality Python project

By the final chapter, you will not just have learned Python. You will have built something that demonstrates real-world skills — something you can show, share, and extend.

Beyond This Book

The system you build here is only the beginning.

You can extend it into:

- a personal assistant
- a productivity tool
- a web application
- an AI-powered product
- a physical robot (like the XiaoDi companion)

This book gives you the foundation.

What you build next is up to you.

Who This Book Is For

This book is designed for:

- absolute beginners who has little or no prior programming experience
- beginners who want to learn by building something meaningful with it
- learners who know some programming but feel stuck
- developers who want to quickly move into AI applications

This book assumes you know how to use a computer, open a terminal, and follow instructions — but no prior programming or AI experience is required. Every concept is explained clearly before it appears in code.

If you already know some Python, you can move quickly through the early chapters and focus on AI, async programming, vision, and packaging. If you are new, type the examples yourself instead of copying and pasting — the practice matters.

Most importantly, be willing to experiment and learn through trial and error. If you can think logically and keep trying, you can complete this book.

How to Use This Book

Read it in order — each chapter builds on the previous one, and later code depends on earlier concepts.

Type the code — type the code yourself instead of copying and pasting. Writing the code helps you notice details, build familiarity, and learn faster.

Break things and fix them - when errors happen, read the error messages carefully before searching for answers. Python usually tells you exactly what went wrong and where.

Build your own features - complete the exercises and experiment with your own features. Small modifications and hands-on practice are where real learning happens.

This book uses Python 3.14 and the OpenAI API with the gpt-4.1-mini model, but newer versions should work with little or no change. Check the companion repository for updates.

Final Thought

Learning programming is not about memorizing syntax.

It is about creating something meaningful.

By the time you finish this book, you won't just know Python.

👉 You will have built something real.

Shouke Wei, PhD
Deepsim Intelligent Technology Inc.
Deepsim Academy
Abbotsford, Canada
May 8, 2026

Acknowledgments

This book was inspired and shaped by many years of teaching Python programming, data analysis, modeling, and wavelet transforms. I would like to sincerely thank all of my students whose curiosity, questions, feedback, and enthusiasm continuously motivated me to improve the way I teach and explain complex ideas. Many of the practical examples and teaching approaches in this book were refined through real classroom and project experiences.

I am deeply grateful to the open-source community and the developers behind Python and its ecosystem. Their contributions have made modern programming, artificial intelligence, data science, and automation more accessible to learners around the world.

I would also like to thank the researchers, educators, and innovators working in artificial intelligence, data science, computer vision, speech technology, and intelligent systems. Their work continues to inspire new possibilities for teaching and building real-world AI applications.

Most importantly, I would like to thank my family for their support, patience, encouragement, and understanding throughout the writing of this book and many related projects. Their support made this journey possible.

Finally, thank you to every reader and learner who chooses to build, explore, and create with technology. I hope this book helps you move beyond learning syntax and toward building meaningful and intelligent systems.

0 Absolute Beginner Jumpstart

This chapter gets you from zero to your first working Python program — quickly and confidently.

By the end, you will:

- understand what Python is and why it matters
- install Python on your computer
- write and run your very first program
- choose a comfortable coding environment
- Structure the project
- learn programming in the code environment

The goal is simple: **build first, learn as you go.** Do not worry about understanding everything right away. Just follow the steps.

0.1 What Is a Program?

A **program** is a set of instructions you give to a computer. The computer follows them one step at a time, in order, and produces a result.

Think of it like a recipe:

- A recipe says: *crack two eggs, add flour, mix, bake.*
- A program says: *get the user's name, say hello, wait for input.*

The computer does exactly what you tell it — no more, no less. Your job as a programmer is to write those instructions clearly.

0.2 Why Python?

Python is one of the most popular programming languages in the world. It is used in:

- **data science** — analysing large amounts of information
- **AI and machine learning** — teaching computers to make decisions
- **automation** — making repetitive tasks run themselves
- **web development** — building websites and online tools

Python is a great first language because:

- its syntax reads almost like plain English
- it has a huge collection of ready-made libraries
- it has a large, friendly community
- errors are usually easy to understand and fix

You do **not** need any prior programming experience. If you can follow steps and try things out, you can learn Python.

0.3 Install and Run Python

This book uses the latest version available at the time of writing, Python 3.14.4. You can install any version of Python 3.12 or later depending on your computer and operating system.

0.3.1 Install Python

1. Go to https://www.python.org/downloads/
2. Download **Python 3.12** or a newer version
3. Run the installer
4. ✔ On Windows, tick **"Add Python to PATH"** before clicking Install — this is important (Figure 1)
5. Click **Install**

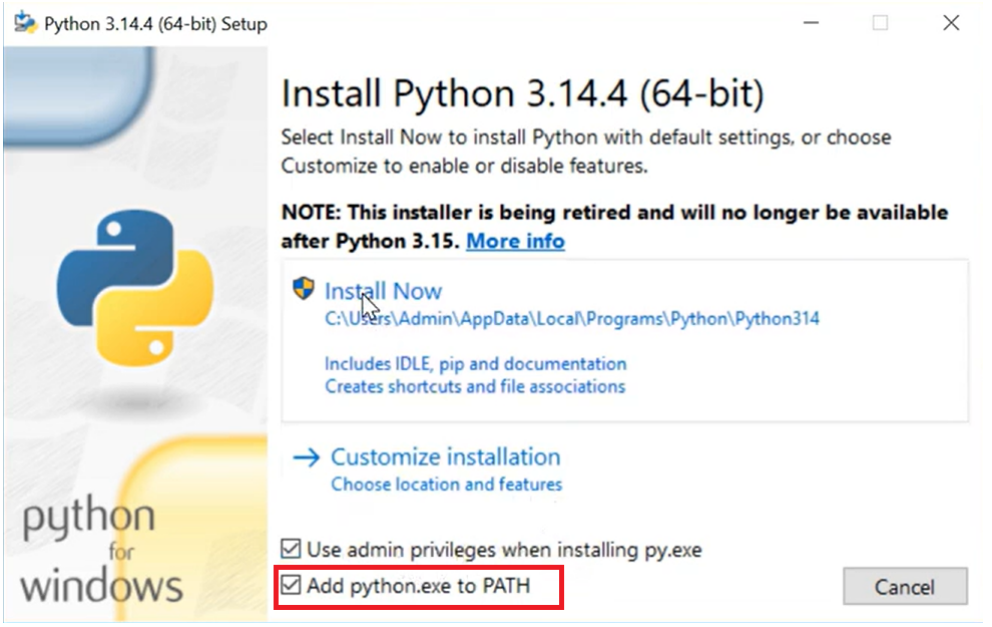


Figure 1: Screenshot of the Python installation process, highlighting the option to tick "Add Python to PATH" before clicking Install Now.

> **Tip**
>
> If you are on macOS or Linux, Python may already be installed. Open a terminal and run `python3 --version` to check.

0.3.2 Check the Installation

Open a terminal window — this is called **Command Prompt** on Windows or **Terminal** on macOS and Linux — and type:

```
python --version
```

You should see something like (Figure 2):

```
Python 3.1x.x
```

For example, on my computer:

```
Python 3.14.4
```

If you see that, Python is installed and ready. If you see an error, revisit step 4 and make sure you ticked "Add Python to PATH".

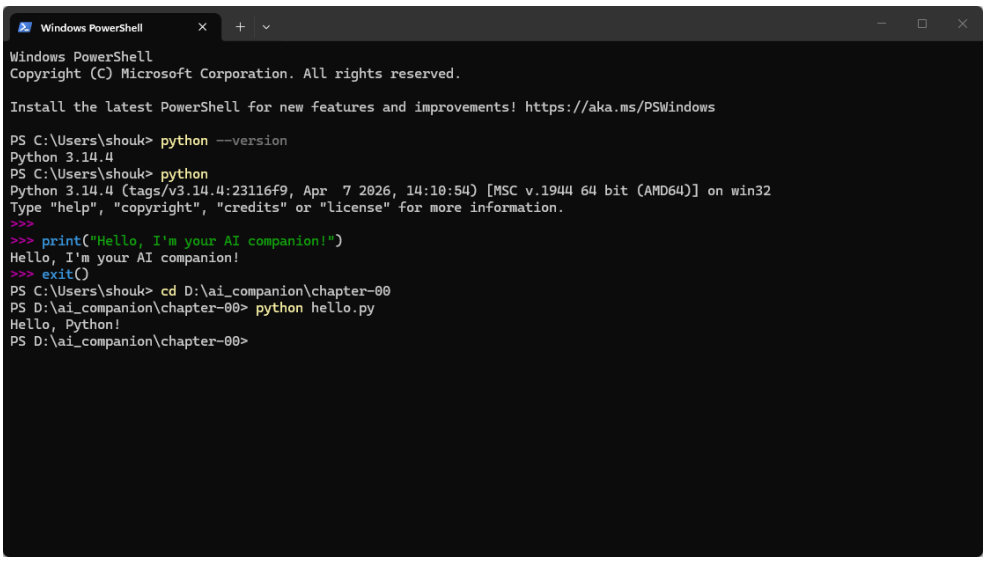


Figure 2: Screenshot of running Python, including checking the Python version, using the interactive shell, exiting Python, and running a Python program.

0.3.3 Try the Interactive Shell

Type this in your terminal and press Enter:

```
python
```

You will see a prompt that looks like this (Figure 2):

```
>>>
```

This is the **Python interactive shell** (also called a REPL — Read, Evaluate, Print, Loop). You can type Python code here and see the result immediately. It is a great way to experiment.

Type the following and press Enter (Figure 2):

```
print("Hello, I'm your AI companion!")
```

Output:

```
Hello, I'm your AI companion!
```

`print()` is one of the most commonly used functions in Python. It takes whatever you put inside the parentheses and displays it on the screen.

To leave the interactive shell, type `exit()` and press Enter (Figure 2).

0.4 Your First Program

The interactive shell (Section 0.3.3) is useful for quick experiments, but it does not save your work. When you close it, everything is gone. The standard way to write programs is to save your code in a **file**.

Python files end with `.py`. Let's create your first one.

Open any plain text editor (Notepad on Windows works fine for now) and type:

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

Save the file as `hello.py`.

Now, back in your terminal, navigate to the folder where you saved the file (e.g. `D:\ai_companion\chapter-00`, see Section 0.7) and run (see Figure 2):

```
python hello.py
```

You should see:

```
Hello, Python!
```

Congratulations — you have just written and run your first Python program.

0.5 What Just Happened?

Here is the core idea of every program you will ever write:

👉 **Write code → Run it → See the output**

When you ran `python hello.py`, Python read your file line by line, followed your instructions, and printed the text to the screen. Everything else you learn in this book builds on that simple loop.

0.6 Choose a Coding Environment

A plain text editor works, but a proper **code editor** or **IDE** (Integrated Development Environment) makes your life much easier. IDEs offer features like:

- colour-coding that makes code easier to read (syntax highlighting)
- automatic suggestions as you type (code completion)
- built-in tools for finding and fixing errors (debugging)

You only need to learn the basics right now: how to create a file, write code, and run it.

0.6.1 Recommended: Visual Studio Code

Visual Studio Code (VS Code) is free, beginner-friendly, and used by professionals worldwide. It works on Windows, macOS, and Linux.

1. Download and install VS Code: https://code.visualstudio.com/
2. Open VS Code
3. Click the **Extensions** icon like four squares in the left sidebar (Figure 3)
4. Search for **Python** and install the extension published by Microsoft

That is all you need for now. Do not worry about any other settings.

The screenshot below shows the VS Code interface with the key areas labelled:

> **Enable Auto-Save**
>
> To avoid manually saving files each time, enable Auto-Save in VS Code by clicking **File > Auto Save**. Once checked, your files will be saved automatically as you type.

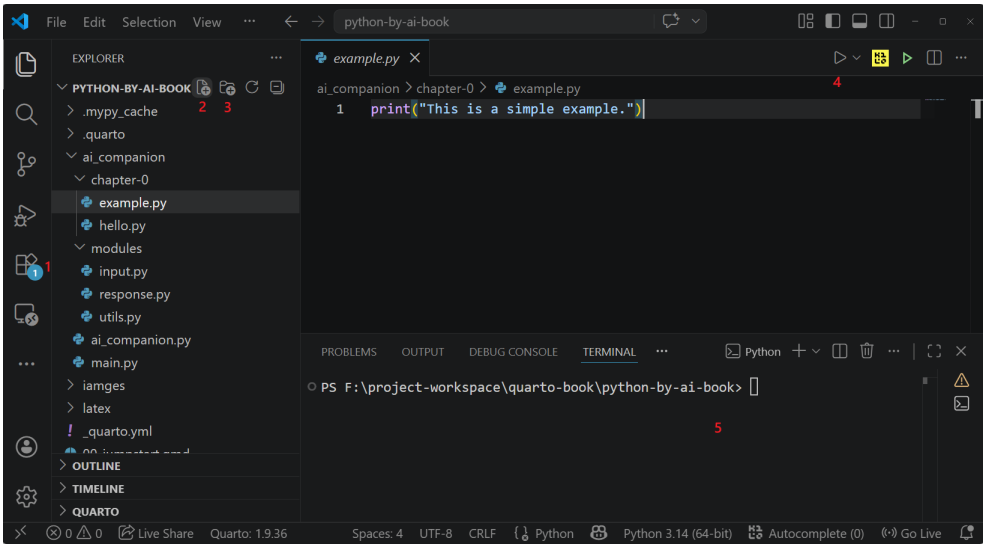


Figure 3: Screenshot of VS Code interface with the key areas labelled: 1. Extensions icon, 2. New file button, 3. Project folder panel, 4. Run Python file button, and 5. Integrated terminal.

0.6.2 Beginner-Friendly Alternative: Thonny

If VS Code feels overwhelming, try **Thonny** (https://thonny.org). It is designed specifically for beginners, has a very simple interface, and comes with Python bundled — so there is less to install.

0.7 Suggested Project Structure

0.7.1 Project Structure

As you work through this book, it helps to keep your files organised. Here is a simple folder structure to follow:

```
ai_companion/
├── chapter-00/
│   ├── hello.py
│   └── example.py
├── chapter-01/
│   └── ai_companion.py
```

```
├── chapter-02/
│   ├── main.py
│   └── modules/
│       ├── input.py
│       ├── response.py
│       └── utils.py
└── ...
```

Create a folder called `ai_companion` somewhere on your computer. Inside it, create a sub-folder for each chapter as you go. This keeps everything tidy and makes it easy to find your work later.

0.7.2 Creating Project Structure with VS Code

VS Code makes it easy to set up your project structure (see Figure 4) and for the process using buttons (see Figure 3).

1. Open VS Code and click **File > Open Folder...** to open the `ai_companion` folder you created.
2. Click **Terminal > New Terminal** to open an integrated terminal.
3. In the terminal, type `mkdir chapter-00` to create the Chapter 0 folder — or use the **New Folder** button in the Explorer panel.
4. Navigate into the folder with `cd chapter-00`.
5. Create a new Python file by typing `code hi.py` in the terminal — or click the **New File** button in the Explorer panel.
6. Save the file via **File > Save**, or press `Ctrl+S`.
7. Run the `.py` file directly in the VS Code terminal by typing `python hi.py`, or click the **Run Python File** button in the top-right corner when the file is open in the editor.
8. Type `cd ..` to return to the parent folder `ai_companion`, then repeat the process to create folders for the remaining chapters.

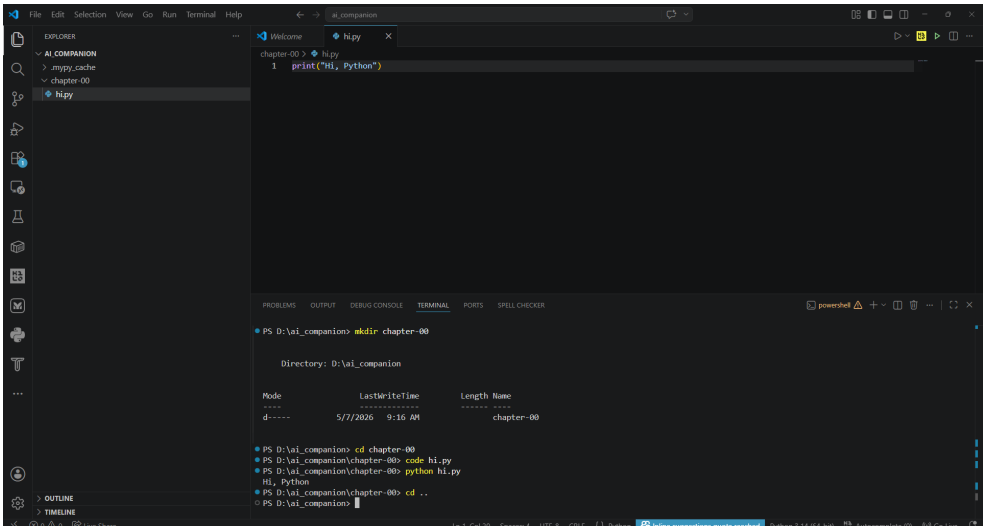


Figure 4: Screenshot showing the process of using VS Code to create a project structure, navigate into a folder, create and run `.py` files, and return to the parent folder.

0.8 Companion Resources

The structured folders and scripts used in this book are available online.

Official resource hub: https://press.deepsim.ca/ai-companion

GitHub repository (code and reproducible workflows): https://github.com/shoukewei/ai-companion

The repository contains:

- all structured folders used throughout the chapters
- all scripts developed throughout the chapters
- reproducible examples and workflows
- supporting project files and configurations

0.9 Mini Exercise

Open `hello.py` in your editor and replace the contents with this:

```
print("Hello, AI!")
print("I am learning Python.")
```

Save the file in the chapter-00 folder, and run it again:

```
python hello.py
```

You should see two lines of output. Notice that each `print()` call produces one line. You are already writing real programs.

0.10 A Quick Note on Mistakes

You will make mistakes. Every programmer does — including experienced ones. When Python shows an error message, it is not scolding you. It is giving you a clue about what went wrong and where.

Read the error message carefully. It usually tells you the line number and the type of problem. You will learn to read these messages quickly, and they will become your best debugging tool.

0.11 What's Next

Many beginners stop before they start because setup feels confusing. You just crossed that barrier.

In the next chapter, you will build your first AI companion in under 30 minutes — no theory overload, just building.

👉 **Keep going. The best way to learn Python is to write Python.**

Part I

Build Something on Day One

1 Your First AI Companion

In this chapter you will build your **first working AI companion** — a real program that listens to what you type and responds intelligently. You will have it running in about 30 minutes.

By the end, your program will behave like this:

```
AI Companion started. Type 'quit' to exit.

You: hello
AI: Hi! Nice to meet you.

You: how are you
AI: I'm doing great. Thanks!

You: quit
AI: Goodbye!
```

It is simple — and that is the point. You will understand every line of code before the chapter is done, and you will know exactly how to extend it.

1.1 Copy and Run the Code

Inside your `chapter-01` folder, create a new file called `ai_companion.py` and paste in the following code:

Index

P

W

Y