GOHOLIC / DIGITAL BOOKS FOR CURIOUS MAKERS

Python: The Art of Making Things

From Your First Line of Code to Building with AI



ABIR SARKAR

GOHOLIC.IN · PYTHON 3.12+ EDITION

GOHOLIC · PRACTICAL COMPUTING

Python: The Art of Making Things

From Your First Line of Code
to Building with AI

Abir Sarkar

FIRST DIGITAL EDITION · GOHOLIC

Python: The Art of Making Things
From Your First Line of Code to Building with AI

© 2026 Abir Sarkar and Goholic. All rights reserved.
Published digitally by Goholic · goholic.in

Readers may run, modify and adapt the code listings for personal study, classroom instruction and independent projects.

Python 3.12+ edition · November 2026
Book design and typesetting: Goholic Digital Editions.

Contents

From a first instruction to three useful projects, with practice at every stage.

01	What a Program Actually Does	7
02	Running Your First Experiments in Python 3.12	14
03	Values, Variables, and Program State	21
04	Text, Input, and Conversation with a Program	26
05	Comparisons, Booleans, and Branching	32
06	Compound Decisions and Edge Cases	38
07	Loops, Counters, and Accumulators	44
08	Indefinite Loops, Stopping Conditions, and State Traces	49
09	Lists, Tuples, and Ordered Data	54
10	Dictionaries, Sets, and Choosing Representations	59
11	Functions, Parameters, and Return Values	65
12	Scope, Decomposition, and Testing Small Tools	69
13	Files, CSV, JSON, and Persistent Data	75
14	Modules, Exceptions, and Systematic Debugging	79
15	Project 1: Personal Expense & Receipt Ledger Automator	85
16	Project 2: Climate & Greenhouse Sensor Investigator	91
17	Project 3: Capstone: Spaced-Repetition Study Deck Engine	95
18	Working with LLMs as a Learning and Development Partner	98
19	Choosing Your Next Direction & 90-Day Continuation Plan	103
APP	Appendix A: Progressive Hints & Selected Solutions (Chapters 1-18)	106
APP	Appendix B: Quick-Reference AI Learning Guide	114
APP	Appendix C: Glossary of Programming Terminology	115
APP	Appendix D: Quick-Reference Python 3.12 Cheat Sheet	117
APP	Appendix E: References and Colophon	118

Turning an Idea Into Something That Works

Every useful computer program begins as a practical question or a recurring annoyance. Perhaps you have forty monthly expense receipts sitting in a folder and you are tired of adding them up on a pocket calculator. Perhaps you are recording temperature measurements from a balcony weather sensor and want to see which afternoons crossed thirty-five degrees Celsius. Perhaps you work with spreadsheets all day and want to know how software actually makes decisions behind the screen.

People who try to learn programming on their own usually run into one of two dead ends. The first dead end is the encyclopedia approach: books that make you memorize long lists of rules, data types, and abstract terminology for a month before you build anything that solves a real problem. The second dead end is the copy-paste shortcut: typing a vague wish into an artificial intelligence chat box, pasting the resulting block of code into a file, and hoping it works. When that generated code crashes or quietly produces the wrong total, a beginner who cannot read code is stuck.

This book takes a third route. We treat programming as the craft of discovering how a problem behaves, expressing that behavior clearly enough for a computer to carry it out, testing our explanation against reality, and building a working tool. You do not need any prior experience with programming or command-line tools. You do not need to want a career as a full-time software engineer, though if you later choose to pursue automation, data analysis, scientific computing, backend development, or artificial intelligence, the foundation you build here will serve you well.

Our Learning Approach: Four Habits That Build Real Competence

The structure of every chapter in this book is grounded in empirical research on how beginners learn to program (Sentance, Waite, & Kallia, 2019; Sorva, 2013; Margulieux et al., 2020). Rather than burying you in academic theory, we have built four concrete habits into the rhythm of the book:

1. Read and Predict Before You Run: Whenever a new program appears, we ask you to pause for fifteen seconds and write down what you expect it to print before you execute it. Even when your prediction turns out to be wrong, the gap between what you expected and what Python actually prints sharpens your understanding faster than passive reading ever could.

2. Trace the Invisible State: When a program runs, values change inside the computer's memory out of sight. Throughout every chapter, we use step-by-step state tables and diagrams to make those memory updates visible on the page.

3. Treat Error Messages as Diagnostic Clues: Every programmer triggers errors daily. An error traceback is a report from Python showing the exact line where your instructions and Python's rules diverged. Starting in Chapter 1, we will trigger errors on purpose so you learn to read them calmly and fix them systematically.

4. Learn Independently First, Then Partner With AI: Eye-tracking studies of novice programmers (Prather et al., 2024) show that turning on automatic AI code-completion popups while you are still learning basic syntax interrupts your concentration and creates a false sense of mastery. While working through Parts I through VIII, keep automatic code completion turned off and write every line with your own fingers. Once you have built a solid mental model of Python, Part IX (Chapter 18) teaches you a seven-mode method for using conversational AI to test your understanding, review your code, and design edge-case tests without becoming dependent on generated answers.

Setting Up Your Workbench (Python 3.12+)

All code in this book follows modern Python 3 (specifically targeting Python 3.12 and newer stable releases). To run the examples on your laptop or desktop computer:

Install Python: Download the official installer for Windows, macOS, or Linux from python.org/downloads. On Windows, check the box labeled "Add python.exe to PATH" on the first installer screen before clicking Install.

Choose a Simple Editor: You can start with IDLE (which comes bundled with Python automatically), Thonny (thonny.org, a free beginner-friendly editor with a built-in variable inspector), or VS Code with inline AI autocomplete disabled during your initial study.

Verify Your Installation: Open your computer's terminal (Terminal on macOS/Linux, or Command Prompt / PowerShell on Windows) and type python3 --version (or python --version on Windows). You should see a version number such as Python 3.12.x.