Preface

Contents

You can order print and ebook versions of *Think Python 3e* from Bookshop.org and Amazon.

Who Is This Book For?

If you want to learn to program, you have come to the right place. Python is one of the best programming languages for beginners – and it is also one of the most in-demand skills.

You have also come at the right time, because learning to program now is probably easier than ever. With virtual assistants like ChatGPT, you don't have to learn alone. Throughout this book, I'll suggest ways you can use these tools to accelerate your learning.

This book is primarily for people who have never programmed before and people who have some experience in another programming language. If you have substantial experience in Python, you might find the first few chapters too slow.

One of the challenges of learning to program is that you have to learn *two* languages: one is the programming language itself; the other is the vocabulary we use to talk about programs. If you learn only the programming language, you are likely to have problems when you need to interpret an error message, read documentation, talk to another person, or use virtual assistants. If you have done some programming, but you have not also learned this second language, I hope you find this book helpful.

Goals of the Book

Writing this book, I tried to be careful with the vocabulary. I define each term when it first appears. And there is a glossary that the end of each chapter that reviews the terms that were introduced.

I also tried to be concise. The less mental effort it takes to read the book, the more capacity you will have for

programming.

But you can't learn to program just by reading a book – you have to practice. For that reason, this book includes exercises at the end of every chapter where you can practice what you have learned.

If you read carefully and work on exercises consistently, you will make progress. But I'll warn you now – learning to program is not easy, and even for experienced programmers it can be frustrating. As we go, I will suggest strategies to help you write correct programs and fix incorrect ones.

Navigating the Book

Each chapter in this book builds on the previous ones, so you should read them in order and take time to work on the exercises before you move on.

The first six chapters introduce basic elements like arithmetic, conditionals, and loops. They also introduce the most important concept in programming, functions, and a powerful way to use them, recursion.

Chapters 7 and 8 introduce strings – which can represent letter, words, and sentences – and algorithms for working with them.

Chapters 9 through 12 introduce Python's core data structures – lists, dictionaries, and tuples – which are powerful tools for writing efficient programs. Chapter 12 presents algorithms for analyzing text and randomly generating new text. Algorithms like these are at the core of large language models (LLMs), so this chapter will give you an idea of how tools like ChatGPT work.

Chapter 13 is about ways to store data in long-term storage – files and databases. As an exercise, you can write a program that searches a file system and finds duplicate files.

Chapters 14 through 17 introduce object-oriented programming (OOP), which is a way to organize programs and the data they work with. Many Python libraries are written in object-oriented style, so these chapters will help you understand their design – and define your own objects.

The goal of this book is not to cover the entire Python language. Rather, I focus on a subset of the language that provides the greatest capability with the fewest concepts. Nevertheless, Python has a lot of features you can use to solve common problems efficiently. Chapter 18 presents some of these features.

Finally, Chapter 19 presents my parting thoughts and suggestions for continuing your programming journey.

What's new in the third edition?

The biggest changes in this edition were driven by two new technologies – Jupyter notebooks and virtual assistants.

Each chapter of this book is a Jupyter notebook, which is a document that contains both ordinary text and code. For me, that makes it easier to write the code, test it, and keep it consistent with the text. For you, it means you can run the code, modify it, and work on the exercises, all in one place. Instructions for working with

the notebooks are in the first chapter.

The other big change is that I've added advice for working with virtual assistants like ChatGPT and using them to accelerate your learning. When the previous edition of this book was published in 2016, the predecessors of these tools were far less useful and most people were unaware of them. Now they are a standard tool for software engineering, and I think they will be a transformational tool for learning to program – and learning a lot of other things, too.

The other changes in the book were motivated by my regrets about the second edition.

The first is that I did not emphasize software testing. That was already a regrettable omission in 2016, but with the advent of virtual assistants, automated testing has become even more important. So this edition presents Python's most widely-used testing tools, `doctest` and `unittest`, and includes several exercises where you can practice working with them.

My other regret is that the exercises in the second edition were uneven – some were more interesting than others and some were too hard. Moving to Jupyter notebooks helped me develop and test a more engaging and effective sequence of exercises.

In this revision, the sequence of topics is almost the same, but I rearranged a few of the chapters and compressed two short chapters into one. Also, I expanded the coverage of strings to include regular expressions.

A few chapters use turtle graphics. In previous editions, I used Python's `turtle` module, but unfortunately it doesn't work in Jupyter notebooks. So I replaced it with a new turtle module that should be easier to use.

Finally, I rewrote a substantial fraction of the text, clarifying places that needed it and cutting back in places where I was not as concise as I could be.

I am very proud of this new edition – I hope you like it!

Getting started

For most programming languages, including Python, there are many tools you can use to write and run programs. These tools are called integrated development environments (IDEs). In general, there are two kinds of IDEs:

- Some work with files that contain code, so they provide tools for editing and running these files.
- Others work primarily with notebooks, which are documents that contain text and code.

For beginners, I recommend starting with a notebook development environment like Jupyter.

The notebooks for this book are available from an online repository at https://allendowney.github.io/ThinkPython.

There are two ways to use them:

- You can download the notebooks and run them on your own computer. In that case, you have to install Python and Jupyter, which is not hard, but if you want to learn Python, it can be frustrating to spend a lot of time installing software.
- An alternative is to run the notebooks on Colab, which is a Jupyter environment that runs in a web browser, so you don't have to install anything. Colab is operated by Google, and it is free to use.

If you are just getting started, I strongly recommend you start with Colab.

Resources for Teachers

If you are teaching with this book, here are some resources you might find useful.

- You can find notebooks with solutions to the exercises at https://allendowney.github.io/ThinkPython, along with links to the additional resources below.
- Quizzes for each chapter, and a summative quiz for the whole book, are available on request.
- *Teaching and Learning with Jupyter* is an online book with suggestions for using Jupyter effectively in the classroom. You can read the book at https://jupyter4edu.github.io/jupyter-edu-book
- One of the best ways to use notebooks is live coding, where an instructor writes code and students follow along in their own notebooks. To learn about live coding – and get other great advice about teaching programming – I recommend the instructor training provided by The Carpentries, at https://carpentries.github.io/instructor-training

Acknowledgments

Many thanks to Jeff Elkner, who translated my Java book into Python, which got this project started and introduced me to what has turned out to be my favorite language. Thanks also to Chris Meyers, who contributed several sections to *How to Think Like a Computer Scientist*.

Thanks to the Free Software Foundation for developing the GNU Free Documentation License, which helped make my collaboration with Jeff and Chris possible, and thanks to the Creative Commons for the license I am using now.

Thanks to the developers and maintainers of the Python language and the libraries I used, including the Turtle graphics module; the tools I used to develop the book, including Jupyter and JupyterBook; and the services I used, including ChatGPT, Copilot, Colab and GitHub.

Thanks to the editors at Lulu who worked on *How to Think Like a Computer Scientist* and the editors at O'Reilly Media who worked on *Think Python*.

Special thanks to the technical reviewers for the second edition, Melissa Lewis and Luciano Ramalho, and for the third edition, Sam Lau and Luciano Ramalho (again!). I am also grateful to Luciano for developing the turtle graphics module I use in several chapters, called `jupyturtle`.

Thanks to all the students who worked with earlier versions of this book and all the contributors who sent in

corrections and suggestions. More than 100 sharp-eyed and thoughtful readers have sent in suggestions and corrections over the past few years. Their contributions, and enthusiasm for this project, have been a huge help.

If you have a suggestion or correction, please send email to `feedback@thinkpython.com`. If you include at least part of the sentence the error appears in, that makes it easy for me to search. Page and section numbers are fine, too, but not quite as easy to work with. Thanks!

Think Python: 3rd Edition