about this book

Who should read this book

After you read this book and write all the programs, I would hope that you will be a zealot for creating programs that are documented, tested, and reproducible.

I think my ideal reader is someone who's been trying to learn to code well but isn't quite sure how to level up. Perhaps you are someone who's been playing with Python or some other language that has a similar syntax, like Java(Script) or Perl. Maybe you've cut your teeth on something really different, like Haskell or Scheme, and you're wondering how to translate your ideas to Python. Maybe you've been writing Python for a while and are looking for interesting challenges with enough structure to help you know when you're moving in the right direction.

This is a book that will teach you to write well-structured, documented, testable code in Python. The material introduces best practices from industry such as *test-driven development*—that's when the *tests* for a program exist even before the program itself is written! I will show you how to read documentation and Python Enhancement Proposals (PEPs) and how to write idiomatic code that other Python programmers would immediately recognize and understand.

This is probably not an ideal book for the absolute beginning programmer. I assume no prior knowledge of the Python language specifically, because I'm thinking of someone who is coming from another language. If you've never written a program in *any* language at all, you might do well to come back to this material when you are comfortable with ideas like variables, loops, and functions.

How this book is organized: A roadmap

The book is written with chapters building on previous chapters, so I really recommend you start at the beginning and work sequentially through the material.

- Every program uses command-line arguments, so we start off discussing how to use `argparse` to handle this. Every program is also tested, so you'll have to learn how to install and use `pytest`. The introduction and chapter 1 will get you up and running.
- Chapters 2–4 discuss the basic Python structures like strings, lists, and dictionaries.
- Chapters 5 and 6 move into how we can work with files as input and output and how files are related to "standard in" and "standard out" (`STDIN/STDOUT`).
- Chapters 7 and 8 start combining ideas so you can write more complicated programs.
- Chapters 9 and 10 introduce the `random` module and how to control and test random events.
- In chapters 11–13 you'll learn more about compartmentalizing code into functions and how to write and run tests for them.
- In chapters 14–18 we'll start digging into denser topics like higher-order functions as well as regular expressions to find patterns of text.
- In chapters 19–22 we'll start writing more complex, "real-world" programs that will put all your skills together while pushing your knowledge of the Python language and testing.

About the code

Every program and test shown in the book can be found at https://github.com/kyclark/tiny_python_projects.

Software/hardware requirements

All the program were written and tested with Python 3.8, but version 3.6 would be sufficient for almost every program. Several additional modules are required, such as `pytest` for running the tests. There are instructions for how to use the `pip` module to install these.

iiveBook discussion forum

Purchase of *Tiny Python Projects* includes free access to a private web forum run by Manning Publications where you can make comments about the book, ask technical questions, and receive help from the author and from other users. To access the forum, go to https://livebook.manning.com/book/tiny-python-projects/welcome/v-6. You can also learn more about Manning's forums and the rules of conduct at https://livebook.manning.com/#!/discussion.

Manning's commitment to our readers is to provide a venue where a meaningful dialogue between individual readers and between readers and the author can take

place. It is not a commitment to any specific amount of participation on the part of the author, whose contribution to the forum remains voluntary (and unpaid). We suggest you try asking him some challenging questions lest his interest stray! The forum and the archives of previous discussions will be accessible from the publisher's website as long as the book is in print.

Other online resources

One element missing from many programming courses is a demonstration of how one can go from having no program to having one that works. In my classroom teaching, I spend a lot of time showing students how to start writing a program and then how to work through the process of adding and testing new features. I've recorded videos for each chapter and shared them at www.youtube.com/user/kyclark. There is a playlist for each chapter, and the videos follow the pattern of each chapter by introducing the problem and the language features you might use to write your program, followed by a discussion of the solution(s).