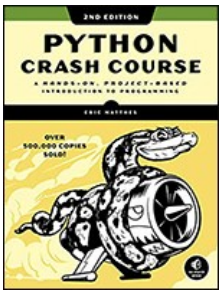

Python Crash Course: A Hands-On, Project-Based Introduction to Programming, 2nd Edition

by Eric Matthes

Introduction

Overview

Every programmer has a story about how they learned to write their first program. I started programming as a child when my father was working for Digital Equipment Corporation, one of the pioneering companies of the modern computing era. I wrote my first program on a kit computer that my dad had assembled in our basement. The computer consisted of nothing more than a bare motherboard connected to a keyboard without a case, and its monitor was a bare cathode ray tube. My initial program was a simple number guessing game, which looked something like this:

```
I'm thinking of a number! Try to guess the number I'm thinking of: 25
Too low! Guess again: 50
Too high! Guess again: 42
That's it! Would you like to play again? (yes/no) no
Thanks for playing!
```

I'll always remember how satisfied I felt watching my family play a game that I created and that worked as I intended it to.

That early experience had a lasting impact. There is real satisfaction in building something with a purpose, something that solves a problem. The software I write now meets a more significant need than my childhood efforts, but the sense of satisfaction I get from creating a program that works is still largely the same.

Who is This Book for?

The goal of this book is to bring you up to speed with Python as quickly as possible so you can build programs that work—games, data visualizations, and web applications—while developing a foundation in programming that will serve you well for the rest of your life. *Python Crash Course* is written for people of any age who have never before programmed in Python or have never programmed at all. This book is for those who want to learn the basics of programming quickly so they can focus on interesting projects, and those who like to test their understanding of new concepts by solving meaningful problems. *Python Crash Course* is also perfect for middle school and high school teachers who want to offer their students a project-based introduction to programming. If you're taking a college class and want a friendlier introduction to Python than the text you've been assigned, this book could make your class easier as well.

What Can You Expect to Learn?

The purpose of this book is to make you a good programmer in general and a good Python programmer in particular. You'll learn efficiently and adopt good habits as I provide you with a solid foundation in general programming concepts. After working your way through *Python Crash Course*, you should be ready to move on to more advanced Python techniques, and your next programming language will be even easier to grasp.

In the first part of this book, you'll learn basic programming concepts you need to know to write Python programs. These concepts are the same as those you'd learn when starting out in almost any programming language. You'll learn about different kinds of data and the ways you can store data in lists and dictionaries within your programs. You'll learn to build collections of data and work through those collections in efficient ways. You'll learn to use `while` loops and `if` statements to test for certain conditions so you can run specific sections of code while those conditions are true and run other sections when they're not—a technique that greatly helps you automate processes.

You'll learn to accept input from users to make your programs interactive and to keep your programs running as long as the user is active. You'll explore how to write functions to make parts of your program reusable, so you only have to write blocks of code that perform certain actions once and then use that code as many times as you like. You'll then extend this concept to more complicated behavior with classes, making fairly simple programs respond to a variety of situations. You'll learn to write programs that handle common errors gracefully. After working through each of these basic concepts, you'll write a few short programs that solve some well-defined problems. Finally, you'll take your first step toward intermediate programming by learning how to write tests for your code so you can develop your programs further without worrying about introducing bugs. All the information in Part I will prepare you for taking on larger, more complex projects.

In Part II, you'll apply what you learned in Part I to three projects. You can do any or all of these projects in whichever order works best for you. In the first project (Chapters 12–14), you'll create a *Space Invaders*–style shooting game called *Alien Invasion*, which consists of levels of increasing difficulty. After you've completed this project, you should be well on your way to being able to develop your own 2D games.

The second project (Chapters 15–17) introduces you to data visualization. Data scientists aim to make sense of the vast amount of information available to them through a variety of visualization techniques. You'll work with data sets that you generate through code, data sets that you download from online sources, and data sets your programs download automatically. After you've completed this project, you'll be able to write programs that sift through large data sets and make visual representations of that stored information.

In the third project (Chapters 18–20), you'll build a small web application called Learning Log. This project allows you to keep a journal of ideas and concepts you've learned about a specific topic. You'll be able to keep separate logs for different topics and allow others to create an account and start their own journals. You'll also learn how to deploy your project so anyone can access it online from anywhere.

Online Resources

You can find all the supplementary resources for the book online at *https://nostarch.com/pythoncrashcourse2e/* or *http://ehmatthes.github.io/pcc_2e/*. These resources include:

Setup instructions These instructions are identical to what's in the book but include active links you can click for all the different pieces. If you're having any setup issues, refer to this resource.

Updates Python, like all languages, is constantly evolving. I maintain a thorough set of updates, so if anything isn't working, check here to see whether instructions have changed.

Solutions to exercises You should spend significant time on your own attempting the exercises in the "Try It Yourself" sections. But if you're stuck and can't make any progress, solutions to most of the exercises are online.

Cheat sheets A full set of downloadable cheat sheets for a quick reference to major concepts is also online.

Why Python?

Every year I consider whether to continue using Python or whether to move on to a different language—perhaps one that's newer to the programming world. But I continue to focus on Python for many reasons. Python is an incredibly efficient language: your programs will do more in fewer lines of code than many other languages would require. Python's syntax will also help you write "clean" code. Your code will be easy to read, easy to debug, and easy to extend and build upon compared to other languages.

People use Python for many purposes: to make games, build web applications, solve business problems, and develop internal tools at all kinds of interesting companies. Python is also used heavily in scientific fields for academic research and applied work.

One of the most important reasons I continue to use Python is because of the Python community, which includes an incredibly diverse and welcoming group of people. Community is essential to programmers because programming isn't a solitary pursuit. Most of us, even the most experienced programmers, need to ask advice from others who have already solved similar problems. Having a well-connected and supportive community is critical in helping you solve problems, and the Python community is fully supportive of people like you who are learning Python as your first programming language.

Python is a great language to learn, so let's get started!