Python® Projects

Laura Cassell, Alan Gauld

PYTHON® PROJECTS

Python® Projects

Python® Projects

Laura Cassell
Alan Gauld

A Wiley Brand

Python® Projects

Published by
John Wiley & Sons, Inc.
10475 Crosspoint Boulevard
Indianapolis, IN 46256
www.wiley.com

Published simultaneously in Canada

ISBN: 978-1-118-90866-2
ISBN: 978-1-118-90889-1 (ebk)
ISBN: 978-1-118-90919-5 (ebk)

Manufactured in the United States of America

10 9 8 7 6 5 4 3 2 1

For my truly great boys—Nathan, Ben, and Matt:
We will do so many things now that I'm not writing.
Thank you for all the quiet time while I was writing;
you can come out of your rooms now.

—Laura Cassell

To my wife, Heather, for her continued support and
patience with my eccentric working hours.

—Alan Gauld

ABOUT THE AUTHORS

LAURA CASSELL has been poking at code on the web since 1997. She taught herself Perl in the early 2000s where she discovered that programming materials were in dire need of an overhaul and the barrier to entry to teach programming was incredibly high. Thus, her journey to learn programming so she can bring it to other people began.

Originally from Atlanta, GA, Laura founded PyLadies Atlanta, and got her start teaching Python and JavaScript for Big Nerd Ranch. She has since moved on to engineering management and currently resides in Portland, OR where she manages a team of Pythonistas doing software analytics for New Relic, Inc. She still volunteers for teaching and speaking gigs when time permits.

ALAN GAULD is an enterprise architect with more than 40 years of experience in IT, working mainly in the telecommunications and customer service areas. He has programmed in more than 20 languages, producing everything from mainframe billing systems through embedded micro-controllers. For the last 15 years, his preference has been for Python. He is the author of a beginner's book on Python and is co-moderator of the python-tutor mailing list.

When not programming, he likes climbing, backpacking, and skiing. He is also a keen photographer, artist, and acoustic music enthusiast. He lives in Scotland with his wife, Heather.

ABOUT THE TECHNICAL EDITORS

ALEX BRADBURY is a compiler hacker, Linux geek, and Free Software enthusiast. He has been a long time contributor to the Raspberry Pi project and also co-authored *Learning Python with Raspberry Pi*. He is currently a researcher at the University of Cambridge Computer Laboratory and is a co-founder of lowRISC, a non-profit project to produce a complete open-source System-on-Chip (SoC).

TODD SHANDELMAN fondly remembers coding assembly language programs on punch cards for IBM System/370 mainframes. After occupying various other ecological niches in software technology over the years (C, C++, and Perl, to name but a few), Todd's best days are now spent coding pure Python 2.x and 3.x from the Linux command line. In his spare time he is a professional translator of Russian and Hebrew, specializes in foreign-language typography, and can find his way in the dark around Unicode and UTF-8. Todd earned a Bachelor of Science degree in Business Administration from the University of the State of New York. He lives in Houston, Texas, with his wife and son.

CREDITS

Acquisitions Editor
Jim Minatel

Project Editor
Jennifer Lynn

Technical Editors
Alex Bradbury
Todd Shandelman

Production Editor
Christine Mugnolo

Copy Editor
Kimberly A. Cofer

Production Manager
Kathleen Wisor

Manager of Content Development and Assembly
Mary Beth Wakefield

Marketing Director
David Mayhew

Marketing Manager
Carrie Sherrill

Professional Technology & Strategy Director
Barry Pruett

Business Manager
Amy Knies

Associate Publisher
Jim Minatel

Project Coordinator, Cover
Patrick Redmond

Proofreader
Sarah Kaikini, Word One New York

Indexer
Johnna Dinse

Cover Designer
Wiley

Cover Image
© iStock.com/skodonnell

ACKNOWLEDGMENTS

I'D LIKE TO GIVE a big thank you to Alan Gauld, who helped put so much of this book together, and to Mary James and Jennifer Lynn for making sure this idea came to fruition. It's a better book because of all of you.

Thank you also to the Python community, you are all the most welcoming community that I've ever found. You have created a place where people feel welcome and able to approach everyone, experts to novices alike. Keep up the good work and thank you for allowing me to be a part of this wonderful community.

—Laura Cassell

I'D LIKE TO GIVE a shout out to Laura Cassell for kick-starting this project, Jennifer Lynn for steering us through it, and the Python community for their support over the last 15 years.

—Alan Gauld

CONTENTS

INTRODUCTION

AFTER A CONFERENCE ONE YEAR, an e-mail went around the PyLadies organizers mailing list asking, "Is anyone interested in writing a Python book?" I had kicked around the idea of a programming book for a while. After teaching for a couple years and mentoring at PyLadies and other coding meetups, I realized there was a need for a new, specific sort of programming book. I didn't jump to replying to the e-mail, however. I knew that writing a book would be a big process (boy is it!) and that it would take a lot of time and effort on my part, in the way of working on the weekends and holidays (yep, check!). I also knew that I had a full-time job teaching programming, I was the lead organizer for my local PyLadies chapter in Atlanta, Georgia, and I had children that would soon start asking, "Are you writing this weekend?"

All of the above was true (a little more than I originally thought, actually), but I knew that the book was important. There were so many of my students asking me at the end of class, "Now that I know the basics of Python, what do I do?" My answer was always something along the lines of, "You can get involved in open-source projects!" or "Take the advanced Python class!" But none of those answers satisfied them or me. The answer is, "You have to really start looking for something to work on—a problem to solve, a need that must be met." Because, the only way to really know and understand programming and a programming language is to solve problems with said language.

But then the problem of "But I don't have a problem that really needs to be solved" cropped up. So while I could send my students off to look at open-source projects that do, in fact, need the help, if they didn't understand the technology, they'd be lost and give up. Then the community loses yet another programmer who may have brought interesting things to the table. So, that's when, after lots of talking to friends and family, I realized that this book needed to be written.

WHY WE WROTE THIS BOOK

For all those people who came up to us and asked, over the years, "What can I do now that I understand Python basics? What things can I learn? Where do I go?" That's why we wrote this book.

The most chronic problem in programming books that we've experienced and that others have also felt they experienced is that it goes from "These are the basics of a language" into very deep concepts that only people who hold Computer Science (CS) degrees would understand. And that's not cool. Programming should be open to anyone who is interested. We should all be working toward making the bar into programming a little lower. We feel that Python accomplishes this, but we need to take it a step further and begin to understand how people learn abstract ideas and concepts, to help us help them learn.

Think of programming like learning how to build a house, but only understanding that wood is needed and how the wood works to build a house. You still need to understand structural engineering, electrical, plumbing, ventilation, HVAC, etc. The same concept is true for

programming. Languages just explain the wood being used in a house. There is plenty more that is happening in harmony with the wood, and we want to help you uncover those concepts.

WHO THIS BOOK FOR

This book is not for beginners who want to learn Python. Rather, as a reader of this book, you need to already have some Python programming under your belt. That means you've done some tutorials. You also understand that whitespace matters in Python and that lists are denoted with hard braces (`[ ]`) while dicts (dictionaries) are denoted with curly braces (`{ }`). This book is for those people who are still beginners, but who have completed a tutorial or two—folks who understand the basics of Python, but are interested to learn what all they can do with Python.

"Need is the mother of invention" goes the saying, and when you're learning to program, this is very true. If you need a piece of software that can perform a specific function or task, it's easy to learn a language around that need. You have a need, the language will help you, you learn the language, you solve the problem, you've learned a thing, and you've put it to use immediately. This is awesome and fantastic! However, what if you think programming is interesting, but you're missing the need? What if you don't know what to make? That's where this book comes in.

This book will help you to learn the parts of Python that most people don't think to tell new programmers about. Most of the things covered in this book are tools and technologies that one may only discover when they are faced with working with them. However, for new programmers who don't have a specific problem to solve, learning these tools can be difficult. Most of the time no one thinks to introduce these topics to programmers because they are used so regularly. We hope to take you on a journey through the power of Python and all of its splendor.

You will learn how to make a web app, how to talk to a database using Python libraries, and which system tools can help speed up your workflow, if you're a systems administrator. We will briefly touch on topics such as security and best practices. You'll get an overview of creating graphical user interfaces (GUIs) using Python libraries. We will cover consuming and producing application programming interfaces (APIs) and many other topics that are beneficial to Python programmers.

WHAT YOU WILL LEARN

We hope to take you on a small tour of the basics that are available in the Python ecosystem. We'll introduce you to many concepts that are usually discovered only while working on a problem to solve. While we can't put everything into the context of problems that you may need to solve in the future, we hope that we can illustrate the powerful features of the Python language and the available packages and technology that are available to you, the new Python programmer.

We will start out with a brief "crash course" in Python, in case you've forgotten anything. We'll go over the basics, and then you can decide if you want to read that chapter in its entirety or not. Next, we go over Python as a scripting language. You'll get to get your hands dirty, as it were, by writing small scripts to access parts of your system, using Python. This should illustrate the

very basic power you have with the language. Third, we'll start talking about data, which is what programming is all about—manipulating data. You'll get to dive in and work through some examples using some of the standard libraries that come out of the box with Python. We'll even discuss databases so that you can get a quick intro into those. We want you to see and touch every part of a system that you may come in contact with.

After the first three chapters, you'll dive into desktop applications. While these aren't incredibly popular in Python, it is a feature of the language, and it could be useful down the line in your tenure as a Python programmer. Next, we will step out of the desktop and onto the Internet with Python as a data communicator. You'll learn all about HTTP and the Web and how websites work under the hood. You'll even be able to play with producing and consuming APIs. APIs confuse many new programmers; we hope to have removed much of the mystery with this chapter.

In the final chapters, we'll show more advanced topics in Python, such as how to work with Python in bigger projects, debugging your code, creating testing-harnesses, handling errors, and even creating your own exceptions and exception handlers! Finally, there are appendices for reference while you are going through the book and after, when you're spreading your Python wings and programming.

It is a lot of information, and it's like a huge sandbox of tools and ideas to get you started on your journey with the language. We hope that you try things out and research more on your own time with certain concepts and ideas that interest you. We've included plenty of hands-on exercises to help you try out the concepts as they are presented, as well as some challenge questions in most chapters to help you exercise your newfound knowledge.

WHAT YOU NEED TO USE THIS BOOK

In order to get the most out of this book, we recommend you have a modern computer running Python 3.3 or later, a good text editor that you are comfortable using, an Internet connection (for some parts of the book), and a healthy dose of patience and wonderment. We also recommend that you utilize Internet searching for any problems that may arise. Professional programmers don't actually know it all; they usually only know those problems that they deal with day in and day out. A lot of the time of a professional programmer is spent researching and tracking down why a behavior is occurring. No one should ever feel bad for relying on a Google search to solve a problem. Sometimes, your Googling abilities are just as important as your programming ones.

To work through the examples and projects in this book, you will also need the source code. The source code for the samples is available for download from the Wrox website at:

`www.wrox.com/go/pythonprojects`

CONVENTIONS

To help you get the most from the text and keep track of what's happening, we've used a number of conventions throughout the book.

TRY IT OUT

The *Try It Out* is an exercise you should work through, following the text in the book.

1. They usually consist of a set of steps.
2. Each step has a number.
3. Follow the steps through with your copy of the database.

How It Works

After each *Try It Out*, the code you've typed will be explained in detail.

> ***WARNING*** *Warnings hold important, not-to-be-forgotten information that is directly relevant to the surrounding text.*

> ***NOTE*** *Notes indicate notes, tips, hints, tricks, or asides to the current discussion.*

As for styles in the text:

- We highlight new terms and important words when we introduce them.
- We show keyboard strokes like this: Ctrl+A.
- We show filenames, URLs, and code within the text like so: `persistence.properties`.
- We present code in two different ways:

```
We use a monofont type with no highlighting for most code examples.

We use bold to emphasize code that is particularly important in the present
context or to show changes from a previous code snippet.
```

SOURCE CODE

As you work through the examples in this book, you may choose either to type in all the code manually, or to use the source code files that accompany the book. All the source code used in this book is available for download at `www.wrox.com`. Specifically for this book, the code download is on the Download Code tab at:

`www.wrox.com/go/pythonprojects`