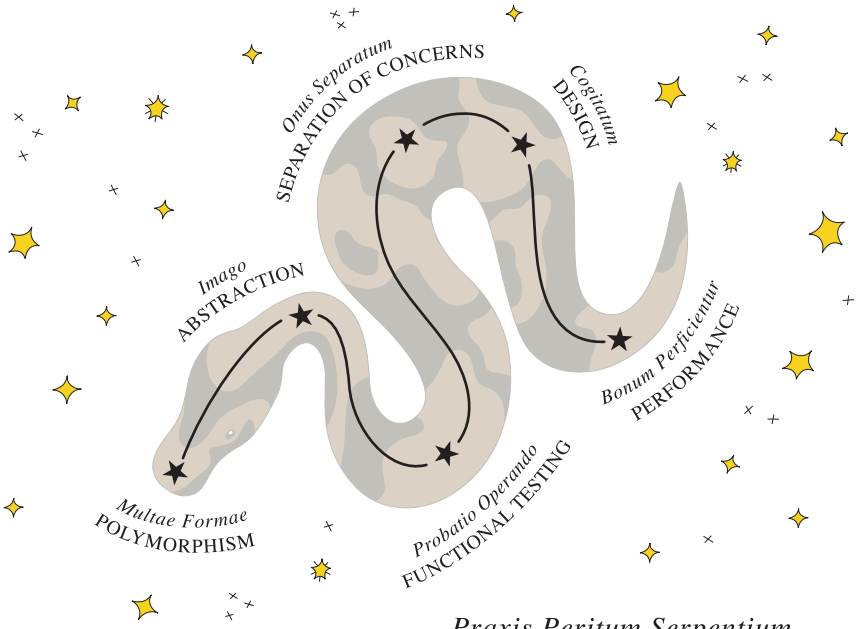
Practices of the Python Pro

Dane Hillard

MANNING



Praxis Peritum Serpentium

PRACTICES OF THE PYTHON PRO

Practices of the Python Pro

Practices of the Python Pro

DANE HILLARD

MANNING
SHELTER ISLAND

For online information and ordering of this and other Manning books, please visit www.manning.com. The publisher offers discounts on this book when ordered in quantity. For more information, please contact

Special Sales Department
Manning Publications Co.
20 Baldwin Road
PO Box 761
Shelter Island, NY 11964
Email: orders@manning.com

Manning Publications Co.
20 Baldwin Road
PO Box 761
Shelter Island, NY 11964

Development editor: Toni Arritola
Technical development editor: Nick Watts
Review editor: Aleks Dragosavljević
Production editor: Lori Weidert
Copy editor: Andy Carroll
Proofreader: Carl Quesnel
Technical proofreader: Jens Christian Bredahl Madson
Typesetter: Gordan Salinovic
Cover designer: Marija Tudor

ISBN 9781617296086
Printed in the United States of America

brief contents

contents

preface

Python, like me, was born in December of 1989. Although I've accomplished a great deal in the subsequent three decades, Python's success is prolific. More people than ever before are picking it up to accomplish fascinating things in data science, machine learning, and more. Since I learned Python, this "second-best language for everything" has in reality been my first choice for many endeavors.

I had a rather traditional path into programming through the Electrical Engineering and Computer Science Department at the University of Michigan. At that time, the coursework focused mainly on C++ and MATLAB—languages I continued to use in my first job out of school. I developed some shell scripting and SQL chops in my next position, processing big data for bioinformatics. I also started using PHP to work on a personal WordPress site from scratch.

Although I was getting results (and cool ones, in some cases), none of the languages I was using *resonated* with me. But I was oblivious. I assumed that programming languages were purely means to an end, and they had little chance of being *fun* to work with. Around this time, a friend invited me to join him in a hackathon project to build a Ruby library.

The world exploded with color, fruits tasted sweeter, and all that. The ease of using an interpreted language and the human-friendly syntax of Ruby really made me think about the tools I'd been using. Although I didn't stick with Ruby for too long, I decided to give Python and the Django web framework a try for the next iteration of my personal site. It gave me the same joy and shallow learning curve I'd seen with Ruby, and I haven't looked back since!

Now that Python is recognized widely as a language of choice for many tasks, folks coming into software development don't need to go through the trial and error process I did. New and interesting pathways into a career in software are opening up all around too. Despite these differences, I hope we can all share in the common experience of finding joy in programming with Python. I also hope this book can contribute to that joy.

Come along on the wonderful Python journey I fell into somewhat haphazardly. I want to see you build a website, a data pipeline, or an automated plant-watering system. Whatever you fancy. Python's got your back. Send photos and code samples of your projects to python-pro-projects@danehillard.com.

acknowledgments

I didn't write this book alone. My appreciation runs deep for everyone who helped me along the way, at every stage and in every capacity. You are loved.

Most anyone who's been involved in the production of a book can tell you that it's always more work than you think. I heard this many times throughout the process, and it certainly was a lot of work. What's not always clear is that the real struggle is balancing all that extra work with your existing life.

To my partner, Stefanie: your support, encouragement, and tolerance of my ranting and raving were paramount in making this book a reality. Thank you for judging my neglect lightly and extricating me from this project during the roughest times. I could not have done this without you.

Thank you to my parents, Kim and Donna, for always funneling my energy toward curiosity, creativity, and compassion.

Thanks to my dear friend Vincent Zhang for spending countless nights at the coffee shop coding by my side. You were there when the concept for this book was born, and your validation helped spur me to take on this endeavor.

Thank you to James Nguyen for persevering as you changed paths to become a developer. You embody the audience for this book, and your input has been invaluable. I'm proud of your accomplishments.

My gratitude goes to all my colleagues at ITHAKA and beyond for your input and support. I thank you for enduring what has undoubtedly been a flighty period for me.

To Toni Arritola, my editor: thank you for your determination in pushing me ever toward higher-quality teaching. The writing process is fraught with many unexpected snags, but you provided me consistency and stability. Thank you.

To Nick Watts, my technical editor: your feedback has pushed the content of this book from frantic ramblings to plausible software teachings. Your candor and insight are much appreciated.

Thank you to Mike Stephens and Marjan Bace at Manning for believing in this idea and trusting me as its shepherd. Thank you to everyone at Manning for working tirelessly to bring authors' ideas to life.

To all the reviewers—Al Krinker, Bonnie Bailey, Burkhard Nestmann, Chris Wayman, David Kerns, Davide Cadamuro, Eriks Zelenka, Graham Wheeler, Gregory Matuszek, Jean-François Morin, Jens Christian Bredahl Madsen, Joseph Perenia, Mark Thomas, Markus Maucher, Mike Stevens, Patrick Regan, Phil Sorensen, Rafael Cassemiro Freire, Richard Fieldsend, Robert Walsh, Steven Parr, Sven Stumpf, and Willis Hampton—your suggestions helped make this a better book.

A final thank you to anyone and everyone else who has had a positive influence—directly, intentionally, or otherwise—on my journey in programming and this book. I cannot hope to produce an exhaustive list; names not appearing here are due expressly to the limitations of my own mind. Thank you to Mark Brehob, Dr. Andrew DeOrio, Jesse Sielaff, Trek Glowacki, everyone at SAIC (in our little Ann Arbor office), everyone at Compendia Bioscience (and friends), Brandon Rhodes, Kenneth Love, Trey Hunner, Jeff Triplett, Mariatta Wijaya, Ali Spittel, Chris Coyier, Sarah Drasner, David Beazley, Dror Ayalon, Tim Allen, Sandi Metz, and Martin Fowler.

about this book

Practices of the Python Pro introduces several concepts that software developers in almost any language can use to improve their work. This would be a great book to read after learning the fundamentals of the Python language.

Who should read this book

Practices of the Python Pro is for anyone in the early stages of their programming journey. In fact, people outside the software industry altogether who use software to supplement their work can find value in this book. The concepts contained in these pages will help readers build software that's more maintainable, which in turn makes their software easier to collaborate on.

In the sciences, reproducibility and provenance are important aspects of the research process. As more research comes to rely on software, code that people can understand, update, and improve is a major consideration. But college curricula are still catching up to this intersection of software with other disciplines. For those with limited experience in formal software development, this book provides a set of principles for producing shareable, reusable software.

If you're seasoned in object-oriented programming and domain-driven design, you may find this book too introductory for your benefit. On the other hand, if you're relatively new to Python, software, or software design, give this book a try. There's something in here for you.

How this book is organized: A roadmap

Practices of the Python Pro consists of 11 chapters in 4 parts. Parts 1 and 2 provide discussion along with short examples and an occasional exercise. Part 3 builds on what you've learned in earlier chapters and contains a variety of exercises. Part 4 provides strategies for learning more, along with recommendations about what to try after reading this book.

Part 1, "Why it all matters," sets the stage for Python's rise to fame and why software design is valuable.

- Chapter 1 covers some recent history of Python and why I enjoy developing Python programs. It goes on to explain software design, why it's important, and how it manifests in your day-to-day work.

Part 2, "Foundations of design," covers the high-level concepts that underpin software design and development.

- Chapter 2 covers separation of concerns, a fundamental activity that provides a basis for several others in the book.
- Chapter 3 explains abstraction and encapsulation, showing you how hiding information and providing simpler interfaces to more complex logic helps you keep a handle on your code.
- Chapter 4 prompts you to think about performance, covering different data structures, approaches, and tools to help you build speedy programs.
- Chapter 5 teaches you about testing your software, using a variety of approaches, from unit testing to end-to-end testing.

Part 3, "Nailing down large systems," walks you through building a real application using the principles you've learned.

- Chapter 6 introduces the application you'll build in the book and provides exercises for creating a program's foundation.
- Chapter 7 covers the concepts of extensibility and flexibility and includes exercises that add extensibility to the application.
- Chapter 8 helps you understand class inheritance, providing recommendations about where and when it should be used. It continues on with exercises that examine inheritance in the application you're building.
- Chapter 9 steps back a bit, introducing tools and an approach for keeping code from growing too large as you go along.
- Chapter 10 explains loose coupling, providing some final exercises to reduce the coupling in the application you're building.

Part 4, "What's next?" gives you some recommendations for how and what to learn next.

- Chapter 11 shows you how I map out new learning material and gives you a few areas of study to try if you're interested in going deeper into software development.