# Preface

As the title promises, this book will introduce you to one of the world's most popular programming languages: Python. It's aimed at beginning programmers as well as more experienced programmers who want to add Python to the languages they already know.

In most cases, it's easier to learn a computer language than a human language. There's less ambiguity and fewer exceptions to keep in your head. Python is one of the most consistent and clear computer languages. It balances ease of learning, ease of use, and expressive power.

Computer languages are made of *data* (like nouns in spoken languages) and *instructions* or *code* (like verbs). You need both. In alternating chapters, you'll be introduced to Python's basic code and data structures, learn how to combine them, and build up to more advanced ones. The programs that you read and write will get longer and more complex. Using a woodworking analogy, we'll start with a hammer, nails, and scraps of wood. Over the first half of this book, we'll introduce more specialized components, up to the equivalents of lathes and other power tools.

You'll not only learn the language, but also what to do with it. We'll begin with the Python language and its "batteries included" standard library, but I'll also show you how to find, download, install, and use some good third-party packages. My emphasis is on whatever I've actually found useful in more than 10 years of production Python development, rather than fringe topics or complex hacks.

Although this is an introduction, some advanced topics are included because I want to expose them to you. Areas like databases and the web are still covered, but technology changes fast. A Python programmer might now be expected to know something about cloud computing, machine learning, or event streaming. You'll find something here on all of these.

Python has some special features that work better than adapting styles

from other languages that you may know. For example, using `for` and *iterators* is a more direct way of making a loop than manually incrementing some counter variable.

When you're learning something new, it's hard to tell which terms are specific instead of colloquial, and which concepts are actually important. In other words, "Is this on the test?" I'll highlight terms and ideas that have specific meaning or importance in Python, but not too many at once. Real Python code is included early and often.

---

NOTE

I'll include a note such as this when something might be confusing, or if there's a more appropriate *Pythonic* way to do it.

---

Python isn't perfect. I'll show you things that seem odd or that should be avoided—and offer alternatives you can use, instead.

Now and then, my opinions on some subjects (such as object inheritance, or MVC and REST designs for the web) may vary a bit from the common wisdom. See what you think.

# Audience

This book is for anybody interested in learning one of the world's most popular computing languages, regardless of whether you have previously learned any programming.

# Changes in the Second Edition

What's changed since the first edition?

- About a hundred more pages, including cat pictures.
- Twice the chapters, each shorter now.
- An early chapter devoted to data types, variables, and names.

- New standard Python features like *f-strings*.
- New or improved third-party packages.
- New code examples throughout.
- An appendix on basic hardware and software, for new programmers.
- An appendix on *asyncio*, for not-so-new programmers.
- "New stack" coverage: containers, clouds, data science, and machine learning.
- Hints on getting a job programming in Python.

What hasn't changed? Examples using bad poetry and ducks. These are evergreen.

# Outline

Part I (Chapters 1–11) explains Python's basics. You should read these chapters in order. I work up from the simplest data and code structures, combining them on the way into more detailed and realistic programs. Part II (Chapters 12–22) shows how Python is used in specific application areas such as the web, databases, networks, and so on; read these chapters in any order you like.

Here's a brief preview of the chapters and appendixes, including some of the terms that you'll run into there:

*Chapter 1, A Taste of Py*
: Computer programs are not that different from directions that you see every day. Some little Python programs give you a glimpse of the language's looks, capabilities, and uses in the real world. You'll see how to run a Python program within its *interactive interpreter* (or *shell*), or from a text *file* saved on your computer.

*Chapter 2, Data: Types, Values, Variables, and Names*
: Computer languages mix data and instructions. Different *types* of data are stored and treated differently by the computer. They may allow their values to be changed (*mutable*) or not (*immutable*). In a Python program, data can be *literal* (numbers like `78`, text *strings* like `"waffle"`) or represented by named *variables*. Python treats

variables like *names*, which is different from many other languages and has some important consequences.

*Chapter 3, Numbers*

This chapter shows Python's simplest data types: *booleans*, *integers*, and *floating-point* numbers. You'll also learn the basic math operations. The examples use Python's interactive interpreter like a calculator.

*Chapter 4, Choose with if*

We'll bounce between Python's nouns (data types) and verbs (program structures) for a few chapters. Python code normally runs a line at a time, from the start to the end of a program. The `if` code structure lets you run different lines of code, depending on some data comparison.

*Chapter 5, Text Strings*

Back to nouns, and the world of text *strings*. Learn how to create, combine, change, retrieve, and print strings.

*Chapter 6, Loop with while and for*

Verbs again, and two ways to make a *loop*: `for` and `while`. You'll be introduced to a core Python concept: *iterators*.

*Chapter 7, Tuples and Lists*

It's time for the first of Python's higher-level built-in data structures: `lists` and `tuples`. These are sequences of values, like LEGO for building much more complex data structures. Step through them with *iterators*, and build lists quickly with *comprehensions*.

*Chapter 8, Dictionaries and Sets*

*Dictionaries* (aka *dicts*) and *sets* let you save data by their values rather than their position. This turns out to be very handy and will be among your favorite Python features.

*Chapter 9, Functions*

Weave the data and code structures of the previous chapters to

compare, choose, or repeat. Package code in *functions* and handle errors with *exceptions*.

*Chapter 10, Oh Oh: Objects and Classes*

The word *object* is a bit fuzzy, but important in many computer languages, including Python. If you've done *object-oriented programming* in other languages, Python is a bit more relaxed. This chapter explains how to use objects and classes, and when it's better to use alternatives.

*Chapter 11, Modules, Packages, and Goodies*

This chapter demonstrates how to scale out to larger code structures: *modules*, *packages*, and *programs*. You'll see where to put code and data, how to get data in and out, handle options, tour the Python Standard Library, and take a glance at what lies beyond.

*Chapter 12, Wrangle and Mangle Data*

Learn to manage (or mangle) data like a pro. This chapter is all about text and binary data, joy with Unicode characters, and *regex* text searching. It also introduces the data types *bytes* and *bytearray*, counterparts of strings that contain raw binary values instead of text characters.

*Chapter 13, Calendars and Clocks*

Dates and times can be messy to handle. This chapter shows common problems and useful solutions.

*Chapter 14, Files and Directories*

Basic data storage uses *files* and *directories*. This chapter shows you how to create and use them.

*Chapter 15, Data in Time: Processes and Concurrency*

This is the first hard-core system chapter. Its theme is data in time—how to use *programs*, *processes*, and *threads* to do more things at a time (*concurrency*). Python's recent *async* additions are mentioned, with details in Appendix C.

*Chapter 16, Data in a Box: Persistent Storage*

Data can be stored and retrieved with basic flat files and directories within filesystems. They gain some structure with common text formats such as CSV, JSON, and XML. As data get larger and more complex, they need the services of *databases*—traditional *relational* ones, and some newer *NoSQL* data stores.

*Chapter 17, Data in Space: Networks*

Send your code and data through space in *networks* with *services*, *protocols*, and *APIs*. Examples range from low-level TCP *sockets*, to *messaging* libraries and queuing systems, to *cloud* deployment.

*Chapter 18, The Web, Untangled*

The *web* gets its own chapter—clients, servers, APIs, and frameworks. You'll *crawl* and *scrape* websites, and then build real websites with *request* parameters and *templates*.

*Chapter 19, Be a Pythonista*

This chapter contains tips for Python developers—installation with `pip` and `virtualenv`, using IDEs, testing, debugging, logging, source control, and documentation. It also helps you to find and install useful third-party packages, package your own code for reuse, and learn where to get more information.

*Chapter 20, Py Art*

People are doing cool things with Python in the arts: graphics, music, animation, and games.

*Chapter 21, Py at Work*

Python has specific applications for business: data visualization (plots, graphs, and maps), security, and regulation.

*Chapter 22, Py Sci*

In the past few years, Python has emerged as a top language for science: math and statistics, physical science, bioscience, and medicine. *Data science* and *machine learning* are notable strengths. This chapter touches on NumPy, SciPy, and Pandas.

*Appendix A, Hardware and Software for Beginning Programmers*

If you're fairly new to programming, this describes how hardware and software actually work. It introduces some terms that you'll keep running into.

*Appendix B, Install Python 3*
: If you don't already have Python 3 on your computer, this appendix shows you how to install it, whether you're running Windows, macOS, Linux, or some other variant of Unix.

*Appendix C, Something Completely Different: Async*
: Python has been adding asynchronous features in different releases, and they're not easy to understand. I mention them as they come up in various chapters, but save a detailed discussion for this appendix.

*Appendix D, Answers to Exercises*
: This has the answers to the end-of-chapter exercises. Don't peek here until you've tried the exercises yourself, or you might be turned into a newt.

*Appendix E, Cheat Sheets*
: This appendix contains cheat sheets to use as a quick reference.

# Python Versions

Computer languages change over time as developers add features and fix mistakes. The examples in this book were written and tested while running Python version 3.7. Version 3.7 was the most current as this book was being edited, and I'll talk about its notable additions. Version 3.8 is scheduled for general release in late 2019, and I'll include a few things to expect from it. If you want to know what was added to Python and when, try the What's New in Python page. It's a technical reference; a bit heavy when you're just starting with Python, but may be useful in the future if you ever have to get programs to work on computers with different Python versions.

# Conventions Used in This Book

The following typographical conventions are used in this book:

*Italic*
: Indicates new terms, URLs, email addresses, filenames, and file extensions.

`Constant width`
: Used for program listings, as well as within paragraphs to refer to program elements such as variables, functions, and data types.

**`Constant width bold`**
: Shows commands or other text that should be typed literally by the user.

*`Constant width italic`*
: Shows text that should be replaced with user-supplied values or by values determined by context.

---

**NOTE**

This icon signifies a tip, suggestion, or general note.

---

---

**WARNING**

This icon indicates a warning or caution.

---

# Using Code Examples

The substantial code examples and exercises in this book are available online for you to download. This book is here to help you get your job done. In general, you may use the code in this book in your programs and documentation. You do not need to contact us for permission unless you're reproducing a significant portion of the code. For example, writing

a program that uses several chunks of code from this book does not require permission. Selling or distributing examples from O'Reilly books does require permission. Answering a question by citing this book and quoting example code does not require permission. Incorporating a significant amount of example code from this book into your product's documentation does require permission.

We appreciate, but do not require, attribution. An attribution usually includes the title, author, publisher, and ISBN. For example: "*Introducing Python* by Bill Lubanovic (O'Reilly). Copyright 2020 Bill Lubanovic, 978-1-492-05136-7."

If you feel your use of code examples falls outside fair use or the permission given here, feel free to contact us at permissions@oreilly.com.

## O'Reilly Online Learning

---

**NOTE**

For over 40 years, *O'Reilly Media* has provided technology and business training, knowledge, and insight to help companies succeed.

---

Our unique network of experts and innovators share their knowledge and expertise through books, articles, conferences, and our online learning platform. O'Reilly's online learning platform gives you on-demand access to live training courses, in-depth learning paths, interactive coding environments, and a vast collection of text and video from O'Reilly and 200+ other publishers. For more information, please visit *http://oreilly.com*.

## How to Contact Us

Please address comments and questions concerning this book to the publisher:

- O’Reilly Media, Inc.
- 1005 Gravenstein Highway North
- Sebastopol, CA 95472
- 800-998-9938 (in the United States or Canada)
- 707-829-0515 (international or local)
- 707-829-0104 (fax)

We have a web page for this book, where we list errata, examples, and any additional information. You can access this page at *https://oreil.ly/introducing-python-2e*.

To comment or ask technical questions about this book, send an email to *bookquestions@oreilly.com*.

For more information about our books, courses, conferences, and news, see our website at *http://www.oreilly.com*.

Find us on Facebook: *http://facebook.com/oreilly*

Follow us on Twitter: *http://twitter.com/oreillymedia*

Watch us on YouTube: *http://www.youtube.com/oreillymedia*

# Acknowledgments

My sincere thanks to the reviewers and readers who helped make this better:

Corbin Collins, Charles Givre, Nathan Stocks, Dave George, and Mike James