Python 3面向对象编程

第2版（影印版）

Python 3 Object-oriented Programming

Second Edition

Dusty Phillips 著

[PACKT] PUBLISHING

东南大学出版社
SOUTHEAST UNIVERSITY PRESS

Python 3 面向对象编程

第 2 版

(影印版)

Dusty Phillips 著

南京　东南大学出版社

图书在版编目（CIP）数据

Python 3面向对象编程：第2版：英文/（加）杜斯提·菲利普（Dusty Phillips）著. —影印本. —南京：东南大学出版社，2017.4

书名原文：Python 3 Object-oriented Programming, Second Edition

ISBN 978-7-5641-7074-5

Ⅰ. ①P… Ⅱ. ①杜… Ⅲ. ①软件工具－程序设计－英文 Ⅳ. ①TP311.561

中国版本图书馆CIP数据核字（2017）第051743号

Python 3面向对象编程 第2版（影印版）

出版发行：东南大学出版社
地　　址：南京四牌楼2号　　邮编：210096
出 版 人：江建中
网　　址：http://www.seupress.com
电子邮件：press@seupress.com
印　　刷：常州市武进第三印刷有限公司
开　　本：787毫米×980毫米　16开本
印　　张：28.5
字　　数：558千字
版　　次：2017年4月第1版
印　　次：2017年4月第1次印刷
书　　号：ISBN 978-7-5641-7074-5
定　　价：88.00元

本社图书若有印装质量问题，请直接与营销部联系。电话（传真）：025-83791830

Credits

Author
Dusty Phillips

Reviewers
AMahdy AbdElAziz
Grigoriy Beziuk
Krishna Bharadwaj
Justin Cano
Anthony Petitbois
Claudio Rodriguez

Commissioning Editor
Edward Gordon

Acquisition Editors
Indrajit Das
Rebecca Pedley
Greg Wild

Content Development Editors
Divij Kotian
Arvind Koul
Anila Vincent

Technical Editor
Siddhi Rane

Copy Editor
Janbal Dharmaraj

Project Coordinator
Neha Bhatnagar

Proofreader
Safis Editing

Indexer
Hemangini Bari

Graphics
Sheetal Aute
Jason Monteiro

Production Coordinator
Komal Ramchandani

Cover Work
Komal Ramchandani

About the Author

Dusty Phillips is a Canadian software developer and author currently living in Seattle, Washington. He has been active in the open source community for a decade and a half and programming in Python for nearly all of it. He cofounded the popular Puget Sound Programming Python meetup group; drop by and say hi if you're in the area.

Python 3 Object Oriented Programming, Packt Publishing, was the first of his books. He has also written *Creating Apps In Kivy, O'Reilly*, the mobile Python library, and self-published *Hacking Happy*, a journey to mental wellness for the technically inclined. He was hospitalized for suicidal tendencies shortly after the first edition of this book was published and has been an outspoken proponent for positive mental health ever since.

About the Reviewers

AMahdy AbdElAziz has more than 8 years of experience in software engineering using several languages and frameworks. Over the last 5 years, he has focused on Android and mobile development, including cross-platform tools, and Android internals, such as building custom ROMs and customizing AOSP for embedded devices.

He is currently teaching Python at Information Technology Institution. You can visit his website, `http://www.amahdy.net/`, to find out more about him.

Grigoriy Beziuk is a former CIO of Crowdage Foundation, acting as an independent software developer as this book was being written. He has worked with a wide variety of programming languages and technologies, including different versions of Python in different environments, ranging from purely scientific ones to modern production-scale web development issues.

> I would like to thank my mom, Larisa Beziuk, for giving me the gift of life; all of my teachers and friends for making my life more interesting; and all of my beloved ones, former and present for… well, everything.

Krishna Bharadwaj is the cofounder of SMERGERS (`https://www.smergers.com/`), a Fintech start-up helping small and medium businesses raise capital from investors and different financial institutions. In the past, he has worked with early stage start-ups such as BlockBeacon (Santa Monica) and PricePoint (CA) and large organizations such as National Instruments, Bangalore, and Google, New York. Krishna got introduced to Python and FOSS during his college days and has continued to use it extensively in his personal projects and also professionally at work. Because of his liking for teaching and mentoring, he visits different universities, conducting workshops whenever he gets an opportunity.

He holds a master's degree in computer science from the University of Southern California, Los Angeles, and a bachelor's degree in information science and engineering from the BMS College of Engineering, Bangalore. He can be reached through his e-mail, `krishna@krishnabharadwaj.info`, or his website, `http://www.krishnabharadwaj.info/`.

Justin Cano is a recent graduate from the University of California, Riverside, with a BS in computer engineering and is currently working as a software engineer in the Silicon Valley area with hopes of moving to a big tech company such as Google or Apple.

He first started programming in the sixth grade, creating small, primitive websites in HTML and CSS. He started to learn computer science theory and C++ in his first year at UC Riverside and then started learning Python in his third year.

Justin admits that at first, he wasn't immediately attracted to Python, since abstractions between C++ and Python are very different. It wasn't until he began to express more of an interest in coding contests and challenges that he began to show interest in Python, mainly because he feels that the readability and elegance of the Python syntax allows him to quickly and more naturally turn ideas and thought processes into Python code. He now writes Python code regularly, often to create mock-ups or prototypes of software applications before moving on to a more domain-specific language.

> I would like to thank the author for taking the time to write this book as I have received a lot of valuable insights and information on the Python language and design patterns. This book has strengthened my understanding of Python, and I believe that I am now a more knowledgeable Python programmer.

Anthony Petitbois is an online architect in the video game industry with 13 years of professional experience in operations and development and more than 20 years of software development experience. He is passionate about new technologies and loves to take creative approaches to solve complex problems.

In his spare time, he learns new languages and new platforms, plays video games, and spends time with his family in the beautiful region of British Columbia, Canada, where he now lives after emigrating from France in 2009.

Claudio Rodriguez started working on PLCs for GE, but his main goal has always been research and development and turning dreams into reality. This made him move from automation engineering to software engineering and the structured way of software, OOD; the remote team working from the comfort of his computer was just too powerful not to take advantage of. During his master's, he got to learn the proper place to look for resources and found a friend in books and research papers and conferences. Eventually, he started working on a system to control an electric arc furnace, but the needs of his clients moved him into taking further control of technology. He has a deep love for complex AI and can be seen surrounded by papers, books, and a computer to test things, but he keeps things real by delivering beautiful and dynamic applications for his customers.

Introduction to the second edition

I have a confession to make. When I wrote the first edition of this book, I didn't have a clue what I was doing. I thought I knew Python and I thought I knew how to write. I quickly learned that this was false. Luckily, I became adept at both by finishing the book!

I was so afraid that people wouldn't like *Python 3 Object Oriented Programming* that I skipped Pycon for two years straight. After a couple dozen positive reviews, my confidence was boosted and I finally attended Pycon 2012 in Santa Clara. I soon discovered that nobody had ever heard of me or my book. So much for arrogance!

I was also afraid to reread the book after completing it. So while it has received many accolades, the copy on my shelf has remained firmly shut, save for when I open it for reference to answer a reader's query. In preparing this second edition, I was finally forced to face my demons. To my surprise and joy, I discovered that the book I wrote five years ago was both accurate and enjoyable, just as many reviewers had suggested.

Shortly after that initial rereading, I got my first ever negative review on Amazon. It would have been devastating had I read it directly after completing the project. Fortunately, four years of good reviews and my own confidence in the writing allowed me to ignore the vitriol and take the remainder as constructive feedback. The truth is many of the flaws the reviewer had pointed out were features at the time the book was originally published. *Python 3 Object Oriented Programming* was showing its age, and it was clearly time for an update. You're holding the result in your hands (or flipping through it on your e-reader).

I've often wondered why authors describe in detail what has changed between the editions of a technical book. I mean, seriously, how many people reading this second edition have read the first one? As with software versions, you safely assume the latest edition is the best, and you don't really care about the project's history. And yet, this project has consumed so much of my life over the past year that I can't leave without a few words about how much better the book has become.

The original book was a little disorganized. Many chapters flowed directly into the next one, but there were a few key places where the topic change was jarring, or worse, irrelevant. The two chapters preceding the discussions about design patterns have been reorganized, reversed, and split into three chapters that flow cleanly into the next topic.

I've also removed an entire chapter on third-party libraries for Python 3. This chapter made more sense when both the book and Python 3 were new. There were only a few libraries that had been ported to Python 3 and it was reasonable to have a best of breed discussion about each of them. However, I was unable to cover any of those topics in detail, and frankly, I could write an entire book on any one of them.

Finally, I've added an entire new chapter on concurrency. I struggled with this chapter and I can freely admit that it's not directly related to object-oriented programming. However, much like the chapter on unit testing, I think that understanding concurrency is an integral part of all programming and especially of object-oriented programming in the Python ecosystem. You are, of course, free to skip those chapters if you disagree (or until you discover a reason to change your mind).

Enjoy the book and your journey into the world of object-oriented programming.

Dusty Phillips

Preface

This book introduces the terminology of the object-oriented paradigm. It focuses on object-oriented design with step-by-step examples. It guides us from simple inheritance, one of the most useful tools in the object-oriented programmer's toolbox through exception handling to design patterns, an object-oriented way of looking at object-oriented concepts.

Along the way, we'll learn to integrate the object-oriented and not-so-object-oriented aspects of the Python programming language. We will learn the complexities of string and file manipulation, emphasizing (as Python 3 does) the difference between binary and textual data.

We'll then cover the joys of unit testing, using not one, but two unit testing frameworks. Finally, we'll explore, through Python's various concurrency paradigms, how to make objects work well together at the same time.

What this book covers

This book is loosely divided into four major parts. In the first four chapters, we will dive into the formal principles of object-oriented programming and how Python leverages them. In chapters 5 through 8, we will cover some of Python's idiosyncratic applications of these principles by learning how they are applied to a variety of Python's built-in functions. Chapters 9 through 11 cover design patterns, and the final two chapters discuss two bonus topics related to Python programming that may be of interest.

Chapter 1, Object-oriented Design, covers important object-oriented concepts. It deals mainly with terminology such as abstraction, classes, encapsulation, and inheritance. We also briefly look at UML to model our classes and objects.

Chapter 2, Objects in Python, discusses classes and objects and how they are used in Python. We will learn about attributes and behaviors on Python objects, and also the organization of classes into packages and modules. Lastly, we will see how to protect our data.

Chapter 3, When Objects Are Alike, gives us a more in-depth look into inheritance. It covers multiple inheritance and shows us how to extend built-ins. This chapter also covers how polymorphism and duck typing work in Python.

Chapter 4, Expecting the Unexpected, looks into exceptions and exception handling. We will learn how to create our own exceptions and how to use exceptions for program flow control.

Chapter 5, When to Use Object-oriented Programming, deals with creating and using objects. We will see how to wrap data using properties and restrict data access. This chapter also discusses the DRY principle and how not to repeat code.

Chapter 6, Python Data Structures, covers the object-oriented features of Python's built-in classes. We'll cover tuples, dictionaries, lists, and sets, as well as a few more advanced collections. We'll also see how to extend these standard objects.

Chapter 7, Python Object-oriented Shortcuts, as the name suggests, deals with time-savers in Python. We will look at many useful built-in functions such as method overloading using default arguments. We'll also see that functions themselves are objects and how this is useful.

Chapter 8, Strings and Serialization, looks at strings, files, and formatting. We'll discuss the difference between strings, bytes, and bytearrays, as well as various ways to serialize textual, object, and binary data to several canonical representations.

Chapter 9, The Iterator Pattern, introduces us to the concept of design patterns and covers Python's iconic implementation of the iterator pattern. We'll learn about list, set, and dictionary comprehensions. We'll also demystify generators and coroutines.

Chapter 10, Python Design Patterns I, covers several design patterns, including the decorator, observer, strategy, state, singleton, and template patterns. Each pattern is discussed with suitable examples and programs implemented in Python.

Chapter 11, Python Design Patterns II, wraps up our discussion of design patterns with coverage of the adapter, facade, flyweight, command, abstract, and composite patterns. More examples of how idiomatic Python code differs from canonical implementations are provided.

Chapter 12, Testing Object-oriented Programs, opens with why testing is so important in Python applications. It emphasizes test-driven development and introduces two different testing suites: unittest and py.test. Finally, it discusses mocking test objects and code coverage.

Chapter 13, Concurrency, is a whirlwind tour of Python's support (and lack thereof) of concurrency patterns. It discusses threads, multiprocessing, futures, and the new AsyncIO library.

Each chapter includes relevant examples and a case study that collects the chapter's contents into a working (if not complete) program.

What you need for this book

All the examples in this book rely on the Python 3 interpreter. Make sure you are not using Python 2.7 or earlier. At the time of writing, Python 3.4 was the latest release of Python. Most examples will work on earlier revisions of Python 3, but you are encouraged to use the latest version to minimize frustration.

All of the examples should run on any operating system supported by Python. If this is not the case, please report it as a bug.

Some of the examples need a working Internet connection. You'll probably want to have one of these for extracurricular research and debugging anyway!

In addition, some of the examples in this book rely on third-party libraries that do not ship with Python. These are introduced within the book at the time they are used, so you do not need to install them in advance. However, for completeness, here is a list:

- pip
- requests
- pillow
- bitarray

Who this book is for

This book specifically targets people who are new to object-oriented programming. It assumes you have basic Python skills. You'll learn object-oriented principles in depth. It is particularly useful for system administrator types who have used Python as a "glue" language and would like to improve their programming skills.

If you are familiar with object-oriented programming in other languages, then this book will help you understand the idiomatic ways to apply your knowledge in the Python ecosystem.

Conventions

This book uses a variety of text styles to distinguish between different kinds of information. Here are some examples of these styles, and an explanation of their meaning.

Code words in text, database table names, folder names, filenames, file extensions, pathnames, dummy URLs, user input, and Twitter handles are shown as follows: "We look up the class in the dictionary and store it in a variable named `PropertyClass`."

A block of code is set as follows:

```
def add_property(self):
    property_type = get_valid_input(
            "What type of property? ",
            ("house", "apartment")).lower()
    payment_type = get_valid_input(
            "What payment type? ",
            ("purchase", "rental")).lower()
```

When we wish to draw your attention to a particular part of a code block, the relevant lines or items are set in bold:

```
def add_property(self):
    property_type = get_valid_input(
            "What type of property? ",
            ("house", "apartment")).lower()
    payment_type = get_valid_input(
            "What payment type? ",
            ("purchase", "rental")).lower()
```

Any command-line input or output is written as follows:

```
>>> c1 = Contact("John A", "johna@example.net")
>>> c2 = Contact("John B", "johnb@example.net")
>>> c3 = Contact("Jenna C", "jennac@example.net")
>>> [c.name for c in Contact.all_contacts.search('John')]
['John A', 'John B']
```

New terms and **important words** are shown in bold. Words that you see on the screen, in menus or dialog boxes for example, appear in the text like this: "It will fail with a **not enough arguments** error similar to the one we received earlier when we forgot the `self` argument."

> Warnings or important notes appear in a box like this.

> Tips and tricks appear like this.

Reader feedback

Feedback from our readers is always welcome. Let us know what you think about this book—what you liked or disliked. Reader feedback is important for us as it helps us develop titles that you will really get the most out of.

To send us general feedback, simply e-mail `feedback@packtpub.com`, and mention the book's title in the subject of your message.

If there is a topic that you have expertise in and you are interested in either writing or contributing to a book, see our author guide at `www.packtpub.com/authors`.

Customer support

Now that you are the proud owner of a Packt book, we have a number of things to help you to get the most from your purchase.

Downloading the example code

You can download the example code files from your account at http://www.packtpub.com for all the Packt Publishing books you have purchased. If you purchased this book elsewhere, you can visit http://www.packtpub.com/support and register to have the files e-mailed directly to you.

Errata

Although we have taken every care to ensure the accuracy of our content, mistakes do happen. If you find a mistake in one of our books—maybe a mistake in the text or the code—we would be grateful if you could report this to us. By doing so, you can save other readers from frustration and help us improve subsequent versions of this book. If you find any errata, please report them by visiting http://www.packtpub.com/submit-errata, selecting your book, clicking on the **Errata Submission Form** link, and entering the details of your errata. Once your errata are verified, your submission will be accepted and the errata will be uploaded to our website or added to any list of existing errata under the Errata section of that title.

To view the previously submitted errata, go to https://www.packtpub.com/books/content/support and enter the name of the book in the search field. The required information will appear under the **Errata** section.

Piracy

Piracy of copyrighted material on the Internet is an ongoing problem across all media. At Packt, we take the protection of our copyright and licenses very seriously. If you come across any illegal copies of our works in any form on the Internet, please provide us with the location address or website name immediately so that we can pursue a remedy.

Please contact us at copyright@packtpub.com with a link to the suspected pirated material.

We appreciate your help in protecting our authors and our ability to bring you valuable content.

Questions

If you have a problem with any aspect of this book, you can contact us at questions@packtpub.com, and we will do our best to address the problem.