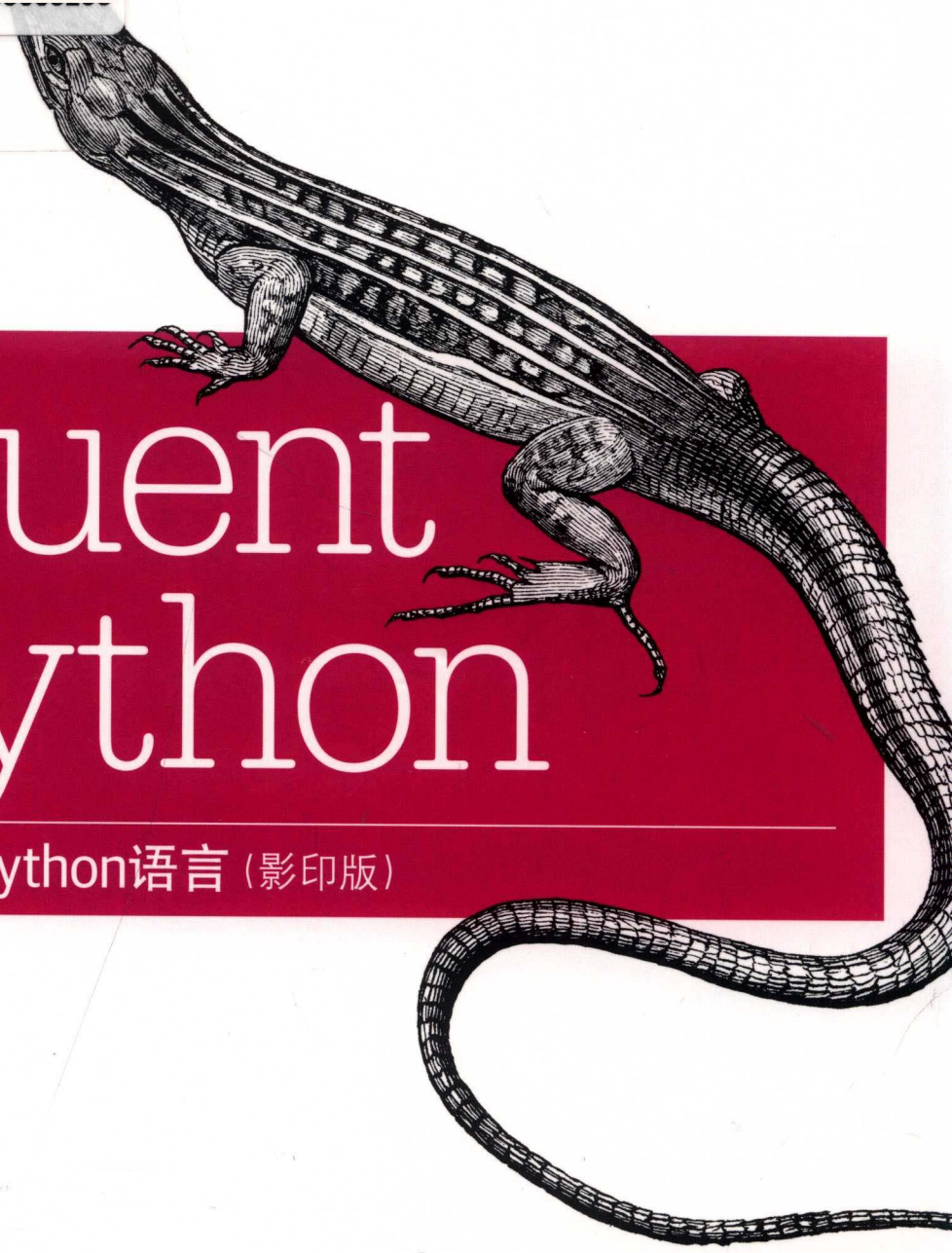
O'REILLY®



Fluent Python

流利的Python语言（影印版）

東南大學出版社

Luciano Ramalho 著

O'REILLY®

流利的Python语言（影印版）

Python的简洁让你可以快速达到高产水平，但是这通常意味着你并没有用到它提供的所有功能特性。有了这本容易上手的指南，你将学会如何利用那些十分优秀但很容易被忽视的特性来编写有效且通顺的Python代码。作者Luciano Ramalho将带领你们遍历Python的核心语言特性和程序库，并展示精简代码、优化速度和可读性的方法。

很多富有经验的程序员尝试将Python扭曲成适合他们从其他语言中学到的模式，而从未发现超出他们经验的Python特性。通过这本书，那些Python程序员将完全学会如何高效率地使用Python 3。

本书涵盖：

- Python数据模型：理解特殊方法如何成为对象一致行为的关键
- 数据结构：利用内置类型提供的所有便利并理解Unicode时代的文本和字节的二元性
- 作为对象的函数：将Python函数作为第一类对象，并理解这将如何影响流行的设计模式
- 面向对象的习语：通过学习引用、易变性、接口、操作符重载和多重继承来构建类
- 控制流：利用concurrent.futures和asyncio包中的上下文管理、生成器、协程和并发
- 元编程：理解特性、属性描述符、类装饰器和元类如何工作

Luciano Ramalho，1998开始成为Python程序员，是Python软件基金会的会员，巴西培训公司Python.pro.br的共同所有人，巴西第一家黑客空间Garoa Hacker Clube的联合创始人。他领导过软件开发团队，并在巴西的媒体、银行和政府部门教授过Python课程。

“我[illegible]的技术[illegible]帮助很多Python入门程序员走上精通的道路，也教会了我很多东西。”

——Alex Martelli
Python软件基金会会员

“《流利的Python语言》是一座充满有用编程技巧的宝藏，为Python程序员提供了从入门到精通的知识拓展途径。”

——Daniel和Audrey Roy Greenfield
《Two Scoops of Django》作者

PROGRAMMING/PYTHON

责任编辑：张烨
封面设计：Ellie Volckhausen，张健

O'Reilly Media, Inc.授权东南大学出版社出版

ISBN 978-7-5641-6874-2
9 787564 168742 >
定价：118.00元

流利的Python语言（影印版）

Fluent Python

Luciano Ramalho 著

Beijing • Cambridge • Farnham • Köln • Sebastopol • Tokyo O'REILLY®

O'Reilly Media, Inc. 授权东南大学出版社出版

南京 东南大学出版社

图书在版编目(CIP)数据

流利的 Python 语言：英文/(巴西)卢西亚诺·拉马略(Luciano Ramalho)著. —影印本. —南京：东南大学出版社，2017.1（2019.1 重印）

书名原文：Fluent Python

ISBN 978-7-5641-6874-2

Ⅰ. ①流… Ⅱ. ①卢… 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(2016)第 294337 号

图字：10-2015-246 号

流利的 Python 语言(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：江苏凤凰数码印务有限公司
开　　本：787 毫米×980 毫米　　16 开本
印　　张：48.25
字　　数：945 千字
版　　次：2017 年 1 月第 1 版
印　　次：2019 年 1 月第 5 次印刷
书　　号：ISBN 978-7-5641-6874-2
定　　价：118.00 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

Para Marta, com todo o meu amor.

Preface

> Here's the plan: when someone uses a feature you don't understand, simply shoot them. This is easier than learning something new, and before too long the only living coders will be writing in an easily understood, tiny subset of Python 0.9.6 <wink>.[1]
>
> — Tim Peters
> *Legendary core developer and author of The Zen of Python*

"Python is an easy to learn, powerful programming language." Those are the first words of the official Python Tutorial (*https://docs.python.org/3/tutorial/*). That is true, but there is a catch: because the language is easy to learn and put to use, many practicing Python programmers leverage only a fraction of its powerful features.

An experienced programmer may start writing useful Python code in a matter of hours. As the first productive hours become weeks and months, a lot of developers go on writing Python code with a very strong accent carried from languages learned before. Even if Python is your first language, often in academia and in introductory books it is presented while carefully avoiding language-specific features.

As a teacher introducing Python to programmers experienced in other languages, I see another problem that this book tries to address: we only miss stuff we know about. Coming from another language, anyone may guess that Python supports regular expressions, and look that up in the docs. But if you've never seen tuple unpacking or descriptors before, you will probably not search for them, and may end up not using those features just because they are specific to Python.

This book is not an A-to-Z exhaustive reference of Python. Its emphasis is on the language features that are either unique to Python or not found in many other popular languages. This is also mostly a book about the core language and some of its libraries. I will rarely talk about packages that are not in the standard library, even though the

1. Message to the comp.lang.python Usenet group, Dec. 23, 2002: "Acrimony in c.l.p." (*https://mail.python.org/pipermail/python-list/2002-December/147293.html*)

Python package index now lists more than 60,000 libraries and many of them are incredibly useful.

Who This Book Is For

This book was written for practicing Python programmers who want to become proficient in Python 3. If you know Python 2 but are willing to migrate to Python 3.4 or later, you should be fine. At the time of this writing, the majority of professional Python programmers are using Python 2, so I took special care to highlight Python 3 features that may be new to that audience.

However, *Fluent Python* is about making the most of Python 3.4, and I do not spell out the fixes needed to make the code work in earlier versions. Most examples should run in Python 2.7 with little or no changes, but in some cases, backporting would require significant rewriting.

Having said that, I believe this book may be useful even if you must stick with Python 2.7, because the core concepts are still the same. Python 3 is not a new language, and most differences can be learned in an afternoon. What's New in Python 3.0 (*https://docs.python.org/3.0/whatsnew/3.0.html*) is a good starting point. Of course, there have been changes since Python 3.0 was released in 2009, but none as important as those in 3.0.

If you are not sure whether you know enough Python to follow along, review the topics of the official Python Tutorial (*https://docs.python.org/3/tutorial/*). Topics covered in the tutorial will not be explained here, except for some features that are new in Python 3.

Who This Book Is Not For

If you are just learning Python, this book is going to be hard to follow. Not only that, if you read it too early in your Python journey, it may give you the impression that every Python script should leverage special methods and metaprogramming tricks. Premature abstraction is as bad as premature optimization.

How This Book Is Organized

The core audience for this book should not have trouble jumping directly to any chapter in this book. However, each of the six parts forms a book within the book. I conceived the chapters within each part to be read in sequence.

I tried to emphasize using what is available before discussing how to build your own. For example, in Part II, Chapter 2 covers sequence types that are ready to use, including some that don't get a lot of attention, like `collections.deque`. Building user-defined

sequences is only addressed in Part IV, where we also see how to leverage the abstract base classes (ABCs) from `collections.abc`. Creating your own ABCs is discussed even later in Part IV, because I believe it's important to be comfortable using an ABC before writing your own.

This approach has a few advantages. First, knowing what is ready to use can save you from reinventing the wheel. We use existing collection classes more often than we implement our own, and we can give more attention to the advanced usage of available tools by deferring the discussion on how to create new ones. We are also more likely to inherit from existing ABCs than to create a new ABC from scratch. And finally, I believe it is easier to understand the abstractions after you've seen them in action.

The downside of this strategy are the forward references scattered throughout the chapters. I hope these will be easier to tolerate now that you know why I chose this path.

Here are the main topics in each part of the book:

Part I

A single chapter about the Python data model explaining how the special methods (e.g., `__repr__`) are the key to the consistent behavior of objects of all types—in a language that is admired for its consistency. Understanding various facets of the data model is the subject of most of the rest of the book, but Chapter 1 provides a high-level overview.

Part II

The chapters in this part cover the use of collection types: sequences, mappings, and sets, as well as the `str` versus `bytes` split—the cause of much celebration among Python 3 users and much pain for Python 2 users who have not yet migrated their code bases. The main goals are to recall what is already available and to explain some behavior that is sometimes surprising, like the reordering of `dict` keys when we are not looking, or the caveats of locale-dependent Unicode string sorting. To achieve these goals, the coverage is sometimes high level and wide (e.g., when many variations of sequences and mappings are presented) and sometimes deep (e.g., when we dive into the hash tables underneath the `dict` and `set` types).

Part III

Here we talk about functions as first-class objects in the language: what that means, how it affects some popular design patterns, and how to implement function decorators by leveraging closures. Also covered here is the general concept of callables in Python, function attributes, introspection, parameter annotations, and the new `nonlocal` declaration in Python 3.

Part IV

Now the focus is on building classes. In Part II, the `class` declaration appears in few examples; Part IV presents many classes. Like any object-oriented (OO) language, Python has its particular set of features that may or may not be present in

the language in which you and I learned class-based programming. The chapters explain how references work, what mutability really means, the lifecycle of instances, how to build your own collections and ABCs, how to cope with multiple inheritance, and how to implement operator overloading—when that makes sense.

Part V

Covered in this part are the language constructs and libraries that go beyond sequential control flow with conditionals, loops, and subroutines. We start with generators, then visit context managers and coroutines, including the challenging but powerful new `yield from` syntax. Part V closes with a high-level introduction to modern concurrency in Python with `collections.futures` (using threads and processes under the covers with the help of futures) and doing event-oriented I/O with `asyncio` (leveraging futures on top of coroutines and `yield from`).

Part VI

This part starts with a review of techniques for building classes with attributes created dynamically to handle semi-structured data such as JSON datasets. Next, we cover the familiar properties mechanism, before diving into how object attribute access works at a lower level in Python using descriptors. The relationship between functions, methods, and descriptors is explained. Throughout Part VI, the step-by-step implementation of a field validation library uncovers subtle issues that lead to the use of the advanced tools of the final chapter: class decorators and metaclasses.

Hands-On Approach

Often we'll use the interactive Python console to explore the language and libraries. I feel it is important to emphasize the power of this learning tool, particularly for those readers who've had more experience with static, compiled languages that don't provide a read-eval-print#loop (REPL).

One of the standard Python testing packages, `doctest` (*https://docs.python.org/3/library/doctest.html*), works by simulating console sessions and verifying that the expressions evaluate to the responses shown. I used `doctest` to check most of the code in this book, including the console listings. You don't need to use or even know about `doctest` to follow along: the key feature of doctests is that they look like transcripts of interactive Python console sessions, so you can easily try out the demonstrations yourself.

Sometimes I will explain what we want to accomplish by showing a doctest before the code that makes it pass. Firmly establishing what is to be done before thinking about how to do it helps focus our coding effort. Writing tests first is the basis of test driven development (TDD) and I've also found it helpful when teaching. If you are unfamiliar with `doctest`, take a look at its documentation (*https://docs.python.org/3/library/doctest.html*) and this book's source code repository (*https://github.com/fluentpython/*

example-code). You'll find that you can verify the correctness of most of the code in the book by typing `python3 -m doctest example_script.py` in the command shell of your OS.

Hardware Used for Timings

The book has some simple benchmarks and timings. Those tests were performed on one or the other laptop I used to write the book: a 2011 MacBook Pro 13" with a 2.7 GHz Intel Core i7 CPU, 8GB of RAM, and a spinning hard disk, and a 2014 MacBook Air 13" with a 1.4 GHz Intel Core i5 CPU, 4GB of RAM, and a solid-state disk. The MacBook Air has a slower CPU and less RAM, but its RAM is faster (1600 versus 1333 MHz) and the SSD is much faster than the HD. In daily usage, I can't tell which machine is faster.

Soapbox: My Personal Perspective

I have been using, teaching, and debating Python since 1998, and I enjoy studying and comparing programming languages, their design, and the theory behind them. At the end of some chapters, I have added "Soapbox" sidebars with my own perspective about Python and other languages. Feel free to skip these if you are not into such discussions. Their content is completely optional.

Python Jargon

I wanted this to be a book not only about Python but also about the culture around it. Over more than 20 years of communications, the Python community has developed its own particular lingo and acronyms. At the end of this book, Python Jargon contains a list of terms that have special meaning among Pythonistas.

Python Version Covered

I tested all the code in the book using Python 3.4—that is, CPython 3.4, the most popular Python implementation written in C. There is only one exception: "The New @ Infix Operator in Python 3.5" on page 387 shows the `@` operator, which is only supported by Python 3.5.

Almost all code in the book should work with any Python 3.x–compatible interpreter, including PyPy3 2.4.0, which is compatible with Python 3.2.5. The notable exceptions are the examples using `yield from` and `asyncio`, which are only available in Python 3.3 or later.

Most code should also work with Python 2.7 with minor changes, except the Unicode-related examples in Chapter 4, and the exceptions already noted for Python 3 versions earlier than 3.3.

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 variable or function names, databases, data types, environment variables, statements, and keywords.

Note that when a line break falls within a `constant_width` term, a hyphen is not added —it could be misunderstood as part of the term.

`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.

This element signifies a tip or suggestion.

This element signifies a general note.

This element indicates a warning or caution.

Using Code Examples

Every script and most code snippets that appear in the book are available in the Fluent Python code repository (*https://github.com/fluentpython/example-code*) on GitHub.

We appreciate, but do not require, attribution. An attribution usually includes the title, author, publisher, and ISBN. For example: "*Fluent Python* by Luciano Ramalho (O'Reilly). Copyright 2015 Luciano Ramalho, 978-1-491-94600-8."

Safari® Books Online

Safari Books Online is an on-demand digital library that delivers expert content in both book and video form from the world's leading authors in technology and business.

Technology professionals, software developers, web designers, and business and creative professionals use Safari Books Online as their primary resource for research, problem solving, learning, and certification training.

Safari Books Online offers a range of product mixes and pricing programs for organizations, government agencies, and individuals. Subscribers have access to thousands of books, training videos, and prepublication manuscripts in one fully searchable database from publishers like O'Reilly Media, Prentice Hall Professional, Addison-Wesley Professional, Microsoft Press, Sams, Que, Peachpit Press, Focal Press, Cisco Press, John Wiley & Sons, Syngress, Morgan Kaufmann, IBM Redbooks, Packt, Adobe Press, FT Press, Apress, Manning, New Riders, McGraw-Hill, Jones & Bartlett, Course Technology, and dozens more. For more information about Safari Books Online, please visit us online.

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 *http://bit.ly/fluent-python*.

To comment or ask technical questions about this book, send email to *bookques tions@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

The Bauhaus chess set by Josef Hartwig is an example of excellent design: beautiful, simple, and clear. Guido van Rossum, son of an architect and brother of a master font designer, created a masterpiece of language design. I love teaching Python because it is beautiful, simple, and clear.

Alex Martelli and Anna Ravenscroft were the first people to see the outline of this book and encouraged me to submit it to O'Reilly for publication. Their books taught me idiomatic Python and are models of clarity, accuracy, and depth in technical writing. Alex's 5,000+ Stack Overflow posts (*http://stackoverflow.com/users/95810/alex-martelli*) are a fountain of insights about the language and its proper use.

Martelli and Ravenscroft were also technical reviewers of this book, along with Lennart Regebro and Leonardo Rochael. Everyone in this outstanding technical review team has at least 15 years of Python experience, with many contributions to high-impact Python projects in close contact with other developers in the community. Together they sent me hundreds of corrections, suggestions, questions, and opinions, adding tremendous value to the book. Victor Stinner kindly reviewed Chapter 18, bringing his expertise as an `asyncio` maintainer to the technical review team. It was a great privilege and a pleasure to collaborate with them over these past several months.

Editor Meghan Blanchette was an outstanding mentor, helping me improve the organization and flow of the book, letting me know when it was boring, and keeping me from delaying even more. Brian MacDonald edited chapters in Part III while Meghan was away. I enjoyed working with them, and with everyone I've contacted at O'Reilly, including the Atlas development and support team (Atlas is the O'Reilly book publishing platform, which I was fortunate to use to write this book).

Mario Domenech Goulart provided numerous, detailed suggestions starting with the first Early Release. I also received valuable feedback from Dave Pawson, Elias Dorneles, Leonardo Alexandre Ferreira Leite, Bruce Eckel, J. S. Bueno, Rafael Gonçalves, Alex Chiaranda, Guto Maia, Lucas Vido, and Lucas Brunialti.

Over the years, a number of people urged me to become an author, but the most persuasive were Rubens Prates, Aurelio Jargas, Rudá Moura, and Rubens Altimari. Mauricio Bussab opened many doors for me, including my first real shot at writing a book. Renzo Nuccitelli supported this writing project all the way, even if that meant a slow start for our partnership at *python.pro.br*.

The wonderful Brazilian Python community is knowledgeable, generous, and fun. The Python Brasil group (*https://groups.google.com/group/python-brasil*) has thousands of people and our national conferences bring together hundreds, but the most influential in my journey as a Pythonista were Leonardo Rochael, Adriano Petrich, Daniel Vainsencher, Rodrigo RBP Pimentel, Bruno Gola, Leonardo Santagada, Jean Ferri, Rodrigo Senra, J. S. Bueno, David Kwast, Luiz Irber, Osvaldo Santana, Fernando Masanori, Henrique Bastos, Gustavo Niemayer, Pedro Werneck, Gustavo Barbieri, Lalo Martins, Danilo Bellini, and Pedro Kroger.

Dorneles Tremea was a great friend (incredibly generous with his time and knowledge), an amazing hacker, and the most inspiring leader of the Brazilian Python Association. He left us too early.

My students over the years taught me a lot through their questions, insights, feedback, and creative solutions to problems. Érico Andrei and Simples Consultoria made it possible for me to focus on being a Python teacher for the first time.

Martijn Faassen was my Grok mentor and shared invaluable insights with me about Python and Neanderthals. His work and that of Paul Everitt, Chris McDonough, Tres Seaver, Jim Fulton, Shane Hathaway, Lennart Regebro, Alan Runyan, Alexander Limi, Martijn Pieters, Godefroid Chapelle, and others from the Zope, Plone, and Pyramid planets have been decisive in my career. Thanks to Zope and surfing the first web wave, I was able to start making a living with Python in 1998. José Octavio Castro Neves was my partner in the first Python-centric software house in Brazil.

I have too many gurus in the wider Python community to list them all, but besides those already mentioned, I am indebted to Steve Holden, Raymond Hettinger, A.M. Kuchling, David Beazley, Fredrik Lundh, Doug Hellmann, Nick Coghlan, Mark Pilgrim, Martijn Pieters, Bruce Eckel, Michele Simionato, Wesley Chun, Brandon Craig Rhodes, Philip Guo, Daniel Greenfeld, Audrey Roy, and Brett Slatkin for teaching me new and better ways to teach Python.

Most of these pages were written in my home office and in two labs: CoffeeLab and Garoa Hacker Clube. CoffeeLab (*http://coffeelab.com.br/*) is the caffeine-geek headquarters in Vila Madalena, São Paulo, Brazil. Garoa Hacker Clube (*https://garoa.net.br/*) is a hackerspace open to all: a community lab where anyone can freely try out new ideas.

The Garoa community provided inspiration, infrastructure, and slack. I think Aleph would enjoy this book.

My mother, Maria Lucia, and my father, Jairo, always supported me in every way. I wish he was here to see the book; I am glad I can share it with her.

My wife, Marta Mello, endured 15 months of a husband who was always working, but remained supportive and coached me through some critical moments in the project when I feared I might drop out of the marathon.

Thank you all, for everything.

Table of Contents