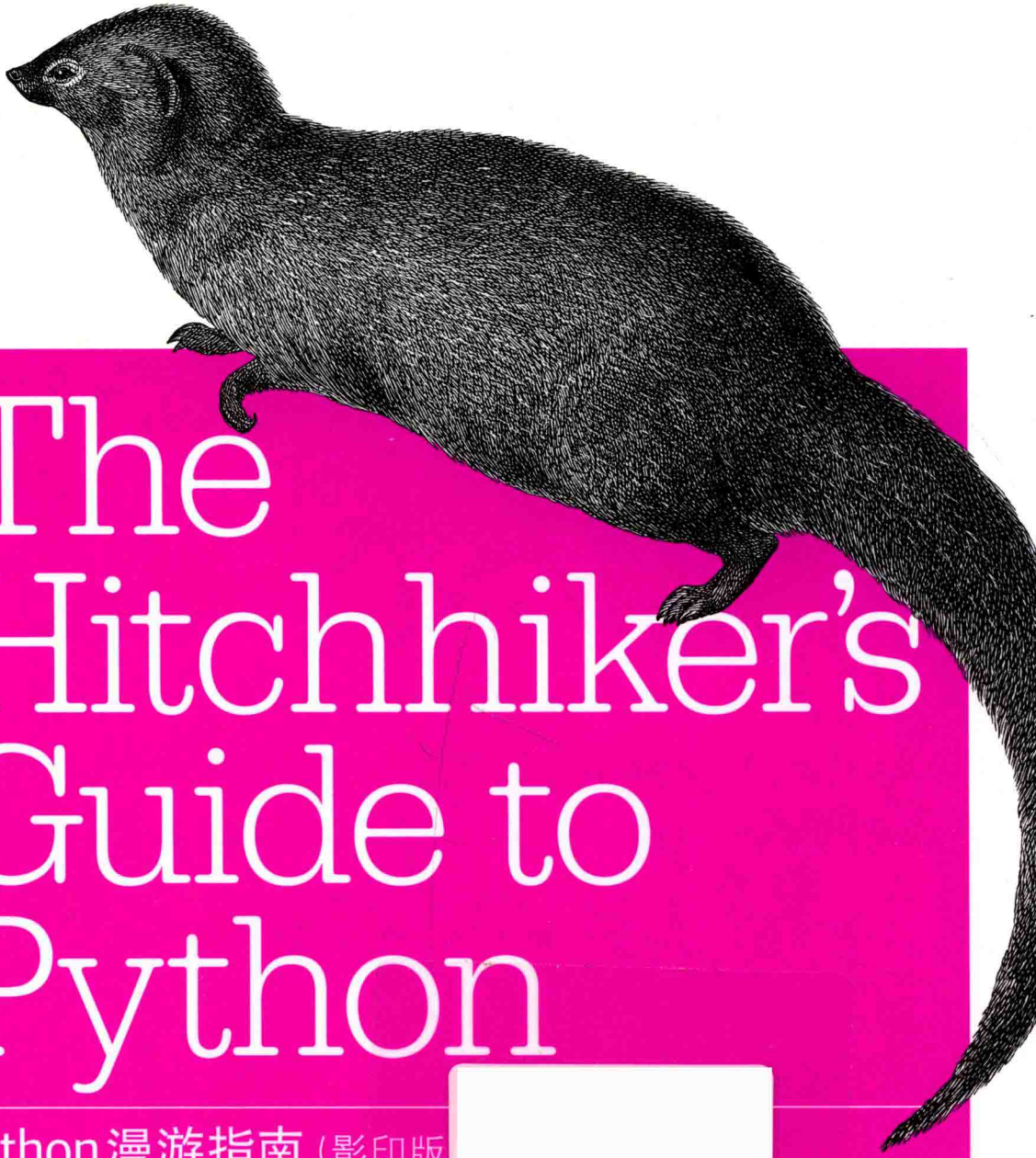
O'REILLY®



The Hitchhiker's Guide to Python

Python漫游指南（影印版）

東南大學出版社

Kenneth Reitz, Tanya Schlusser 著

Python漫游指南（影印版）

The Hitchhiker's Guide to Python

Kenneth Reitz, Tanya Schlusser 著

Beijing • Boston • Farnham • 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

Python 漫游指南:英文/(美)肯尼思·赖茨(Kenneth Reitz),(美)坦尼娅·胥卢瑟(Tanya Schlusser)著. —影印本. —南京:东南大学出版社,2017.10

书名原文:The Hitchhiker's Guide to Python

ISBN 978-7-5641-7374-6

Ⅰ.①P… Ⅱ.①肯… ②坦… Ⅲ.①软件工具—程序设计—指南—英文 Ⅳ.①TP311.561-62

中国版本图书馆 CIP 数据核字(2017)第 195475 号

图字:10-2017-348 号

Python 漫游指南(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:21.25
字　　数:416 千字
版　　次:2017 年 10 月第 1 版
印　　次:2017 年 10 月第 1 次印刷
书　　号:ISBN 978-7-5641-7374-6
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

Preface

Python is big. Really big. You just won't believe how vastly hugely mind-bogglingly big it is.

This guide is *not* intended to teach you the Python language (we cite lots of great resources that do that) but is rather an (opinionated) insider's guide to our community's favorite tools and best practices. The primary audience is new to mid-level Python programmers who are interested in contributing to open source or in beginning a career or starting a company using Python, although casual Python users should also find Part I and Chapter 5 helpful.

The first part will help you choose the text editor or interactive development environment that fits your situation (for example, those using Java frequently may prefer Eclipse with a Python plug-in) and surveys options for other interpreters that may meet needs you don't yet know Python could address (e.g., there's a MicroPython implementation based around the ARM Cortex-M4 chip). The second section demonstrates Pythonic style by highlighting exemplary code in the open source community that will hopefully encourage more in-depth reading and experimentation with open source code. The final section briefly surveys the vast galaxy of libraries most commonly used in the Python community—providing an idea of the scope of what Python can do right now.

All of the royalties from the print version of this book will be directly donated to the Django Girls (*https://djangogirls.org/*), a giddily joyous global organization dedicated to organizing free Django and Python workshops, creating open-sourced online tutorials, and curating amazing first experiences with technology. Those who wish to contribute to the online version can read more about how to do it at our website (*http://docs.python-guide.org/en/latest/notes/contribute/*).

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.

`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.

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 plans and pricing for enterprise, government, education, and individuals.

Members 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 hundreds 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/the-hitchhikers-guide-to-python*.

To comment or ask technical questions about this book, send 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

Welcome, friends, to *The Hitchhiker's Guide to Python*.

This book is, to the best of my knowledge, the first of its kind: designed and curated by a single author (myself—Kenneth), with the majority of the content provided by hundreds of people from all over the world, for free. Never before in the history of mankind has the technology been available to allow a beautiful collaboration of this size and scale.

This book was made possible with:

Community
: Love brings us together to conquer all obstacles.

Software projects
: Python, Sphinx, Alabaster, and Git.

Services
: GitHub and Read the Docs.

Lastly, I'd like to extend a personal thank you to Tanya, who did all the hard work of converting this work into book form and preparing it for publication, and the incredible O'Reilly team—Dawn, Jasmine, Nick, Heather, Nicole, Meg, and the dozens of other people who worked behind the scenes to make this book the best it could be.

Table of Contents

PART I

Getting Started

This part of the guide focuses on setting up a Python environment. It was inspired by Stuart Ellis's guide for Python on Windows (*http://www.stuartellis.eu/articles/python-development-windows/*), and consists of the following chapters and topics:

Chapter 1, Picking an Interpreter
: We compare Python 2 and Python 3, and share some interpreter options other than CPython.

Chapter 2, Properly Installing Python
: We show how to get Python, pip, and virtualenv.

Chapter 3, Your Development Environment
: We describe our favorite text editors and IDEs for Python development.

CHAPTER 1

Picking an Interpreter

The State of Python 2 Versus Python 3

When choosing a Python interpreter, one looming question is always present: "Should I choose Python 2 or Python 3?" The answer is not as obvious as one might think (although 3 is becoming more compelling every day).

Here is the state of things:

- Python 2.7 has been the standard for a *long* time.
- Python 3 introduced major changes to the language, which some developers are unhappy with.[1]
- Python 2.7 will receive necessary security updates until 2020 (*https://www.python.org/dev/peps/pep-0373/*).
- Python 3 is continually evolving, like Python 2 did in years past.

You can now see why this is not such an easy decision.

1 If you don't do much low-level networking programming, the change was barely noticeable outside of the `print` statement becoming a function. Otherwise, "unhappy with" is kind of a polite understatement—developers responsible for large, popular web, socket, or networking libraries that deal with unicode and byte strings had (or still have) *extensive* changes to make. Details about the change, direct from the first introduction of Python 3 to the world, start off with: "Everything you thought you knew about binary data and Unicode has changed." (*http://bit.ly/text-vs-data*)

Recommendations

The way we see it, a truly hoopy frood[2] would use Python 3. But if you can only use Python 2, at least you're still using Python. These are our recommendations:

Use Python 3 if...

- You love Python 3.
- You don't know which one to use.
- You embrace change.

Use Python 2 if...

- You love Python 2 and are saddened by the future being Python 3.
- The stability requirements of your software would be impacted.[3]
- Software that you depend on requires it.

So...3?

If you're choosing a Python interpreter to use, and aren't opinionated, then use the newest Python 3.x—every version brings new and improved standard library modules, security, and bug fixes. Progress is progress. So only use Python 2 if you have a strong reason to, such as a Python 2–exclusive library that has no adequate Python 3–ready alternative, a need for a specific implementation (see "Implementations" on page 5), or you (like some of us) love and are inspired by Python 2.

Check out Can I Use Python 3? (*https://caniusepython3.com/*) to see whether any Python projects you're depending on will block adoption of Python 3.

For further reading, try Python2orPython3 (*http://bit.ly/python2-or-python3*), which lays out some of the reasoning behind a backward-incompatible break in the language specification, and links to detailed specifications of the differences.

If you're a beginner, there are far more important things to worry about than cross-compatibility between all of the Python versions. Just get something working for the system you've got, and cross this bridge later.

2 Someone who's really amazingly together. We mean, who really knows where their towel is.

3 Here's a link to a high-level list of changes (*http://python3porting.com/stdlib.html*) to Python's Standard Library.

Implementations

When people speak of *Python*, they often mean not just the language but also the CPython implementation. *Python* is actually a specification for a language that can be implemented in many different ways.

The different implementations may be for compatibility with other libraries, or maybe for a little speed. Pure Python libraries should work regardless of your Python implementation, but those built on C (like NumPy) won't. This section provides a quick rundown on the most popular implementations.

This guide presumes you're working with the standard CPython implementation of Python 3, although we'll frequently add notes when relevant for Python 2.

CPython

CPython (*http://www.python.org/*) is the reference implementation[4] of Python, written in C. It compiles Python code to intermediate bytecode which is then interpreted by a virtual machine. CPython provides the highest level of compatibility with Python packages and C extension modules.[5]

If you are writing open source Python code and want to reach the widest possible audience, use CPython. To use packages that rely on C extensions to function, CPython is your only implementation option.

All versions of the Python language are implemented in C because CPython is the reference implementation.

Stackless

Stackless Python (*https://bitbucket.org/stackless-dev/stackless/wiki/Home*) is regular CPython (so it should work with all of the libraries that CPython can use), but with a patch that decouples the Python interpreter from the call stack, making it possible to change the order of execution of code. Stackless introduces the contepts of *tasklets*, which can wrap functions and turn them into "micro-threads" that can be serialized to disk for future execution and scheduled, by default in round-robin execution.

4 The *reference implementation* accurately reflects the language's definition. Its behavior is how all other implementations should behave.

5 *C extension modules* are written in C for use in Python.