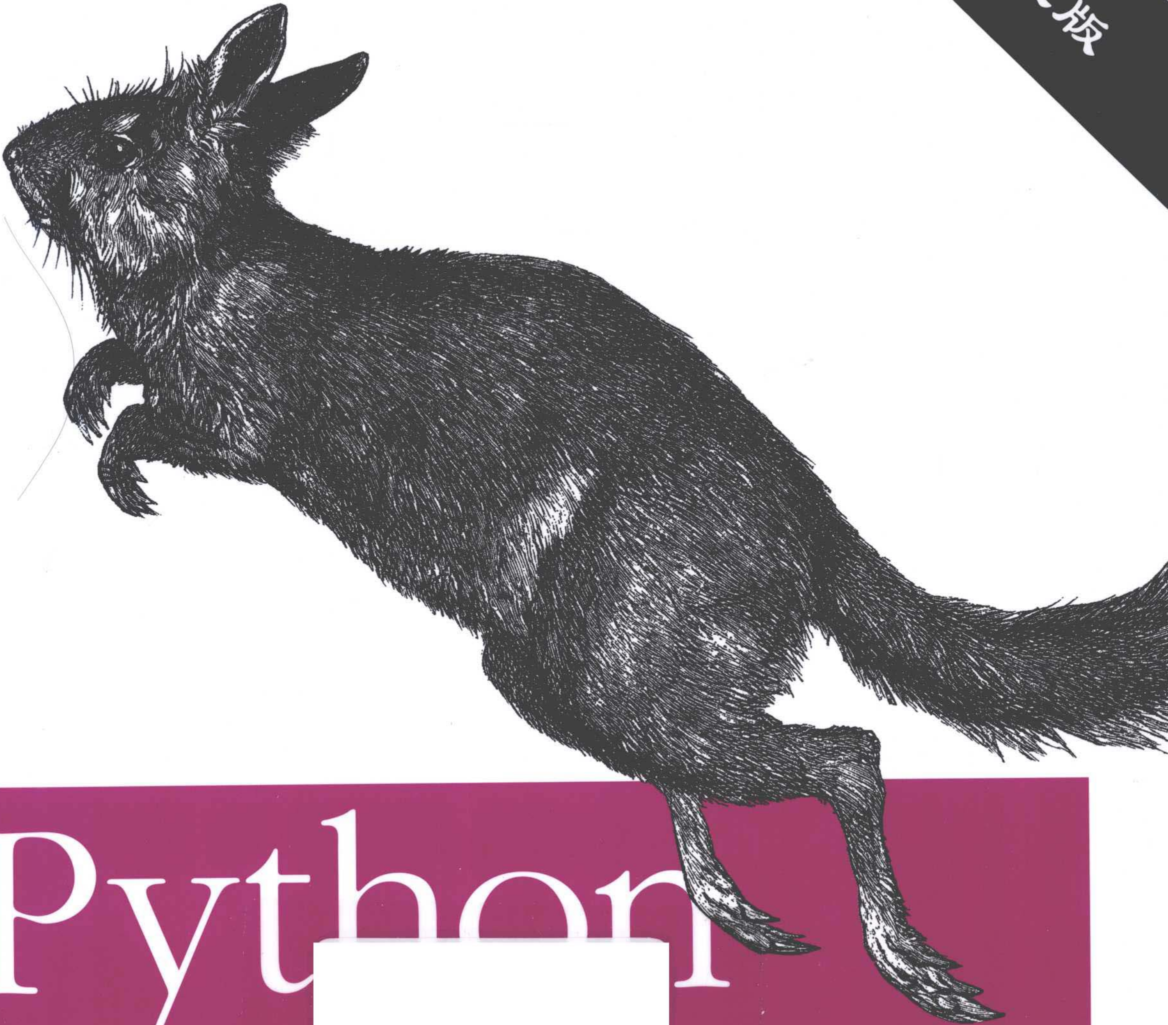
Python Cookbook（影印版）

第三版



Python Cookbook™

O'REILLY®

東南大學出版社

David Beazley & Brian K. Jones 著

第三版

Python Cookbook（影印版）

Python Cookbook

David Beazley, Brian K. Jones 著

O'REILLY®

Beijing · Cambridge · Farnham · Köln · Sebastopol · Tokyo

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 Cookbook: 第 3 版:英文 /(美)比兹利（Beazley, D.), (美)琼斯（Jones, B.K.)著 . —影印本 . —南京：东南大学出版社，2014.1

书名原文：Python Cookbook, 3E

ISBN 978-7-5641-4598-9

I. ① P… II. ①比… ②琼… III. ①软件工具－程序设计－英文 IV. ① TP311.56

中国版本图书馆 CIP 数据核字（2013）第 246079 号

江苏省版权局著作权合同登记

图字：10-2012-2 号

©2013 by O'Reilly Media, Inc.

Reprint of the English Edition, jointly published by O'Reilly Media, Inc. and Southeast University Press, 2014. Authorized reprint of the original English edition, 2012 O'Reilly Media, Inc., the owner of all rights to publish and sell the same.

All rights reserved including the rights of reproduction in whole or in part in any form.

英文原版由 O'Reilly Media, Inc. 出版 2013。

英文影印版由东南大学出版社出版 2014。此影印版的出版和销售得到出版权和销售权的所有者 —— O'Reilly Media, Inc. 的许可。

版权所有、未得书面许可，本书的任何部分和全部不得以任何形式重制。

安徽大学图书馆
藏书
www.lib.ahu.edu.cn

Python Cookbook 第三版（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：	扬中市印刷有限公司
开　　本：	787 毫米 × 980 毫米　16 开本
印　　张：	44.25
字　　数：	867 千字
版　　次：	2014 年 1 月第 1 版
印　　次：	2014 年 1 月第 1 次印刷
书　　号：	ISBN 978-7-5641-4598-9
定　　价：	98.00 元（册）

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

Since 2008, the Python world has been watching the slow evolution of Python 3. It was always known that the adoption of Python 3 would likely take a long time. In fact, even at the time of this writing (2013), most working Python programmers continue to use Python 2 in production. A lot has been made about the fact that Python 3 is not backward compatible with past versions. To be sure, backward compatibility is an issue for anyone with an existing code base. However, if you shift your view toward the future, you'll find that Python 3 offers much more than meets the eye.

Just as Python 3 is about the future, this edition of the *Python Cookbook* represents a major change over past editions. First and foremost, this is meant to be a very forward looking book. All of the recipes have been written and tested with Python 3.3 without regard to past Python versions or the "old way" of doing things. In fact, many of the recipes will only work with Python 3.3 and above. Doing so may be a calculated risk, but the ultimate goal is to write a book of recipes based on the most modern tools and idioms possible. It is hoped that the recipes can serve as a guide for people writing new code in Python 3 or those who hope to modernize existing code.

Needless to say, writing a book of recipes in this style presents a certain editorial challenge. An online search for Python recipes returns literally thousands of useful recipes on sites such as ActiveState's Python recipes or Stack Overflow. However, most of these recipes are steeped in history and the past. Besides being written almost exclusively for Python 2, they often contain workarounds and hacks related to differences between old versions of Python (e.g., version 2.3 versus 2.4). Moreover, they often use outdated techniques that have simply become a built-in feature of Python 3.3. Finding recipes exclusively focused on Python 3 can be a bit more difficult.

Rather than attempting to seek out Python 3-specific recipes, the topics of this book are merely inspired by existing code and techniques. Using these ideas as a springboard, the writing is an original work that has been deliberately written with the most modern Python programming techniques possible. Thus, it can serve as a reference for anyone who wants to write their code in a modern style.

In choosing which recipes to include, there is a certain realization that it is simply impossible to write a book that covers every possible thing that someone might do with Python. Thus, a priority has been given to topics that focus on the core Python language as well as tasks that are common to a wide variety of application domains. In addition, many of the recipes aim to illustrate features that are new to Python 3 and more likely to be unknown to even experienced programmers using older versions. There is also a certain preference to recipes that illustrate a generally applicable programming technique (i.e., programming patterns) as opposed to those that narrowly try to address a very specific practical problem. Although certain third-party packages get coverage, a majority of the recipes focus on the core language and standard library.

Who This Book Is For

This book is aimed at more experienced Python programmers who are looking to deepen their understanding of the language and modern programming idioms. Much of the material focuses on some of the more advanced techniques used by libraries, frameworks, and applications. Throughout the book, the recipes generally assume that the reader already has the necessary background to understand the topic at hand (e.g., general knowledge of computer science, data structures, complexity, systems programming, concurrency, C programming, etc.). Moreover, the recipes are often just skeletons that aim to provide essential information for getting started, but which require the reader to do more research to fill in the details. As such, it is assumed that the reader knows how to use search engines and Python's excellent online documentation.

Many of the more advanced recipes will reward the reader's patience with a much greater insight into how Python actually works under the covers. You will learn new tricks and techniques that can be applied to your own code.

Who This Book Is Not For

This is not a book designed for beginners trying to learn Python for the first time. In fact, it already assumes that you know the basics that might be taught in a Python tutorial or more introductory book. This book is also not designed to serve as a quick reference manual (e.g., quickly looking up the functions in a specific module). Instead, the book aims to focus on specific programming topics, show possible solutions, and serve as a springboard for jumping into more advanced material you might find online or in a reference.

试读结束：需要全本请在线购买： www.ertongbook.com

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 icon signifies a tip, suggestion, or general note.

This icon indicates a warning or caution.

Online Code Examples

Almost all of the code examples in this book are available online at *http://github.com/dabeaz/python-cookbook*. The authors welcome bug fixes, improvements, and comments.

Using Code Examples

This book is here to help you get your job done. In general, if this book includes code examples, 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 a CD-ROM of 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: *Python Cookbook*, 3rd edition, by David Beazley and Brian K. Jones (O'Reilly). Copyright 2013 David Beazley and Brian Jones, 978-1-449-34037-7.

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*.

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://oreil.ly/python_cookbook_3e*.

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

We would like to acknowledge the technical reviewers, Jake Vanderplas, Robert Kern, and Andrea Crotti, for their very helpful comments, as well as the general Python community for their support and encouragement. We would also like to thank the editors of the prior edition, Alex Martelli, Anna Ravenscroft, and David Ascher. Although this edition is newly written, the previous edition provided an initial framework for selecting the topics and recipes of interest. Last, but not least, we would like to thank readers of the early release editions for their comments and suggestions for improvement.

David Beazley's Acknowledgments

Writing a book is no small task. As such, I would like to thank my wife Paula and my two boys for their patience and support during this project. Much of the material in this book was derived from content I developed teaching Python-related training classes over the last six years. Thus, I'd like to thank all of the students who have taken my courses and ultimately made this book possible. I'd also like to thank Ned Batchelder, Travis Oliphant, Peter Wang, Brian Van de Ven, Hugo Shi, Raymond Hettinger, Michael Foord, and Daniel Klein for traveling to the four corners of the world to teach these courses while I stayed home in Chicago to work on this project. Meghan Blanchette and Rachel Roumeliotis of O'Reilly were also instrumental in seeing this project through to completion despite the drama of several false starts and unforeseen delays. Last, but not least, I'd like to thank the Python community for their continued support and putting up with my flights of diabolical fancy.

David M. Beazley

http://www.dabeaz.com

https://twitter.com/dabeaz

Brian Jones' Acknowledgments

I would like to thank both my coauthor, David Beazley, as well as Meghan Blanchette and Rachel Roumeliotis of O'Reilly, for working with me on this project. I would also like to thank my amazing wife, Natasha, for her patience and encouragement in this project, and her support in all of my ambitions. Most of all, I'd like to thank the Python community at large. Though I have contributed to the support of various open source projects, languages, clubs, and the like, no work has been so gratifying and rewarding as that which has been in the service of the Python community.

Brian K. Jones

http://www.protocolostomy.com

https://twitter.com/bkjones

Table of Contents

Preface xi

1. Data Structures and Algorithms 1

1.1. Unpacking a Sequence into Separate Variables 1
1.2. Unpacking Elements from Iterables of Arbitrary Length 3
1.3. Keeping the Last N Items 5
1.4. Finding the Largest or Smallest N Items 7
1.5. Implementing a Priority Queue 8
1.6. Mapping Keys to Multiple Values in a Dictionary 11
1.7. Keeping Dictionaries in Order 12
1.8. Calculating with Dictionaries 13
1.9. Finding Commonalities in Two Dictionaries 15
1.10. Removing Duplicates from a Sequence while Maintaining Order 17
1.11. Naming a Slice 18
1.12. Determining the Most Frequently Occurring Items in a Sequence 20
1.13. Sorting a List of Dictionaries by a Common Key 21
1.14. Sorting Objects Without Native Comparison Support 23
1.15. Grouping Records Together Based on a Field 24
1.16. Filtering Sequence Elements 26
1.17. Extracting a Subset of a Dictionary 28
1.18. Mapping Names to Sequence Elements 29
1.19. Transforming and Reducing Data at the Same Time 32
1.20. Combining Multiple Mappings into a Single Mapping 33

2. Strings and Text 37

2.1. Splitting Strings on Any of Multiple Delimiters 37
2.2. Matching Text at the Start or End of a String 38
2.3. Matching Strings Using Shell Wildcard Patterns 40
2.4. Matching and Searching for Text Patterns 42

2.5. Searching and Replacing Text 45
2.6. Searching and Replacing Case-Insensitive Text 46
2.7. Specifying a Regular Expression for the Shortest Match 47
2.8. Writing a Regular Expression for Multiline Patterns 48
2.9. Normalizing Unicode Text to a Standard Representation 50
2.10. Working with Unicode Characters in Regular Expressions 52
2.11. Stripping Unwanted Characters from Strings 53
2.12. Sanitizing and Cleaning Up Text 54
2.13. Aligning Text Strings 57
2.14. Combining and Concatenating Strings 58
2.15. Interpolating Variables in Strings 61
2.16. Reformatting Text to a Fixed Number of Columns 64
2.17. Handling HTML and XML Entities in Text 65
2.18. Tokenizing Text 66
2.19. Writing a Simple Recursive Descent Parser 69
2.20. Performing Text Operations on Byte Strings 78

3. Numbers, Dates, and Times.. 83
3.1. Rounding Numerical Values 83
3.2. Performing Accurate Decimal Calculations 84
3.3. Formatting Numbers for Output 87
3.4. Working with Binary, Octal, and Hexadecimal Integers 89
3.5. Packing and Unpacking Large Integers from Bytes 90
3.6. Performing Complex-Valued Math 92
3.7. Working with Infinity and NaNs 94
3.8. Calculating with Fractions 96
3.9. Calculating with Large Numerical Arrays 97
3.10. Performing Matrix and Linear Algebra Calculations 100
3.11. Picking Things at Random 102
3.12. Converting Days to Seconds, and Other Basic Time Conversions 104
3.13. Determining Last Friday's Date 106
3.14. Finding the Date Range for the Current Month 107
3.15. Converting Strings into Datetimes 109
3.16. Manipulating Dates Involving Time Zones 110

4. Iterators and Generators.. 113
4.1. Manually Consuming an Iterator 113
4.2. Delegating Iteration 114
4.3. Creating New Iteration Patterns with Generators 115
4.4. Implementing the Iterator Protocol 117
4.5. Iterating in Reverse 119
4.6. Defining Generator Functions with Extra State 120

试读结束：需要全本请在线购买： www.ertongbook.com

4.7. Taking a Slice of an Iterator 122
4.8. Skipping the First Part of an Iterable 123
4.9. Iterating Over All Possible Combinations or Permutations 125
4.10. Iterating Over the Index-Value Pairs of a Sequence 127
4.11. Iterating Over Multiple Sequences Simultaneously 129
4.12. Iterating on Items in Separate Containers 131
4.13. Creating Data Processing Pipelines 132
4.14. Flattening a Nested Sequence 135
4.15. Iterating in Sorted Order Over Merged Sorted Iterables 136
4.16. Replacing Infinite while Loops with an Iterator 138

5. Files and I/O. 141
5.1. Reading and Writing Text Data 141
5.2. Printing to a File 144
5.3. Printing with a Different Separator or Line Ending 144
5.4. Reading and Writing Binary Data 145
5.5. Writing to a File That Doesn't Already Exist 147
5.6. Performing I/O Operations on a String 148
5.7. Reading and Writing Compressed Datafiles 149
5.8. Iterating Over Fixed-Sized Records 151
5.9. Reading Binary Data into a Mutable Buffer 152
5.10. Memory Mapping Binary Files 153
5.11. Manipulating Pathnames 156
5.12. Testing for the Existence of a File 157
5.13. Getting a Directory Listing 158
5.14. Bypassing Filename Encoding 160
5.15. Printing Bad Filenames 161
5.16. Adding or Changing the Encoding of an Already Open File 163
5.17. Writing Bytes to a Text File 165
5.18. Wrapping an Existing File Descriptor As a File Object 166
5.19. Making Temporary Files and Directories 167
5.20. Communicating with Serial Ports 170
5.21. Serializing Python Objects 171

6. Data Encoding and Processing. 175
6.1. Reading and Writing CSV Data 175
6.2. Reading and Writing JSON Data 179
6.3. Parsing Simple XML Data 183
6.4. Parsing Huge XML Files Incrementally 186
6.5. Turning a Dictionary into XML 189
6.6. Parsing, Modifying, and Rewriting XML 191
6.7. Parsing XML Documents with Namespaces 193

6.8. Interacting with a Relational Database 195
6.9. Decoding and Encoding Hexadecimal Digits 197
6.10. Decoding and Encoding Base64 199
6.11. Reading and Writing Binary Arrays of Structures 199
6.12. Reading Nested and Variable-Sized Binary Structures 203
6.13. Summarizing Data and Performing Statistics 214

7. Functions.. 217
7.1. Writing Functions That Accept Any Number of Arguments 217
7.2. Writing Functions That Only Accept Keyword Arguments 219
7.3. Attaching Informational Metadata to Function Arguments 220
7.4. Returning Multiple Values from a Function 221
7.5. Defining Functions with Default Arguments 222
7.6. Defining Anonymous or Inline Functions 224
7.7. Capturing Variables in Anonymous Functions 225
7.8. Making an N-Argument Callable Work As a Callable with Fewer Arguments 227
7.9. Replacing Single Method Classes with Functions 231
7.10. Carrying Extra State with Callback Functions 232
7.11. Inlining Callback Functions 235
7.12. Accessing Variables Defined Inside a Closure 238

8. Classes and Objects.. 243
8.1. Changing the String Representation of Instances 243
8.2. Customizing String Formatting 245
8.3. Making Objects Support the Context-Management Protocol 246
8.4. Saving Memory When Creating a Large Number of Instances 248
8.5. Encapsulating Names in a Class 250
8.6. Creating Managed Attributes 251
8.7. Calling a Method on a Parent Class 256
8.8. Extending a Property in a Subclass 260
8.9. Creating a New Kind of Class or Instance Attribute 264
8.10. Using Lazily Computed Properties 267
8.11. Simplifying the Initialization of Data Structures 270
8.12. Defining an Interface or Abstract Base Class 274
8.13. Implementing a Data Model or Type System 277
8.14. Implementing Custom Containers 283
8.15. Delegating Attribute Access 287
8.16. Defining More Than One Constructor in a Class 291
8.17. Creating an Instance Without Invoking *init* 293
8.18. Extending Classes with Mixins 294
8.19. Implementing Stateful Objects or State Machines 299

8.20. Calling a Method on an Object Given the Name As a String 305
8.21. Implementing the Visitor Pattern 306
8.22. Implementing the Visitor Pattern Without Recursion 311
8.23. Managing Memory in Cyclic Data Structures 317
8.24. Making Classes Support Comparison Operations 321
8.25. Creating Cached Instances 323

9. Metaprogramming.. 329
9.1. Putting a Wrapper Around a Function 329
9.2. Preserving Function Metadata When Writing Decorators 331
9.3. Unwrapping a Decorator 333
9.4. Defining a Decorator That Takes Arguments 334
9.5. Defining a Decorator with User Adjustable Attributes 336
9.6. Defining a Decorator That Takes an Optional Argument 339
9.7. Enforcing Type Checking on a Function Using a Decorator 341
9.8. Defining Decorators As Part of a Class 345
9.9. Defining Decorators As Classes 347
9.10. Applying Decorators to Class and Static Methods 350
9.11. Writing Decorators That Add Arguments to Wrapped Functions 352
9.12. Using Decorators to Patch Class Definitions 355
9.13. Using a Metaclass to Control Instance Creation 356
9.14. Capturing Class Attribute Definition Order 359
9.15. Defining a Metaclass That Takes Optional Arguments 362
9.16. Enforcing an Argument Signature on *args and **kwargs 364
9.17. Enforcing Coding Conventions in Classes 367
9.18. Defining Classes Programmatically 370
9.19. Initializing Class Members at Definition Time 374
9.20. Implementing Multiple Dispatch with Function Annotations 376
9.21. Avoiding Repetitive Property Methods 382
9.22. Defining Context Managers the Easy Way 384
9.23. Executing Code with Local Side Effects 386
9.24. Parsing and Analyzing Python Source 388
9.25. Disassembling Python Byte Code 392

10. Modules and Packages.. 397
10.1. Making a Hierarchical Package of Modules 397
10.2. Controlling the Import of Everything 398
10.3. Importing Package Submodules Using Relative Names 399
10.4. Splitting a Module into Multiple Files 401
10.5. Making Separate Directories of Code Import Under a Common Namespace 404
10.6. Reloading Modules 406

10.7. Making a Directory or Zip File Runnable As a Main Script 407
10.8. Reading Datafiles Within a Package 408
10.9. Adding Directories to sys.path 409
10.10. Importing Modules Using a Name Given in a String 411
10.11. Loading Modules from a Remote Machine Using Import Hooks 412
10.12. Patching Modules on Import 428
10.13. Installing Packages Just for Yourself 431
10.14. Creating a New Python Environment 432
10.15. Distributing Packages 433

11. Network and Web Programming. 437
11.1. Interacting with HTTP Services As a Client 437
11.2. Creating a TCP Server 441
11.3. Creating a UDP Server 445
11.4. Generating a Range of IP Addresses from a CIDR Address 447
11.5. Creating a Simple REST-Based Interface 449
11.6. Implementing a Simple Remote Procedure Call with XML-RPC 454
11.7. Communicating Simply Between Interpreters 456
11.8. Implementing Remote Procedure Calls 458
11.9. Authenticating Clients Simply 461
11.10. Adding SSL to Network Services 464
11.11. Passing a Socket File Descriptor Between Processes 470
11.12. Understanding Event-Driven I/O 475
11.13. Sending and Receiving Large Arrays 481

12. Concurrency. 485
12.1. Starting and Stopping Threads 485
12.2. Determining If a Thread Has Started 488
12.3. Communicating Between Threads 491
12.4. Locking Critical Sections 497
12.5. Locking with Deadlock Avoidance 500
12.6. Storing Thread-Specific State 504
12.7. Creating a Thread Pool 505
12.8. Performing Simple Parallel Programming 509
12.9. Dealing with the GIL (and How to Stop Worrying About It) 513
12.10. Defining an Actor Task 516
12.11. Implementing Publish/Subscribe Messaging 520
12.12. Using Generators As an Alternative to Threads 524
12.13. Polling Multiple Thread Queues 531
12.14. Launching a Daemon Process on Unix 534

13. Utility Scripting and System Administration. 539

13.1. Accepting Script Input via Redirection, Pipes, or Input Files 539
13.2. Terminating a Program with an Error Message 540
13.3. Parsing Command-Line Options 541
13.4. Prompting for a Password at Runtime 544
13.5. Getting the Terminal Size 545
13.6. Executing an External Command and Getting Its Output 545
13.7. Copying or Moving Files and Directories 547
13.8. Creating and Unpacking Archives 549
13.9. Finding Files by Name 550
13.10. Reading Configuration Files 552
13.11. Adding Logging to Simple Scripts 555
13.12. Adding Logging to Libraries 558
13.13. Making a Stopwatch Timer 559
13.14. Putting Limits on Memory and CPU Usage 561
13.15. Launching a Web Browser 563

14. Testing, Debugging, and Exceptions. 565
14.1. Testing Output Sent to stdout 565
14.2. Patching Objects in Unit Tests 567
14.3. Testing for Exceptional Conditions in Unit Tests 570
14.4. Logging Test Output to a File 572
14.5. Skipping or Anticipating Test Failures 573
14.6. Handling Multiple Exceptions 574
14.7. Catching All Exceptions 576
14.8. Creating Custom Exceptions 578
14.9. Raising an Exception in Response to Another Exception 580
14.10. Reraising the Last Exception 582
14.11. Issuing Warning Messages 583
14.12. Debugging Basic Program Crashes 585
14.13. Profiling and Timing Your Program 587
14.14. Making Your Programs Run Faster 590

15. C Extensions. 597
15.1. Accessing C Code Using ctypes 599
15.2. Writing a Simple C Extension Module 605
15.3. Writing an Extension Function That Operates on Arrays 609
15.4. Managing Opaque Pointers in C Extension Modules 612
15.5. Defining and Exporting C APIs from Extension Modules 614
15.6. Calling Python from C 619
15.7. Releasing the GIL in C Extensions 625
15.8. Mixing Threads from C and Python 625
15.9. Wrapping C Code with Swig 627

15.10. Wrapping Existing C Code with Cython 632
15.11. Using Cython to Write High-Performance Array Operations 638
15.12. Turning a Function Pointer into a Callable 643
15.13. Passing NULL-Terminated Strings to C Libraries 644
15.14. Passing Unicode Strings to C Libraries 648
15.15. Converting C Strings to Python 653
15.16. Working with C Strings of Dubious Encoding 654
15.17. Passing Filenames to C Extensions 657
15.18. Passing Open Files to C Extensions 658
15.19. Reading File-Like Objects from C 659
15.20. Consuming an Iterable from C 662
15.21. Diagnosing Segmentation Faults 663

A. Further Reading. 665

Index. 667

试读结束：需要全本请在线购买： www.ertongbook.com