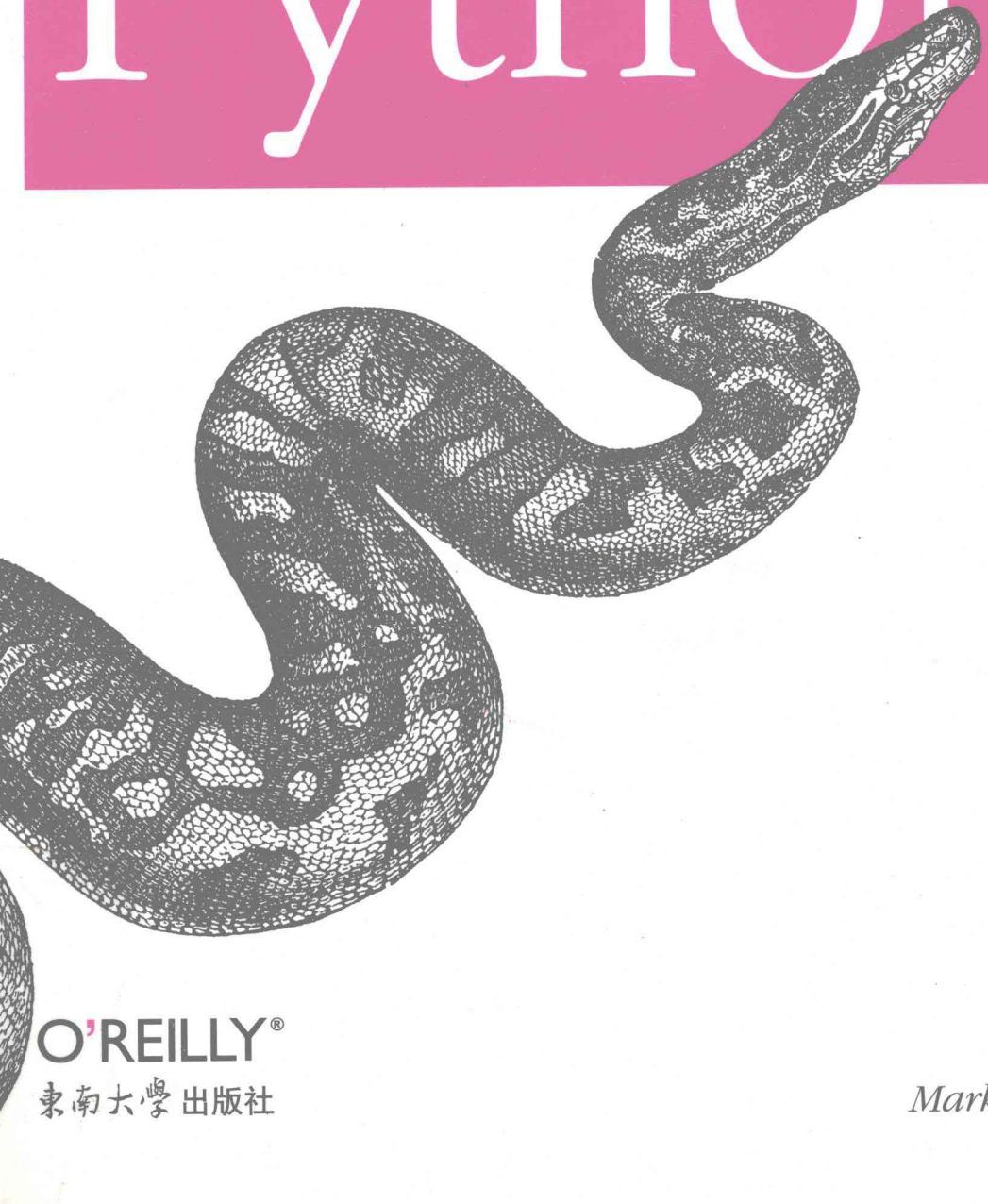
Python编程(影印版)

第四版
上册

Programming

Python



O'REILLY®
東南大學出版社

Mark Lutz 著

（第4版）

Python编程（影印版）

Programming Python

上 册

Mark Lutz

O'REILLY®

Beijing · Cambridge · Farnham · Köln · Sebastopol · Tokyo

O'Reilly Media, Inc.授权东南大学出版社出版

东南大学出版社

图书在版编目（CIP）数据

Python编程：第4版：英文/（美）鲁兹（Lutz, M.）著.—影印本.—南京：东南大学出版社，2011.5

书名原文：Programming Python, 4E

ISBN 978-7-5641-2687-2

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

中国版本图书馆CIP数据核字（2011）第047965号

江苏省版权局著作权合同登记

图字：10-2010-291号

Python编程 第4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：103.25印张

字　　数：2557千字

版　　次：2011年5月第1版

印　　次：2011年5月第1次印刷

书　　号：ISBN 978-7-5641-2687-2

定　　价：148.00元（上下册）

本社图书若有印装质量问题，请直接与读者服务部联系。电话（传真）：025-83792328

Python编程（影印版）

Programming Python

上 册

Preface

"And Now for Something Completely Different..."

This book explores ways to apply the Python programming language in common application domains and realistically scaled tasks. It's about what you can *do* with the language once you've mastered its fundamentals.

This book assumes you are relatively new to each of the application domains it covers—GUIs, the Internet, databases, systems programming, and so on—and presents each from the ground up, in *tutorial* fashion. Along the way, it focuses on commonly used tools and libraries, rather than language fundamentals. The net result is a resource that provides readers with an in-depth understanding of Python's roles in practical, real-world programming work.

As a subtheme, this book also explores Python's relevance as a *software development* tool—a role that many would classify as well beyond those typically associated with "scripting." In fact, many of this book's examples are scaled specifically for this purpose; among these, we'll incrementally develop email clients that top out at thousands of lines of code. Programming at this full scale will always be challenging work, but we'll find that it's also substantially quicker and easier when done with Python.

This Fourth Edition has been updated to present the language, libraries, and practice of Python 3.X. Specifically, its examples use Python 3.1—the most recent version of Python at the time of writing—and its major examples were tested successfully under the third alpha release of Python 3.2 just prior to publication, but they reflect the version of the language common to the entire 3.X line. This edition has also been reorganized in ways that both streamline some of its former material and allow for coverage of newly emerged tools and topics.

Because this edition's readership will include both newcomers as well as prior edition veterans, I want to use this Preface to expand on this book's purpose and scope before we jump into code.

About This Book

This book is a tutorial introduction to using Python in common application domains and tasks. It teaches how to apply Python for system administration, GUIs, and the Web, and explores its roles in networking, databases, frontend scripting layers, text processing, and more. Although the Python language is used along the way, this book's focus is on *application* to real-world tasks instead of language fundamentals.

This Book's Ecosystem

Because of its scope, this book is designed to work best as the second of a two-volume set, and to be supplemented by a third. Most importantly, this book is an applications programming follow-up to the core language book *Learning Python*, whose subjects are officially prerequisite material here. Here's how the three books are related:

- *Learning Python* covers the fundamentals of Python programming in depth. It focuses on the core Python language, and its topics are prerequisite to this book.
- *Programming Python*, this book, covers the application of Python to real-world programming tasks. It focuses on libraries and tools, and it assumes you already know Python fundamentals.
- *Python Pocket Reference* provides a quick reference to details not listed exhaustively here. It doesn't teach much, but it allows you to look up details fast.

In some sense, this book is to application programming what *Learning Python* is to the core language—a gradual tutorial, which makes almost no assumptions about your background and presents each topic from the ground up. By studying this book's coverage of Web basics, for example, you'll be equipped to build simple websites, and you will be able to make sense of more advanced frameworks and tools as your needs evolve. GUIs are similarly taught incrementally, from basic to advanced.

In addition, this book is designed to be supplemented by the quick-reference book *Python Pocket Reference*, which provides the small details finessed here and serves as a resource for looking up the fine points. That book is reference only, and is largely void of both examples and narrative, but it serves to augment and complement both *Learning Python*'s fundamentals and *Programming Python*'s applications. Because its current Fourth Edition gives both Python 2.X and 3.X versions of the tools it covers, that book also serves as a resource for readers transitioning between the two Python lines (more on this in a moment).*

* Disclosure: I am the author of all three books mentioned in this section, which affords me the luxury of tightly controlling their scopes in order to avoid overlap. It also means that as an author, I try to avoid commenting on the many other Python books available, some of which are very good and may cover topics not addressed in any of my own books. Please see the Web for other Python resources. All three of my books reflect my 13 years on the Python training trail and stem from the original *Programming Python* written back in 1995 <insert grizzled prospector photo here>.

What This Book Is Not

Because of the scopes carved out by the related books I just mentioned, this book's scope follows two explicit constraints:

- It does not cover Python language fundamentals
- It is not intended as a language reference

The former of these constraints reflects the fact that core language topics are the exclusive domain of *Learning Python*, and I encourage you to consult that book before tackling this one if you are completely new to the Python language, as its topics are assumed here. Some language techniques are shown by example in this book too, of course, and the larger examples here illustrate how core concepts come together into realistic programs. OOP, for example, is often best sampled in the context of the larger programs we'll write here. Officially, though, this book assumes you already know enough Python fundamentals to understand its example code. Our focus here is mostly on libraries and tools; please see other resources if the basic code we'll use in that role is unclear.

The latter of the two constraints listed above reflects what has been a common misconception about this book over the years (indeed, this book might have been better titled *Applying Python* had we been more clairvoyant in 1995). I want to make this as clear as I can: this is *not* a reference book. It is a *tutorial*. Although you can hunt for some details using the index and table of contents, this book is not designed for that purpose. Instead, *Python Pocket Reference* provides the sort of quick reference to details that you'll find useful once you start writing nontrivial code on your own. There are other reference-focused resources available, including other books and Python's own reference manuals set. Here, the goal is a gradual tutorial that teaches you how to apply Python to common tasks but does not document minute details exhaustively.

About This Fourth Edition

If this is the first edition of this book you've seen, you're probably less interested in recent changes, and you should feel free to skip ahead past this section. For readers of prior editions, though, this Fourth Edition of this book has changed in three important ways:

- It's been updated to cover Python 3.X (only).
- It's been slimmed down to sharpen its focus and make room for new topics.
- It's been updated for newly emerged topics and tools in the Python world.

The first of these is probably the most significant—this edition employs the Python 3.X language, its version of the standard library, and the common practice of its users. To better explain how this and the other two changes take shape in this edition, though, I need to fill in a few more details.

Specific Changes in This Edition

Because the prior versions of this book were widely read, here is a quick rundown of some of the most prominent specific changes in this edition:

Its existing material was shortened to allow for new topics

The prior edition of this book was also a 1600-page volume, which didn't allow much room for covering new Python topics (Python 3.X's Unicode orientation alone implies much new material). Luckily, recent changes in the Python world have allowed us to pare down some less critical existing material this time around, in order to free up room for new coverage.

Depth was not sacrificed in the process, of course, and this is still just as substantial a book as before. In general, though, avoiding new growth was a primary goal of this update; many of the other specific changes and removals I'll mention below were made, in part, to help accommodate new topics.

It covers 3.X (only)

This book's examples and narrative have been updated to reflect and use the 3.X version of Python. Python 2.X is no longer supported here, except where 3.X and 2.X Pythons overlap. Although the overlap is large enough to make this of use to 2.X readers too, this is now officially a 3.X-only text.

This turns out to be a major factor behind the lack of growth in this edition. By restricting our scope to Python 3.X—the incompatible successor to the Python 2.X line, and considered to be Python's future—we were able to avoid doubling the coverage size in places where the two Python lines differ. This version limit is especially important in a book like this that is largely about more advanced examples, which can be listed in only one version's style.

For readers who still straddle the 2.X and 3.X worlds, I'll say more about Python 3.X changes later in this Preface. Probably the most significant 3.X-related change described there is the new Internationalization support in PyEdit and PyMailGUI; though 2.X had Unicode too, its new prominence in 3.X almost forces such systems to rethink their former ASCII-only ways.

Inclusion of newly emerged libraries and tools

Since the prior edition, a variety of new libraries and tools have either come online or risen in popularity, and they get new mention here. This includes new standard library tools such as `subprocess` (in Chapters 2 and 3) and `multiprocessing` (in Chapter 5), as well as new third-party web frameworks and ORM database toolkits. Most of these are not covered extensively (many popular third-party extensions are complex systems in their own right and are best covered by dedicated books), but they are at the least introduced in summary form here.

For example, Python 3.1's new `tkinter.ttk` Tk themed widget set shows up in Chapter 7 now, but only briefly; as a rule, this edition prefers to mention such extensions in passing, rather than attempting to show you code without adequate explanation.

This Preface was tightened up

I've removed all the instructions for using and running program examples. Instead, please consult the `README` file in the examples distribution for example usage details. Moreover, most of the original acknowledgments are gone here because they are redundant with those in *Learning Python*; since that book is now considered a prerequisite, duplication of material here is unwarranted. A description of book contents was also deleted; please see the table of contents for a preview of this book's structure.

The initial Python overview chapter is gone

I've removed the prior edition's "managerial summary" chapter which introduced Python's strong points, prominent users, philosophies, and so on. Proselytizing does play an important role in a field that sometimes asks the "why" questions less often than it should. Indeed, if advocacy had not been part of the Python experience, we'd probably all be using Perl or shell languages today!

However, this chapter has now grown completely redundant with a similar chapter in Learning Python. Since that book is a precursor to this one, I opted to not devote space to restating "Pythonista" propaganda here (fun as it may be). Instead, this book assumes you already know why Python is worth using, and we jump right into applying it here.

The conclusion's postscripts are gone

This book's conclusion comes from the first edition, and it is now 15 years old. Naturally, some of it reflects the Python mindset from that period more than that of today. For example, its focus on Python's role in hybrid applications seemed more important in 1995 than in 2010; in today's much larger Python world, most Python users never deal with linked-in C code at all.

In prior editions, I added postscripts for each edition to elaborate on and update the ideas presented in the book's conclusion. These postscripts are gone now, replaced by a short note at the start of the conclusion. I opted to keep the conclusion itself, though, because it's still relevant to many readers and bears some historic value. Well, that, plus the jokes...

The forewords are gone

For reasons similar to those of the prior two points, the accumulated forewords from the prior three editions were also dropped this time around. You can read all about Python creator Guido van Rossum's historical rationale for Python's evolution in numerous places on the Web, if you are so inclined. If you are interested in how Python has changed technically over the years, see also the "What's New" documents that are part of the Python standard manuals set (available at *http://www.python.org/doc*, and installed alongside Python on Windows and other platforms).

The C integration part has been reduced to just one chapter

I've reduced the C extending and embedding part's material to one shorter chapter at the end of the tools part, which briefly introduces the core concepts in this

domain. Only a fraction of Python users must care about linking in C libraries today, and those who do already have the skills required to read the larger and more complete examples of integration present in the source code of Python itself. There is still enough to hint at possibilities here, but vast amounts of C code have been cut, in deference to the better examples you'll find in Python's own code.

The systems programming part was condensed and reworked
: The former two larger system examples chapters have been merged into one shorter one, with new or greatly rewritten examples. In fact, this part (Part II) was probably overhauled the most of any part in the book. It incorporates new tools such as `subprocess` and `multiprocessing`, introduces sockets earlier, and removes dated topics and examples still lingering from prior editions. Frankly, a few of the file-oriented examples here dated back to the 1990s, and were overdue for a general refresh. The initial chapter in this part was also split into two to make its material easier to read (shell context, including streams, gets its own chapter now), and a few large program listings here (including the auto-configuring launcher scripts) are now external suggested reading.

Some larger examples were removed (but are available in the examples distribution)
: Along the same lines, two of the larger GUI examples in the prior edition, *PyTree* and *PyForm*, have been removed. Instead, their updated code is available in the book's examples distribution package, as suggested supplemental reading. You'll still find many larger examples covered and listed in this edition—including both GUI- and Web-based renderings of full-featured email clients, along with image viewers, calculators, clocks, Unicode-aware text editors, drawing programs, regression test scripts, and more. However, because the code of the examples removed doesn't add much to what is already covered, and because they were already largely self-study examples anyhow, I've made them optional and external to the printed text in this edition.

The advanced Internet topics chapter was replaced by brief summaries
: I've cut the advanced Internet topics chapter completely, leaving only simple summaries at the start of the Internet part (intentionally mirroring the GUI option summaries at the start of the GUI part). This includes prior coverage for tools such as the ZOPE web framework, COM, Windows active scripting and ASP, HTMLgen, Python Server Pages (PSP), Jython, and the now very dated Grail system. Some of these systems still receive honorable mention in the summaries, but none are now presented in any sort of detail. Summaries of new tools (including many of those listed in the following paragraph) were added to this set, but again, in brief fashion with no example code.

  Despite authors' best attempts to foresee the future, the Web domain evolves faster than books like this can. For instance, Web frameworks like Django, Google's App Engine, TurboGears, pylons, and web2py are now popular alternatives to ZOPE. Similarly, the .NET framework supersedes much of COM on Windows; IronPython now provides the same type of integration for .NET as Jython did first

for Java; and active scripting has been eclipsed by AJAX and JavaScript-oriented frameworks on the client such as Flex, Silverlight, and pyjamas (generally known today as rich Internet applications, RIAs). Culture shift aside, the examples formerly presented in this category were by themselves also insufficient to either teach or do justice to the subject tools.

Rather than including incomplete (and nearly useless) coverage of tools that are prone to both evolution and demise during this edition's expected lifespan, I now provide only brief overviews of the current hot topics in the Web domain, and I encourage readers to search the Web for more details. More to the point, the goal of the book you're reading is to impart the sort of in-depth knowledge of Internet and Web fundamentals that will allow you to use more advanced systems well, when you're ready to take the leap.

One exception here: the XML material of this prior chapter was spared and relocated in expanded form to the text processing chapter (where it probably belonged all along). In a related vein, the coverage of ZOPE's ZODB object-oriented database was retained, although it was shortened radically to allow new coverage of ORMs such as SQLObject and SQLAlchemy (again, in overview form).

Use of tools available for 3.X today

At this writing, Python 3.X is still in its adoption phase, and some of the third-party tools that this book formerly employed in its examples are still available in Python 2.X form only. To work around this temporary flux, I've changed some code to use alternatives that already support 3.X today.

The most notable of these is the SQL database section—this now uses the in-process SQLite library, which is a standard part of Python and already in 3.X form, rather than the enterprise-level MySQL interface which is still at 2.X today. Luckily, the Python portable database API allows scripts to work largely the same on both, so this is a minor pragmatic sacrifice.

Of special note, the PIL extension used to display JPEGs in the GUI part was ported to 3.1 just when it was needed for this update, thanks to Fredrik Lundh. It's still not officially released in 3.X form as I submit the final draft of this book in July 2010, but it should be soon, and 3.X patches are provided in the book examples package as a temporary measure.

Advanced core language topics are not covered here

More advanced Python language tools such as descriptors, properties, decorators, metaclasses, and Unicode text processing basics are all part of the core Python language. Because of that, they are covered in the Fourth Edition of Learning Python, not here. For example, Unicode text and the changes it implies for files, filenames, sockets, and much more are discussed as encountered here, but the fundamentals of Unicode itself are not presented in complete depth. Some of the topics in this category are arguably application-level related too (or at least of interest to tool builders and API developers in general), but their coverage in *Learning*

Python allows us to avoid additional growth here. Please see that book for more on these subjects.

Other random bits

Naturally, there were additional smaller changes made along the way. For example, tkinter's `grid` method is used instead of `pack` for layout of most input forms, because it yields a more consistent layout on platforms where label font sizes don't match up with entry widget height (including on a Windows 7 netbook laptop, this edition's development machine). There's also new material scattered throughout, including a new exploration of redirecting streams to sockets in the Internet part; a new threaded and Unicode-aware "grep" dialog and process-wide change tests on exit in the *PyEdit* example; and other things you are probably better off uncovering along the way than reading further about in this Preface.

I also finally replaced some remaining "#" comment blocks at the top of source files with docstrings (even, for consistency, in scripts not meant to be imported, though some "#" lines are retained in larger examples to offset the text); changed a few lingering "while 1" to "while True"; use += more often; and cleaned up a few other cases of now-dated coding patterns. Old habits may die hard, but such updates make the examples both more functional and more representative of common practice today.

Although new topics were added, all told, four chapters were cut outright (the nontechnical introduction, one of the system example chapters, advanced Internet topics, and one integration chapter), some additional examples and material were trimmed (including PyForm and PyTree), and focus was deliberately restricted to Python 3.X and application fundamentals to conserve space.

What's Left, Then?

The combined effect of all the changes just outlined is that this edition more concisely and sharply reflects its core focus—that of a tutorial introduction to ways to apply Python in common programming domains. Nevertheless, as you can tell from this book's page count, it is still a substantial and *in-depth* book, designed to be a first step on your path to mastering realistic applications of Python.

Contrary to recent trends (and at some risk of being branded a heretic), I firmly believe that the job of books like this one is to elevate their readers, not pander to them. Lowering the intellectual bar does a disservice both to readers and to the fields in which they hope to work. While that means you won't find as many cartoons in this book as in some, this book also won't insult you by emphasizing entertainment at the expense of technical depth. Instead, the goal of my books is to impart sophisticated concepts in a satisfying and substantive way and to equip you with the tools you'll need in the real world of software development.

There are many types of learners, of course, and no one book can ever satisfy every possible audience. In fact, that's why the original version of this book later became two, with language basics delegated to Learning Python. Moreover, one can make a case for a distinction between *programmers*, who must acquire deep software development skills, and *scripters*, who do not. For some, a rudimentary knowledge of programming may be enough to leverage a system or library that solves the problem at hand. That is, until their coding forays start encroaching on the realm of full-scale software engineering—a threshold that can inspire disappointment at worst, but a better appreciation of the challenging nature of this field at best.

No matter which camp you're from, it's important to understand this book's intent upfront. If you're looking for a shortcut to proficiency that's light on technical content, you probably won't be happy with this book (or the software field in general). If your goal is to master programming Python well, though, and have some fun along the way, you'll probably find this book to be an important piece of your learning experience.

At the end of the day, learning to program well is much more demanding than implied by some contemporary media. If you're willing to invest the focus and effort required, though, you'll find that it's also much more rewarding. This is especially true for those who equip themselves for the journey with a programmer-friendly tool like Python. While no book or class can turn you into a Python "Master of the Universe" by itself, this book's goal is to help you get there, by shortening your start-up time and providing a solid foundation in Python's most common application domains.

Python 3.X Impacts on This Book

As mentioned, this edition now covers Python 3.X only. Python 3.X is an incompatible version of the language. The 3.X core language itself is very similar to Python 2.X, but there are substantial changes in both the language and its many standard libraries. Although some readers with no prior background in 2.X may be able to bypass the differences, the changes had a big impact on the content of this edition. For the still very large existing Python 2.X user base, this section documents the most noteworthy changes in this category.

If you're interested in 2.X differences, I also suggest finding a copy of the Fourth Edition of the book *Python Pocket Reference* described earlier. That book gives both 2.X and 3.X versions of core language structures, built-in functions and exceptions, and many of the standard library modules and tools used in this book. Though not designed to be a reference or version translator per se, the Fourth Edition of *Learning Python* similarly covers both 2.X and 3.X, and as stated, is prerequisite material to this book. The goal of this 3.X-only *Programming Python* is not to abandon the current vast 2.X user base in favor of a still imaginary one for 3.X; it is to help readers with the migration, and avoid doubling the size of an already massive book.

Specific 3.X Changes

Luckily, many of the 2.X/3.X differences that impact this book's presentation are trivial. For instance, the `tkinter` GUI toolkit, used extensively in this book, is shown under its 3.X `tkinter` name and package structure only; its 2.X `Tkinter` module incarnation is not described. This mostly boils down to different import statements, but only their Python 3 versions are given here. Similarly, to satisfy 3.X module naming conventions, 2.X's `anydbm`, `Queue`, `thread`, `StringIO.StringIO`, and `urllib.open` become `dbm`, `queue`, `_thread`, `io.StringIO`, and `urllib.request.urlopen`, respectively, in both Python 3.X and this edition. Other tools are similarly renamed.

On the other hand, 3.X implies broader idiomatic changes which are, of course, more radical. For example, Python 3.X's new Unicode awareness has inspired fully Internationalized versions of the PyEdit text editor and the PyMailGUI email client examples in this edition (more on this in a moment). Furthermore: the replacement of `os.popen2` with the `subprocess` module required new examples; the demise of `os.path.walk` in favor of `os.walk` allowed some examples to be trimmed; the new Unicode and binary dichotomy of files and strings impacted a host of additional existing examples and material; and new modules such as `multiprocessing` offer new options covered in this edition.

Beyond such library changes, core language changes in Python 3 are also reflected in this book's example code. For instance, changes to 2.X's `print`, `raw_input`, `keys`, `has_key`, `map`, and `apply` all required changes here. In addition, 3.X's new *package-relative* import model impacted a few examples including `mailtools` and expression parsers, and its different flavor of *division* forced some minor math updates in canvas-based GUI examples such as PyClock, PyDraw, and PyPhoto.

Of note here, I did not change all `%` *string formatting* expressions to use the new `str.format`, since both forms are supported in Python 3.1, and it now appears that they will be either indefinitely or forever. In fact, per a "grep" we'll build and run in Chapter 11's PyEdit example, it seems that this expression still appears over 3,000 times in Python 3.1's own library code. Since I cannot predict Python evolution completely, see the first chapter for more on this if it ever requires updates in an unexpected future.

Also because of the 3.X scope, this edition is unable to use some third-party packages that are still in 2.X form only, as described earlier. This includes the leading MySQL interface, ZODB, PyCrypto, and others; as also mentioned, PIL was ported to 3.1 for use in this book, but this required a special patch and an official 3.X release is still presently pending. Many of these may be available in 3.X form by the time you read these words, assuming the Python world can either break some of the current cross dependencies in 2.X packages or adopt new 3.X-only tools.

Language Versus Library: Unicode

As a book focused on applications instead of core language fundamentals, language changes are not always obtrusive here. Indeed, in retrospect the book *Learning Python* may have been affected by 3.X core language changes more than this book. In most cases here, more example changes were probably made in the name of clarity or functionality than in support of 3.X itself.

On the other hand, Python 3.X does impact much code, and the impacts can be subtle at times. Readers with Python 2.X backgrounds will find that while 3.X core language changes are often simple to apply, updates required for changes in the 3.X standard library are sometimes more far reaching.

Chief among these, Python 3.X's Unicode strings have had broad ramifications. Let's be honest: to people who have spent their lives in an ASCII world, the impacts of the 3.X Unicode model can be downright aggravating at times! As we'll see in this book, it affects file content; file names; pipe descriptors; sockets; text in GUIs; Internet protocols such as FTP and email; CGI scripts; and even some persistence tools. For better or worse, once we reach the world of applications programming as covered in this book, Unicode is no longer an optional topic for many or most Python 3.X programmers.

Of course, Unicode arguably never should have been entirely optional for many programmers in the first place. Indeed, we'll find that things that may have appeared to work in 2.X never really did—treating text as raw byte strings can mask issues such as comparison results across encodings (see the grep utility of Chapter 11's PyEdit for a prime example of code that should fail in the face of Unicode mismatches). Python 3.X elevates such issues to potentially every programmer's panorama.

Still, porting nontrivial code to 3.X is not at all an insurmountable task. Moreover, many readers of this edition have the luxury of approaching Python 3.X as their first Python and need not deal with existing 2.X code. If this is your case, you'll find Python 3.X to be a robust and widely applicable scripting and programming language, which addresses head-on many issues that once lurked in the shadows in 2.X.

Python 3.1 Limitations: Email, CGI

There's one exception that I should call out here because of its impact on major book examples. In order to make its code relevant to the widest possible audience, this book's major examples are related to Internet *email* and have much new support in this edition for Internationalization and Unicode in this domain. Chapter 14's PyMailGUI and Chapter 16's PyMailCGI, and all the prior examples they reuse, fall into this category. This includes the PyEdit text editor—now Unicode-aware for files, display, and greps.

On this front, there is both proverbial good news and bad. The good news is that in the end, we will be able to develop the feature-rich and fully Internationalized PyMailGUI email client in this book, using the `email` package as it currently exists. This will include support for arbitrary encodings in both text content and message headers, for

both viewing and composing messages. The less happy news is that this will come at some cost in workaround complexity in Python 3.1.

Unfortunately, as we'll learn in Chapter 13, the `email` package in Python 3.1 has a number of issues related to `str`/`bytes` combinations in Python 3.X. For example, there's no simple way to guess the encoding needed to convert mail `bytes` returned by the `poplib` module to the `str` expected by the `email` parser. Moreover, the `email` package is currently broken altogether for some types of messages, and it has uneven or type-specific support for some others.

This situation appears to be temporary. Some of the issues encountered in this book are already scheduled to be repaired (in fact, one such fix in 3.2 required a last-minute patch to one of this book's 3.1 workarounds in Chapter 13). Furthermore, a new version of `email` is being developed to accommodate the 3.X Unicode/bytes dichotomy more accurately, but it won't materialize until long after this book is published, and it might be backward-incompatible with the current package's API, much like Python 3.X itself. Because of that, this book both codes workarounds and makes some assumption along the way, but please watch its website (described ahead) for required updates in future Pythons. One upside here is that the dilemmas posed neatly reflect those common in realistic programming—an underlying theme of this text.

These issues in the `email` package are also inherited by the `cgi` module for CGI file uploads, which are in large measure broken in 3.1. CGI scripts are a basic technique eclipsed by many web frameworks today, but they still serve as an entry-level way to learn Web fundamentals and are still at the heart of many larger toolkits. A future fix seems likely for this 3.1 flaw as well, but we have to make do with nonbinary CGI file uploads for this edition in Chapters 15 and 16, and limited email attachments in PyMailCGI. This seems less than ideal nearly two years after 3.0's release, but such is life in the dynamic worlds of both software development at large and books that aim to lead the curve instead of following it.

Using Book Examples

Because this book's examples form much of its content, I want to say a few words about them up front.

Where to Look for Examples and Updates

As before, examples, updates, corrections, and supplements for this book will be maintained at the author's website, which lives officially at the following URL:

http://www.rmi.net/~lutz/about-pp4e.html

This page at my book support website will contain links to all supplemental information related to this version of the book. Because I don't own that domain name, though, if

that link ceases to be during this book's shelf life, try the following alternative site as a fallback option:

http://learning-python.com/books/about-pp4e.html (alternative location)

If neither of those links work, try a general web search (which, of course, is what most readers will probably try first anyhow).

Wherever it may live, this website (as well as O'Reilly's, described in the next section) is where you can fetch the book *examples distribution package*—an archive file containing all of the book's examples, as well as some extras that are mentioned but not listed in the book itself. To work along without having to type the examples manually, download the package, unpack it, and consult its *README.txt* file for usage details. I'll describe how example labels and system prompts in this book imply file locations in the package when we use our first script in the first chapter.

As for the first three editions, I will also be maintaining an informal "blog" on this website that describes Python changes over time and provides general book-related notes and updates that you should consider a supplemental appendix to this text.

O'Reilly's website for this book, described later in this Preface, also has an errata report system, and you can report issues at either my site or O'Reilly's. I tend to keep my book websites more up to date, but it's not impossible that O'Reilly's errata page may supersede mine for this edition. In any event, you should consider the union of these two lists to be the official word on book corrections and updates.

Example Portability

The examples in this book were all developed, tested, and run under Windows 7, and Python 3.1. The book's major examples were all tested and ran successfully on the upcoming Python 3.2, too (its alpha 3 release), just before the book went to the printer, so most or all of this book applies to Python 3.2 as well. In addition, the C code of Chapter 20 and a handful of parallel programming examples were run under Cygwin on Windows to emulate a Unix environment.

Although Python and its libraries are generally platform neutral, some of this book's code may require minor changes to run on other platforms, such as Mac OS X, Linux, and other Unix variants. The tkinter GUI examples, as well as some systems programming scripts, may be especially susceptible to platform differences. Some portability issues are pointed out along the way, but others may not be explicitly noted.

Since I had neither time nor budget to test on and accommodate all possible machines that readers might use over the lifespan of this book, updates for platform-specific behaviors will have to fall into the suggested exercises category. If you find a platform dependency and wish to submit a patch for it, though, please see the updates site listed earlier; I'll be happy to post any platform patches from readers there.