Powerful Object-Oriented Programming

4th Edition
Covers Python 2.6 and 3.x

Learning

Python

O'REILLY®

Mark Lutz

Learning Python

FOURTH EDITION

Learning Python

Mark Lutz

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

Learning Python, Fourth Edition
by Mark Lutz

Printed in the United States of America.

Published by O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472.

O'Reilly books may be purchased for educational, business, or sales promotional use. Online editions are also available for most titles (*http://my.safaribooksonline.com*). For more information, contact our corporate/institutional sales department: (800) 998-9938 or *corporate@oreilly.com*.

Editor: Julie Steele
Production Editor: Sumita Mukherji
Copyeditor: Rachel Head
Production Services: Newgen North America
Indexer: John Bickelhaupt
Cover Designer: Karen Montgomery
Interior Designer: David Futato
Illustrator: Robert Romano

Printing History:

March 1999:	First Edition.
December 2003:	Second Edition.
October 2007:	Third Edition.
September 2009:	Fourth Edition.

ISBN: 978-0-596-15806-4

[M]

1252944666

To Vera.

You are my life.

Table of Contents

Part II. Types and Operations

Part V. Modules

Part VII. Exceptions and Tools

Part VIII. Advanced Topics

Preface

This book provides an introduction to the Python programming language. *Python* is a popular open source programming language used for both standalone programs and scripting applications in a wide variety of domains. It is free, portable, powerful, and remarkably easy and fun to use. Programmers from every corner of the software industry have found Python's focus on developer productivity and software quality to be a strategic advantage in projects both large and small.

Whether you are new to programming or are a professional developer, this book's goal is to bring you quickly up to speed on the fundamentals of the core Python language. After reading this book, you will know enough about Python to apply it in whatever application domains you choose to explore.

By design, this book is a tutorial that focuses on the *core Python language* itself, rather than specific applications of it. As such, it's intended to serve as the first in a two-volume set:

- *Learning Python*, this book, teaches Python itself.
- *Programming Python*, among others, shows what you can do with Python after you've learned it.

That is, applications-focused books such as *Programming Python* pick up where this book leaves off, exploring Python's role in common domains such as the Web, graphical user interfaces (GUIs), and databases. In addition, the book *Python Pocket Reference* provides additional reference materials not included here, and it is designed to supplement this book.

Because of this book's foundations focus, though, it is able to present Python fundamentals with more depth than many programmers see when first learning the language. And because it's based upon a three-day Python training class with quizzes and exercises throughout, this book serves as a self-paced introduction to the language.

About This Fourth Edition

This fourth edition of this book has changed in three ways. This edition:

- Covers both Python 3.0 and Python 2.6—it emphasizes 3.0, but notes differences in 2.6
- Includes a set of new chapters mainly targeted at advanced core-language topics
- Reorganizes some existing material and expands it with new examples for clarity

As I write this edition in 2009, Python comes in two flavors—version 3.0 is an emerging and incompatible mutation of the language, and 2.6 retains backward compatibility with the vast body of existing Python code. Although Python 3 is viewed as the future of Python, Python 2 is still widely used and will be supported in parallel with Python 3 for years to come. While 3.0 is largely the same language, it runs almost no code written for prior releases (the mutation of `print` from statement to function alone, aesthetically sound as it may be, breaks nearly every Python program ever written).

This split presents a bit of a dilemma for both programmers and book authors. While it would be easier for a book to pretend that Python 2 never existed and cover 3 only, this would not address the needs of the large Python user base that exists today. A vast amount of existing code was written for Python 2, and it won't be going away any time soon. And while newcomers to the language can focus on Python 3, anyone who must use code written in the past needs to keep one foot in the Python 2 world today. Since it may be years before all third-party libraries and extensions are ported to Python 3, this fork might not be entirely temporary.

Coverage for Both 3.0 and 2.6

To address this dichotomy and to meet the needs of all potential readers, this edition of this book has been updated to cover *both* Python 3.0 and Python 2.6 (and later releases in the 3.X and 2.X lines). It's intended for programmers using Python 2, programmers using Python 3, and programmers stuck somewhere between the two.

That is, you can use this book to learn either Python line. Although the focus here is on 3.0 primarily, 2.6 differences and tools are also noted along the way for programmers using older code. While the two versions are largely the same, they diverge in some important ways, and I'll point these out along the way.

For instance, I'll use 3.0 `print` calls in most examples, but will describe the 2.6 `print` statement, too, so you can make sense of earlier code. I'll also freely introduce new features, such as the `nonlocal` statement in 3.0 and the string `format` method in 2.6 and 3.0, and will point out when such extensions are not present in older Pythons.

If you are learning Python for the first time and don't need to use any legacy code, I encourage you to begin with Python 3.0; it cleans up some longstanding warts in the language, while retaining all the original core ideas and adding some nice new tools.

Many popular Python libraries and tools will likely be available for Python 3.0 by the time you read these words, especially given the file I/O performance improvements expected in the upcoming 3.1 release. If you are using a system based on Python 2.X, however, you'll find that this book addresses your concerns, too, and will help you migrate to 3.0 in the future.

By proxy, this edition addresses other Python version 2 and 3 releases as well, though some older version 2.X code may not be able to run all the examples here. Although class decorators are available in both Python 2.6 and 3.0, for example, you cannot use them in an older Python 2.X that did not yet have this feature. See Tables P-1 and P-2 later in this Preface for summaries of 2.6 and 3.0 changes.

Shortly before going to press, this book was also augmented with notes about prominent extensions in the upcoming Python 3.1 release—comma separators and automatic field numbering in string `format` method calls, multiple context manager syntax in `with` statements, new methods for numbers, and so on. Because Python 3.1 was targeted primarily at optimization, this book applies directly to this new release as well. In fact, because Python 3.1 supersedes 3.0, and because the latest Python is usually the best Python to fetch and use anyhow, in this book the term "Python 3.0" generally refers to the language variations introduced by Python 3.0 but that are present in the entire 3.X line.

New Chapters

Although the main purpose of this edition is to update the examples and material from the preceding edition for 3.0 and 2.6, I've also added five new chapters to address new topics and add context:

- Chapter 27 is a new class tutorial, using a more realistic example to explore the basics of Python object-oriented programming (OOP).
- Chapter 36 provides details on Unicode and byte strings and outlines string and file differences between 3.0 and 2.6.
- Chapter 37 collects managed attribute tools such as properties and provides new coverage of descriptors.
- Chapter 38 presents function and class decorators and works through comprehensive examples.
- Chapter 39 covers metaclasses and compares and contrasts them with decorators.

The first of these chapters provides a gradual, step-by-step tutorial for using classes and OOP in Python. It's based upon a live demonstration I have been using in recent years in the training classes I teach, but has been honed here for use in a book. The chapter is designed to show OOP in a more realistic context than earlier examples and to

illustrate how class concepts come together into larger, working programs. I hope it works as well here as it has in live classes.

The last four of these new chapters are collected in a new final part of the book, "Advanced Topics." Although these are technically core language topics, not every Python programmer needs to delve into the details of Unicode text or metaclasses. Because of this, these four chapters have been separated out into this new part, and are officially *optional reading*. The details of Unicode and binary data strings, for example, have been moved to this final part because most programmers use simple ASCII strings and don't need to know about these topics. Similarly, decorators and metaclasses are specialist topics that are usually of more interest to API builders than application programmers.

If you do use such tools, though, or use code that does, these new advanced topic chapters should help you master the basics. In addition, these chapters' examples include case studies that tie core language concepts together, and they are more substantial than those in most of the rest of the book. Because this new part is optional reading, it has end-of-chapter quizzes but no end-of-part exercises.

Changes to Existing Material

In addition, some material from the prior edition has been *reorganized*, or supplemented with *new examples*. Multiple inheritance, for instance, gets a new case study example that lists class trees in Chapter 30; new examples for generators that manually implement `map` and `zip` are provided in Chapter 20; static and class methods are illustrated by new code in Chapter 31; package relative imports are captured in action in Chapter 23; and the `__contains__`, `__bool__`, and `__index__` operator overloading methods are illustrated by example now as well in Chapter 29, along with the new overloading protocols for slicing and comparison.

This edition also incorporates some reorganization for clarity. For instance, to accommodate new material and topics, and to avoid chapter topic overload, five prior chapters have been split into two each here. The result is new standalone chapters on operator overloading, scopes and arguments, exception statement details, and comprehension and iteration topics. Some reordering has been done within the existing chapters as well, to improve topic flow.

This edition also tries to minimize forward references with some reordering, though Python 3.0's changes make this impossible in some cases: to understand printing and the string `format` method, you now must know keyword arguments for functions; to understand dictionary key lists and key tests, you must now know iteration; to use `exec` to run code, you need to be able to use file objects; and so on. A linear reading still probably makes the most sense, but some topics may require nonlinear jumps and random lookups.

All told, there have been hundreds of changes in this edition. The next section's tables alone document 27 additions and 57 changes in Python. In fact, it's fair to say that this

edition is somewhat more advanced, because Python is somewhat more advanced. As for Python 3.0 itself, though, you're probably better off discovering most of this book's changes for yourself, rather than reading about them further in this Preface.

Specific Language Extensions in 2.6 and 3.0

In general, Python 3.0 is a *cleaner* language, but it is also in some ways a more *sophisticated* language. In fact, some of its changes seem to assume you must already know Python in order to learn Python! The prior section outlined some of the more prominent circular knowledge dependencies in 3.0; as a random example, the rationale for wrapping dictionary views in a `list` call is incredibly subtle and requires substantial foreknowledge. Besides teaching Python fundamentals, this book serves to help bridge this knowledge gap.

Table P-1 lists the most prominent new language features covered in this edition, along with the primary chapters in which they appear.

Table P-1. Extensions in Python 2.6 and 3.0

Extension	Covered in chapter(s)
The `print` function in 3.0	11
The `nonlocal x,y` statement in 3.0	17
The `str.format` method in 2.6 and 3.0	7
String types in 3.0: `str` for Unicode text, `bytes` for binary data	7, 36
Text and binary file distinctions in 3.0	9, 36
Class decorators in 2.6 and 3.0: `@private('age')`	31, 38
New iterators in 3.0: `range`, `map`, `zip`	14, 20
Dictionary views in 3.0: `D.keys`, `D.values`, `D.items`	8, 14
Division operators in 3.0: remainders, `/` and `//`	5
Set literals in 3.0: `{a, b, c}`	5
Set comprehensions in 3.0: `{x**2 for x in seq}`	4, 5, 14, 20
Dictionary comprehensions in 3.0: `{x: x**2 for x in seq}`	4, 8, 14, 20
Binary digit-string support in 2.6 and 3.0: `0b0101`, `bin(I)`	5
The fraction number type in 2.6 and 3.0: `Fraction(1, 3)`	5
Function annotations in 3.0: `def f(a:99, b:str)->int`	19
Keyword-only arguments in 3.0: `def f(a, *b, c, **d)`	18, 20
Extended sequence unpacking in 3.0: `a, *b = seq`	11, 13
Relative import syntax for packages enabled in 3.0: `from .`	23
Context managers enabled in 2.6 and 3.0: `with/as`	33, 35
Exception syntax changes in 3.0: `raise`, `except/as`, superclass	33, 34

Extension	Covered in chapter(s)
Exception chaining in 3.0: `raise e2 from e1`	33
Reserved word changes in 2.6 and 3.0	11
New-style class cutover in 3.0	31
Property decorators in 2.6 and 3.0: `@property`	37
Descriptor use in 2.6 and 3.0	31, 38
Metaclass use in 2.6 and 3.0	31, 39
Abstract base classes support in 2.6 and 3.0	28

Specific Language Removals in 3.0

In addition to extensions, a number of language tools have been removed in 3.0 in an effort to clean up its design. Table P-2 summarizes the changes that impact this book, covered in various chapters of this edition. Many of the removals listed in Table P-2 have direct replacements, some of which are also available in 2.6 to support future migration to 3.0.

Table P-2. Removals in Python 3.0 that impact this book

Removed	Replacement	Covered in chapter(s)
`reload(M)`	`imp.reload(M)` (or `exec`)	3, 22
`apply(f, ps, ks)`	`f(*ps, **ks)`	18
`` `X` ``	`repr(X)`	5
`X <> Y`	`X != Y`	5
`long`	`int`	5
`9999L`	`9999`	5
`D.has_key(K)`	`K in D` (or `D.get(key) != None`)	8
`raw_input`	`input`	3, 10
old `input`	`eval(input())`	3
`xrange`	`range`	14
`file`	`open` (and `io` module classes)	9
`X.next`	`X.__next__`, called by `next(X)`	14, 20, 29
`X.__getslice__`	`X.__getitem__` passed a `slice` object	7, 29
`X.__setslice__`	`X.__setitem__` passed a `slice` object	7, 29
`reduce`	`functools.reduce` (or loop code)	14, 19
`execfile(filename)`	`exec(open(filename).read())`	3
`exec open(filename)`	`exec(open(filename).read())`	3
`0777`	`0o777`	5
`print x, y`	`print(x, y)`	11

Removed	Replacement	Covered in chapter(s)
`print >> F, x, y`	`print(x, y, file=F)`	11
`print x, y,`	`print(x, y, end=' ')`	11
`u'ccc'`	`'ccc'`	7, 36
`'bbb'` for byte strings	`b'bbb'`	7, 9, 36
`raise E, V`	`raise E(V)`	32, 33, 34
`except E, X:`	`except E as X:`	32, 33, 34
`def f((a, b)):`	`def f(x): (a, b) = x`	11, 18, 20
`file.xreadlines`	`for line in file:` (or `X=iter(file)`)	13, 14
`D.keys()`, etc. as lists	`list(D.keys())` (dictionary views)	8, 14
`map()`, `range()`, etc. as lists	`list(map())`, `list(range())` (built-ins)	14
`map(None, ...)`	`zip` (or manual code to pad results)	13, 20
`X=D.keys(); X.sort()`	`sorted(D)` (or `list(D.keys())`)	4, 8, 14
`cmp(x, y)`	`(x > y) - (x < y)`	29
`X.__cmp__(y)`	`__lt__`, `__gt__`, `__eq__`, etc.	29
`X.__nonzero__`	`X.__bool__`	29
`X.__hex__`, `X.__oct__`	`X._index__`	29
Sort comparison functions	Use `key=transform` or `reverse=True`	8
Dictionary <, >, <=, >=	Compare `sorted(D.items())` (or loop code)	8, 9
`types.ListType`	`list` (`types` is for nonbuilt-in names only)	9
`__metaclass__ = M`	`class C(metaclass=M):`	28, 31, 39
`__builtin__`	`builtins` (renamed)	17
`Tkinter`	`tkinter` (renamed)	18, 19, 24, 29, 30
`sys.exc_type`, `exc_value`	`sys.exc_info()[0]`, `[1]`	34, 35
`function.func_code`	`function.__code__`	19, 38
`__getattr__` run by built-ins	Redefine `__X__` methods in wrapper classes	30, 37, 38
`-t`, `-tt` command-line switches	Inconsistent tabs/spaces use is always an error	10, 12
`from ... *`, within a function	May only appear at the top level of a file	22
`import mod`, in same package	`from . import mod`, package-relative form	23
`class MyException:`	`class MyException(Exception):`	34
`exceptions` module	Built-in scope, library manual	34
`thread`, `Queue` modules	`_thread`, `queue` (both renamed)	17
`anydbm` module	`dbm` (renamed)	27
`cPickle` module	`_pickle` (renamed, used automatically)	9
`os.popen2/3/4`	`subprocess.Popen` (`os.popen` retained)	14
String-based exceptions	Class-based exceptions (also required in 2.6)	32, 33, 34

Removed	Replacement	Covered in chapter(s)
String module functions	String object methods	7
Unbound methods	Functions (`staticmethod` to call via instance)	30, 31
Mixed type comparisons, sorts	Nonnumeric mixed type comparisons are errors	5, 9

There are additional changes in Python 3.0 that are not listed in this table, simply because they don't affect this book. Changes in the standard library, for instance, might have a larger impact on applications-focused books like *Programming Python* than they do here; although most standard library functionality is still present, Python 3.0 takes further liberties with renaming modules, grouping them into packages, and so on. For a more comprehensive list of changes in 3.0, see the "What's New in Python 3.0" document in Python's standard manual set.

If you are migrating from Python 2.X to Python 3.X, be sure to also see the *2to3* automatic code conversion script that is available with Python 3.0. It can't translate everything, but it does a reasonable job of converting the majority of 2.X code to run under 3.X. As I write this, a new *3to2* back-conversion project is also underway to translate Python 3.X code to run in 2.X environments. Either tool may prove useful if you must maintain code for both Python lines; see the Web for details.

Because this fourth edition is mostly a fairly straightforward update for 3.0 with a handful of new chapters, and because it's only been two years since the prior edition was published, the rest of this Preface is taken from the prior edition with only minor updating.

About The Third Edition

In the four years between the publication of the second and third editions of this book there were substantial changes in Python itself, and in the topics I presented in Python training sessions. The third edition reflected these changes, and also incorporated a handful of structural changes.

The Third Edition's Python Language Changes

On the language front, the third edition was thoroughly updated to reflect Python 2.5 and all changes to the language since the publication of the second edition in late 2003. (The second edition was based largely on Python 2.2, with some 2.3 features grafted on at the end of the project.) In addition, discussions of anticipated changes in the upcoming Python 3.0 release were incorporated where appropriate. Here are some of the major language topics for which new or expanded coverage was provided (chapter numbers here have been updated to reflect the fourth edition):

- The new `B if A else C` conditional expression (Chapter 19)
- `with`/`as` context managers (Chapter 33)
- `try`/`except`/`finally` unification (Chapter 33)
- Relative import syntax (Chapter 23)
- Generator expressions (Chapter 20)
- New generator function features (Chapter 20)
- Function decorators (Chapter 31)
- The set object type (Chapter 5)
- New built-in functions: `sorted`, `sum`, `any`, `all`, `enumerate` (Chapters 13 and 14)
- The decimal fixed-precision object type (Chapter 5)
- Files, list comprehensions, and iterators (Chapters 14 and 20)
- New development tools: Eclipse, `distutils`, `unittest` and `doctest`, IDLE enhancements, Shedskin, and so on (Chapters 2 and 35)

Smaller language changes (for instance, the widespread use of `True` and `False`; the new `sys.exc_info` for fetching exception details; and the demise of string-based exceptions, string methods, and the `apply` and `reduce` built-ins) are discussed throughout the book. The third edition also expanded coverage of some of the features that were new in the second edition, including three-limit slices and the arbitrary arguments call syntax that subsumed `apply`.

The Third Edition's Python Training Changes

Besides such language changes, the third edition was augmented with new topics and examples presented in my Python training sessions. Changes included (chapter numbers again updated to reflect those in the fourth edition):

- A new chapter introducing built-in types (Chapter 4)
- A new chapter introducing statement syntax (Chapter 10)
- A new full chapter on dynamic typing, with enhanced coverage (Chapter 6)
- An expanded OOP introduction (Chapter 25)
- New examples for files, scopes, statement nesting, classes, exceptions, and more

Many additions and changes were made with Python beginners in mind, and some topics were moved to appear at the places where they proved simplest to digest in training classes. List comprehensions and iterators, for example, now make their initial appearance in conjunction with the `for` loop statement, instead of later with functional tools.

Coverage of many original core language topics also was substantially expanded in the third edition, with new discussions and examples added. Because this text has become something of a de facto standard resource for learning the core Python language, the presentation was made more complete and augmented with new use cases throughout.

In addition, a new set of Python tips and tricks, gleaned from 10 years of teaching classes and 15 years of using Python for real work, was incorporated, and the exercises were updated and expanded to reflect current Python best practices, new language features, and common beginners' mistakes witnessed firsthand in classes. Overall, the core language coverage was expanded.

The Third Edition's Structural Changes

Because the material was more complete, it was split into bite-sized chunks. The core language material was organized into many multichapter parts to make it easier to tackle. Types and statements, for instance, are now two top-level parts, with one chapter for each major type and statement topic. Exercises and "gotchas" (common mistakes) were also moved from chapter ends to part ends, appearing at the end of the last chapter in each part.

In the third edition, I also augmented the end-of-part exercises with end-of-chapter summaries and end-of-chapter quizzes to help you review chapters as you complete them. Each chapter concludes with a set of questions to help you review and test your understanding of the chapter's material. Unlike the end-of-part exercises, whose solutions are presented in Appendix B, the solutions to the end-of-chapter quizzes appear immediately after the questions; I encourage you to look at the solutions even if you're sure you've answered the questions correctly because the answers are a sort of review in themselves.

Despite all the new topics, the book is still oriented toward Python newcomers and is designed to be a first Python text for programmers. Because it is largely based on time-tested training experience and materials, it can still serve as a self-paced introductory Python class.

The Third Edition's Scope Changes

As of its third edition, this book is intended as a tutorial on the core Python language, and nothing else. It's about learning the language in an in-depth fashion, before applying it in application-level programming. The presentation here is bottom-up and gradual, but it provides a complete look at the entire language, in isolation from its application roles.

For some, "learning Python" involves spending an hour or two going through a tutorial on the Web. This works for already advanced programmers, up to a point; Python is, after all, relatively simple in comparison to other languages. The problem with this fast-track approach is that its practitioners eventually stumble onto unusual cases and get

stuck—variables change out from under them, mutable default arguments mutate inexplicably, and so on. The goal here is instead to provide a solid grounding in Python fundamentals, so that even the unusual cases will make sense when they crop up.

This scope is deliberate. By restricting our gaze to language fundamentals, we can investigate them here in more satisfying depth. Other texts, described ahead, pick up where this book leaves off and provide a more complete look at application-level topics and additional reference materials. The purpose of the book you are reading now is solely to teach Python itself so that you can apply it to whatever domain you happen to work in.

About This Book

This section underscores some important points about this book in general, regardless of its edition number. No book addresses every possible audience, so it's important to understand a book's goals up front.

This Book's Prerequisites

There are no absolute prerequisites to speak of, really. Both true beginners and crusty programming veterans have used this book successfully. If you are motivated to learn Python, this text will probably work for you. In general, though, I have found that any exposure to programming or scripting before this book can be helpful, even if not required for every reader.

This book is designed to be an introductory-level Python text for programmers.* It may not be an ideal text for someone who has never touched a computer before (for instance, we're not going to spend any time exploring what a computer is), but I haven't made many assumptions about your programming background or education.

On the other hand, I won't insult readers by assuming they are "dummies," either, whatever that means—it's easy to do useful things in Python, and this book will show you how. The text occasionally contrasts Python with languages such as C, C++, Java, and Pascal, but you can safely ignore these comparisons if you haven't used such languages in the past.

This Book's Scope and Other Books

Although this book covers all the essentials of the Python language, I've kept its scope narrow in the interests of speed and size. To keep things simple, this book focuses on core concepts, uses small and self-contained examples to illustrate points, and

* And by "programmers," I mean anyone who has written a single line of code in any programming or scripting language in the past. If this doesn't include you, you will probably find this book useful anyhow, but be aware that it will spend more time teaching Python than programming fundamentals.

sometimes omits the small details that are readily available in reference manuals. Because of that, this book is probably best described as an introduction and a stepping-stone to more advanced and complete texts.

For example, we won't talk much about Python/C integration—a complex topic that is nevertheless central to many Python-based systems. We also won't talk much about Python's history or development processes. And popular Python applications such as GUIs, system tools, and network scripting get only a short glance, if they are mentioned at all. Naturally, this scope misses some of the big picture.

By and large, Python is about raising the quality bar a few notches in the scripting world. Some of its ideas require more context than can be provided here, and I'd be remiss if I didn't recommend further study after you finish this book. I hope that most readers of this book will eventually go on to gain a more complete understanding of application-level programming from other texts.

Because of its beginner's focus, *Learning Python* is designed to be naturally complemented by O'Reilly's other Python books. For instance, *Programming Python*, another book I authored, provides larger and more complete examples, along with tutorials on application programming techniques, and was explicitly designed to be a follow-up text to the one you are reading now. Roughly, the current editions of *Learning Python* and *Programming Python* reflect the two halves of their author's training materials—the core language, and application programming. In addition, O'Reilly's *Python Pocket Reference* serves as a quick reference supplement for looking up some of the finer details skipped here.

Other follow-up books can also provide references, additional examples, or details about using Python in specific domains such as the Web and GUIs. For instance, O'Reilly's *Python in a Nutshell* and Sams's *Python Essential Reference* serve as useful references, and O'Reilly's *Python Cookbook* offers a library of self-contained examples for people already familiar with application programming techniques. Because reading books is such a subjective experience, I encourage you to browse on your own to find advanced texts that suit your needs. Regardless of which books you choose, though, keep in mind that the rest of the Python story requires studying examples that are more realistic than there is space for here.

Having said that, I think you'll find this book to be a good first text on Python, despite its limited scope (and perhaps because of it). You'll learn everything you need to get started writing useful standalone Python programs and scripts. By the time you've finished this book, you will have learned not only the language itself, but also how to apply it well to your day-to-day tasks. And you'll be equipped to tackle more advanced topics and examples as they come your way.

This Book's Style and Structure

This book is based on training materials developed for a three-day hands-on Python course. You'll find quizzes at the end of each chapter, and exercises at the end of the last chapter of each part. Solutions to chapter quizzes appear in the chapters themselves, and solutions to part exercises show up in Appendix B. The quizzes are designed to review material, while the exercises are designed to get you coding right away and are usually one of the highlights of the course.

I strongly recommend working through the quizzes and exercises along the way, not only to gain Python programming experience, but also because some of the exercises raise issues not covered elsewhere in the book. The solutions in the chapters and in Appendix B should help you if you get stuck (and you are encouraged to peek at the answers as much and as often as you like).

The overall structure of this book is also derived from class materials. Because this text is designed to introduce language basics quickly, I've organized the presentation by major language features, not examples. We'll take a bottom-up approach here: from built-in object types, to statements, to program units, and so on. Each chapter is fairly self-contained, but later chapters draw upon ideas introduced in earlier ones (e.g., by the time we get to classes, I'll assume you know how to write functions), so a linear reading makes the most sense for most readers.

In general terms, this book presents the Python language in a linear fashion. It is organized with one part per major language feature—types, functions, and so forth—and most of the examples are small and self-contained (some might also call the examples in this text artificial, but they illustrate the points it aims to make). More specifically, here is what you will find:

Part I, *Getting Started*
: We begin with a general overview of Python that answers commonly asked initial questions—why people use the language, what it's useful for, and so on. The first chapter introduces the major ideas underlying the technology to give you some background context. Then the technical material of the book begins, as we explore the ways that both we and Python run programs. The goal of this part of the book is to give you just enough information to be able to follow along with later examples and exercises.

Part II, *Types and Operations*
: Next, we begin our tour of the Python language, studying Python's major built-in object types in depth: numbers, lists, dictionaries, and so on. You can get a lot done in Python with these tools alone. This is the most substantial part of the book because we lay groundwork here for later chapters. We'll also look at dynamic typing and its references—keys to using Python well—in this part.

Part III, *Statements and Syntax*

The next part moves on to introduce Python's *statements*—the code you type to create and process objects in Python. It also presents Python's general syntax model. Although this part focuses on syntax, it also introduces some related tools, such as the PyDoc system, and explores coding alternatives.

Part IV, *Functions*

This part begins our look at Python's higher-level program structure tools. *Functions* turn out to be a simple way to package code for reuse and avoid code redundancy. In this part, we will explore Python's scoping rules, argument-passing techniques, and more.

Part V, *Modules*

Python *modules* let you organize statements and functions into larger components, and this part illustrates how to create, use, and reload modules. We'll also look at some more advanced topics here, such as module packages, module reloading, and the `__name__` variable.

Part VI, *Classes and OOP*

Here, we explore Python's object-oriented programming tool, the *class*—an optional but powerful way to structure code for customization and reuse. As you'll see, classes mostly reuse ideas we will have covered by this point in the book, and OOP in Python is mostly about looking up names in linked objects. As you'll also see, OOP is optional in Python, but it can shave development time substantially, especially for long-term strategic project development.

Part VII, *Exceptions and Tools*

We conclude the language fundamentals coverage in this text with a look at Python's exception handling model and statements, plus a brief overview of development tools that will become more useful when you start writing larger programs (debugging and testing tools, for instance). Although exceptions are a fairly lightweight tool, this part appears after the discussion of classes because exceptions should now all be classes.

Part VIII, *Advanced Topics* (new in the fourth edition)

In the final part, we explore some advanced topics. Here, we study Unicode and byte strings, managed attribute tools like properties and descriptors, function and class decorators, and metaclasses. These chapters are all optional reading, because not all programmers need to understand the subjects they address. On the other hand, readers who must process internationalized text or binary data, or are responsible for developing APIs for other programmers to use, should find something of interest in this part.

Part IX, *Appendixes*

The book wraps up with a pair of appendixes that give platform-specific tips for using Python on various computers (Appendix A) and provide solutions to the end-of-part exercises (Appendix B). Solutions to end-of-chapter quizzes appear in the chapters themselves.

Note that the index and table of contents can be used to hunt for details, but there are no reference appendixes in this book (this book is a tutorial, not a reference). As mentioned earlier, you can consult *Python Pocket Reference*, as well as other books, and the free Python reference manuals maintained at *http://www.python.org* for syntax and built-in tool details.

Book Updates

Improvements happen (and so do mis^H^H^H typos). Updates, supplements, and corrections for this book will be maintained (or referenced) on the Web at one of the following sites:

http://www.oreilly.com/catalog/9780596158064 (O'Reilly's web page for the book)
http://www.rmi.net/~lutz (the author's site)
http://www.rmi.net/~lutz/about-lp.html (the author's web page for the book)

The last of these three URLs points to a web page for this book where I will post updates, but be sure to search the Web if this link becomes invalid. If I could become more clairvoyant, I would, but the Web changes faster than printed books.

About the Programs in This Book

This fourth edition of this book, and all the program examples in it, is based on Python version 3.0. In addition, most of its examples run under Python 2.6, as described in the text, and notes for Python 2.6 readers are mixed in along the way.

Because this text focuses on the core language, however, you can be fairly sure that most of what it has to say won't change very much in future releases of Python. Most of this book applies to earlier Python versions, too, except when it does not; naturally, if you try using extensions added after the release you've got, all bets are off.

As a rule of thumb, the latest Python is the best Python. Because this book focuses on the core language, most of it also applies to Jython, the Java-based Python language implementation, as well as other Python implementations described in Chapter 2.

Source code for the book's examples, as well as exercise solutions, can be fetched from the book's website at *http://www.oreilly.com/catalog/9780596158064/*. So, how do you run the examples? We'll study startup details in Chapter 3, so please stay tuned for information on this front.

Using Code Examples

This book is here to help you get your job done. In general, 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: "*Learning Python*, Fourth Edition, by Mark Lutz. Copyright 2009 Mark Lutz, 978-0-596-15806-4."

If you feel your use of code examples falls outside fair use or the permission given above, feel free to contact us at *permissions@oreilly.com*.

Font Conventions

This book uses the following typographical conventions:

Italic
: Used for email addresses, URLs, filenames, pathnames, and emphasizing new terms when they are first introduced

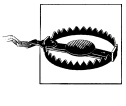
`Constant width`
: Used for the contents of files and the output from commands, and to designate modules, methods, statements, and commands

`Constant width bold`
: Used in code sections to show commands or text that would be typed by the user, and, occasionally, to highlight portions of code

`Constant width italic`
: Used for replaceables and some comments in code sections

`<Constant width>`
: Indicates a syntactic unit that should be replaced with real code

Indicates a tip, suggestion, or general note relating to the nearby text.



Indicates a warning or caution relating to the nearby text.

Notes specific to this book: In this book's examples, the `%` character at the start of a system command line stands for the system's prompt, whatever that may be on your machine (e.g., `C:\Python30>` in a DOS window). Don't type the `%` character (or the system prompt it sometimes stands for) yourself.

Similarly, in interpreter interaction listings, do not type the `>>>` and `...` characters shown at the start of lines—these are prompts that Python displays. Type just the text after these prompts. To help you remember this, user inputs are shown in bold font in this book.

Also, you normally don't need to type text that starts with a `#` in listings; as you'll learn, these are comments, not executable code.

Safari® Books Online

Safari Books Online is an on-demand digital library that lets you easily search over 7,500 technology and creative reference books and videos to find the answers you need quickly.

With a subscription, you can read any page and watch any video from our library online. Read books on your cell phone and mobile devices. Access new titles before they are available for print, and get exclusive access to manuscripts in development and post feedback for the authors. Copy and paste code samples, organize your favorites, download chapters, bookmark key sections, create notes, print out pages, and benefit from tons of other time-saving features.

O'Reilly Media has uploaded this book to the Safari Books Online service. To have full digital access to this book and others on similar topics from O'Reilly and other publishers, sign up for free at *http://my.safaribooksonline.com*.

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 will also maintain a web page for this book, where we list errata, examples, and any additional information. You can access this page at:

http://www.oreilly.com/catalog/9780596158064/

To comment or ask technical questions about this book, send email to:

bookquestions@oreilly.com

For more information about our books, conferences, Resource Centers, and the O'Reilly Network, see our website at:

http://www.oreilly.com

For book updates, be sure to also see the other links mentioned earlier in this Preface.

Acknowledgments

As I write this fourth edition of this book in 2009, I can't help but be in a sort of "mission accomplished" state of mind. I have now been using and promoting Python for 17 years, and have been teaching it for 12 years. Despite the passage of time and events, I am still constantly amazed at how successful Python has been over the years. It has grown in ways that most of us could not possibly have imagined in 1992. So, at the risk of sounding like a hopelessly self-absorbed author, you'll have to pardon a few words of reminiscing, congratulations, and thanks here.

It's been the proverbial long and winding road. Looking back today, when I first discovered Python in 1992, I had no idea what an impact it would have on the next 17 years of my life. Two years after writing the first edition of *Programming Python* in 1995, I began traveling around the country and the world teaching Python to beginners and experts. Since finishing the first edition of *Learning Python* in 1999, I've been an independent Python trainer and writer, thanks largely to Python's exponential growth in popularity.

As I write these words in mid-2009, I have written 12 Python books (4 editions of 3). I have also been teaching Python for more than a decade; have taught some 225 Python training sessions in the U.S., Europe, Canada, and Mexico; and have met over 3,000 students along the way. Besides racking up frequent flyer miles, these classes helped me refine this text as well as my other Python books. Over the years, teaching honed the books, and vice versa. In fact, the book you're reading is derived almost entirely from my classes.

Because of this, I'd like to thank all the students who have participated in my courses during the last 12 years. Along with changes in Python itself, your feedback played a huge role in shaping this text. (There's nothing quite as instructive as watching 3,000 students repeat the same beginner's mistakes!) This edition owes its changes primarily to classes held after 2003, though every class held since 1997 has in some way helped refine this book. I'd especially like to single out clients who hosted classes in Dublin, Mexico City, Barcelona, London, Edmonton, and Puerto Rico; better perks would be hard to imagine.

I'd also like to express my gratitude to everyone who played a part in producing this book. To the editors who worked on this project: Julie Steele on this edition, Tatiana

Apandi on the prior edition, and many others on earlier editions. To Doug Hellmann and Jesse Noller for taking part in the technical review of this book. And to O'Reilly for giving me a chance to work on those 12 book projects—it's been net fun (and only feels a little like the movie *Groundhog Day*).

I want to thank my original coauthor David Ascher as well for his work on the first two editions of this book. David contributed the "Outer Layers" part in prior editions, which we unfortunately had to trim to make room for new core language materials in the third edition. To compensate, I added a handful of more advanced programs as a self-study final exercise in the third edition, and added both new advanced examples and a new complete part for advanced topics in the fourth edition. Also see the prior notes in this Preface about follow-up application-level texts you may want to consult once you've learned the fundamentals here.

For creating such an enjoyable and useful language, I owe additional thanks to Guido van Rossum and the rest of the Python community. Like most open source systems, Python is the product of many heroic efforts. After 17 years of programming Python, I still find it to be seriously fun. It's been my privilege to watch Python grow from a new kid on the scripting languages block to a widely used tool, deployed in some fashion by almost every organization writing software. That has been an exciting endeavor to be a part of, and I'd like to thank and congratulate the entire Python community for a job well done.

I also want to thank my original editor at O'Reilly, the late Frank Willison. This book was largely Frank's idea, and it reflects the contagious vision he had. In looking back, Frank had a profound impact on both my own career and that of Python itself. It is not an exaggeration to say that Frank was responsible for much of the fun and success of Python when it was new. We still miss him.

Finally, a few personal notes of thanks. To OQO for the best toys so far (while they lasted). To the late Carl Sagan for inspiring an 18-year-old kid from Wisconsin. To my Mom, for courage. And to all the large corporations I've come across over the years, for reminding me how lucky I have been to be self-employed for the last decade!

To my children, Mike, Sammy, and Roxy, for whatever futures you will choose to make. You were children when I began with Python, and you seem to have somehow grown up along the way; I'm proud of you. Life may compel us down paths all our own, but there will always be a path home.

And most of all, to Vera, my best friend, my girlfriend, and my wife. The best day of my life was the day I finally found you. I don't know what the next 50 years hold, but I do know that I want to spend all of them holding you.

—Mark Lutz
Sarasota, Florida
July 2009