

Dive Into Python 3



Mark Pilgrim

Apress®

Dive Into Python 3

Copyright © 2009 by Mark Pilgrim

All rights reserved. No part of this work may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording, or by any information storage or retrieval system, without the prior written permission of the copyright owner and the publisher.

ISBN-13 (pbk): 978-1-4302-2415-0

ISBN-13 (electronic): 978-1-4302-2416-7

Printed and bound in the United States of America 9 8 7 6 5 4 3 2 1

Trademarked names may appear in this book. Rather than use a trademark symbol with every occurrence of a trademarked name, we use the names only in an editorial fashion and to the benefit of the trademark owner, with no intention of infringement of the trademark.

Lead Editor: Duncan Parkes

Technical Reviewer: Simon Willison

Editorial Board: Clay Andres, Steve Anglin, Mark Beckner, Ewan Buckingham, Tony Campbell, Gary Cornell, Jonathan Gennick, Michelle Lowman, Matthew Moodie, Jeffrey Pepper, Frank Pohlmann, Ben Renow-Clarke, Dominic Shakeshaft, Matt Wade, Tom Welsh

Project Managers: Richard Dal Porto and Debra Kelly

Copy Editors: Nancy Sixsmith, Heather Lang, Patrick Meader, and Sharon Terdeman

Compositor: folio 2

Indexer: Julie Grady

Distributed to the book trade worldwide by Springer-Verlag New York, Inc., 233 Spring Street, 6th Floor, New York, NY 10013. Phone 1-800-SPRINGER, fax 201-348-4505, e-mail orders-ny@springer-sbm.com, or visit <http://www.springeronline.com>.

For information on translations, please e-mail info@apress.com, or visit <http://www.apress.com>.

Apress and friends of ED books may be purchased in bulk for academic, corporate, or promotional use. eBook versions and licenses are also available for most titles. For more information, reference our Special Bulk Sales—eBook Licensing web page at <http://www.apress.com/info/bulksales>.

The information in this book is distributed on an “as is” basis, without warranty. Although every precaution has been taken in the preparation of this work, neither the author(s) nor Apress shall have any liability to any person or entity with respect to any loss or damage caused or alleged to be caused directly or indirectly by the information contained in this work.

For Michael

Contents at a Glance

■ Foreword	xiii
■ About the Author	xv
■ About the Technical Reviewer	xvi
■ Acknowledgments	xvii
■ Installing Python	xviii
■ Chapter 1: Your First Python Program	1
■ Chapter 2: Native Datatypes	15
■ Chapter 3: Comprehensions	43
■ Chapter 4: Strings	53
■ Chapter 5: Regular Expressions	69
■ Chapter 6: Closures and Generators	87
■ Chapter 7: Classes and Iterators	101
■ Chapter 8: Advanced Iterators	113
■ Chapter 9: Unit Testing	131
■ Chapter 10: Refactoring	155
■ Chapter 11: Files	167
■ Chapter 12: XML	185
■ Chapter 13: Serializing Python Objects	205
■ Chapter 14: HTTP Web Services	225
■ Chapter 15: Case Study: Porting chardet to Python 3	253
■ Chapter 16: Packaging Python Libraries	279
■ Appendix A: Porting Code to Python 3 with 2to3	295
■ Appendix B: Special Method Names	327
■ Appendix C: Where to Go From Here	345
■ Index	347

Contents

■ Foreword	xiii
■ About the Author	xv
■ About the Technical Reviewer	xvi
■ Acknowledgments	xvii
■ Installing Python	xviii
Which Python Is Right for You?	xi
Installing on Microsoft Windows	xii
Installing on Mac OS X	xxi
Installing on Ubuntu Linux	xxxiii
Installing on Other Platforms	xli
Using the Python Shell	xli
Python Editors and IDEs	xliv
■ Chapter 1: Your First Python Program	1
Declaring Functions	2
Optional and Named Arguments	3
Writing Readable Code	5
Documentation Strings	5
The import Search Path	6
Everything Is an Object	7
What's an Object?	8
Indenting Code	8
Exceptions	9
Catching Import Errors	11
Unbound Variables	12
Running Scripts	12
Further Reading Online	13

Chapter 2: Native Datatypes	15
Booleans.....	15
Numbers.....	16
Coercing Integers to Floats and Vice Versa.....	17
Common Numerical Operations.....	18
Fractions.....	19
Trigonometry.....	20
Numbers in a Boolean Context.....	20
Lists.....	21
Creating a List.....	21
Slicing a List.....	22
Adding Items to a List.....	23
Searching For Values in a List.....	25
Removing Items from a List.....	26
Removing Items from a List: Bonus Round.....	26
Lists in a Boolean Context.....	27
Tuples.....	28
Tuples in a Boolean Context.....	30
Assigning Multiple Values at Once.....	30
Sets.....	31
Modifying a Set.....	33
Removing Items from a Set.....	34
Common Set Operations.....	35
Sets in a Boolean Context.....	37
Dictionaries.....	38
Creating a Dictionary.....	38
Modifying a Dictionary.....	38
Mixed-Value Dictionaries.....	39
Dictionaries in a Boolean Context.....	40
None.....	41
None in a Boolean Context.....	41
Further Reading Online.....	42
Chapter 3: Comprehensions	43
Working With Files and Directories.....	43
The Current Working Directory.....	43
Working with Filenames and Directory Names.....	44
Listing Directories.....	46
Getting File Metadata.....	47
Constructing Absolute Pathnames.....	47
List Comprehensions.....	48
Dictionary Comprehensions.....	50

Fun with Dictionary Comprehensions	51
Set Comprehensions	51
Further Reading Online.....	52
Chapter 4: Strings	53
Unicode	54
Diving In.....	56
Formatting Strings.....	56
Compound Field Names.....	57
Format Specifiers.....	59
Other Common String Methods	60
Slicing a String	61
Strings versus Bytes	62
Character Encoding of Python Source Code.....	65
Further Reading Online.....	66
Chapter 5: Regular Expressions	69
Case Study: Street Addresses	69
Case Study: Roman Numerals	71
Checking for Thousands	72
Checking for Hundreds	73
Using the {n,m} Syntax	75
Checking for Tens and Ones	76
Verbose Regular Expressions	78
Case Study: Parsing Phone Numbers	80
Further Reading Online.....	85
Chapter 6: Closures and Generators.....	87
I Know, Let's Use Regular Expressions!.....	88
A List of Functions	90
A List of Patterns	92
A File of Patterns	95
Generators.....	96
A Fibonacci Generator	98
A Plural Rule Generator.....	99
Further Reading Online.....	100
Chapter 7: Classes and Iterators	101
Defining Classes.....	102
The <code>__init__()</code> Method.....	102
Instantiating Classes	103
Instance Variables	104
A Fibonacci Iterator	105
A Plural Rule Iterator.....	107
Further Reading Online.....	112

Chapter 8: Advanced Iterators	113
Finding All Occurrences of a Pattern.....	115
Finding the Unique Items in a Sequence	115
Making Assertions	117
Generator Expressions	117
Calculating Permutations ... the Lazy Way.....	119
Other Fun Stuff in the itertools Module	120
A New Kind of String Manipulation.....	124
Evaluating Arbitrary Strings as Python Expressions	126
Putting It All Together	130
Further Reading Online.....	130
Chapter 9: Unit Testing.....	131
A Single Question.....	132
Halt and Catch Fire.....	138
More Halting, More Fire.....	141
And One More Thing	144
A Pleasing Symmetry.....	146
More Bad Input	150
Chapter 10: Refactoring	155
Handling Changing Requirements.....	158
Refactoring.....	162
Further Reading Online.....	166
Chapter 11: Files	167
Reading from Text Files.....	167
Character Encoding Rears Its Ugly Head	168
Stream Objects	169
Reading Data from a Text File.....	170
Closing Files	172
Closing Files Automatically.....	173
Reading Data One Line at a Time.....	173
Writing to Text Files.....	175
Character Encoding Again	176
Binary Files.....	177
Streams Objects from Nonfile Sources.....	178
Handling Compressed Files	179
Standard Input, Output, and Error.....	180
Redirecting Standard Output.....	181
Further Reading Online.....	184
Chapter 12: XML.....	185
A 5-Minute Crash Course in XML	186

The Structure of an Atom Feed	189
Parsing XML	191
Elements Are Lists.....	192
Attributes Are Dictionaries.....	193
Searching for Nodes Within an XML Document.....	194
Going Further with lxml	197
Generating XML	199
Parsing Broken XML.....	202
Further Reading Online.....	204
Chapter 13: Serializing Python Objects	205
A Quick Note About the Examples in this Chapter	205
Saving Data to a Pickle File	206
Loading Data from a Pickle File.....	208
Pickling Without a File	209
Bytes and Strings Rear Their Ugly Heads Again	210
Debugging Pickle Files	210
Serializing Python Objects to be Read by Other Languages	213
Saving Data to a JSON File	213
Mapping Python Datatypes to JSON	215
Serializing Datatypes Unsupported by JSON.....	215
Loading Data from a JSON File	220
Further Reading Online.....	222
Chapter 14: HTTP Web Services	225
Features of HTTP	226
Caching.....	226
Last-Modified Checking	227
ETags.....	228
Compression	229
Redirects.....	229
How Not to Fetch Data Over HTTP	230
What's On the Wire?.....	231
Introducing httplib2.....	234
Caching with httplib2	237
Handling Last-Modified and ETag Headers with httplib2	240
Handling Compression with httplib2.....	242
Handling Redirects with httplib2	243
Beyond HTTP GET	246
Beyond HTTP POST.....	250
Further Reading Online.....	252
Chapter 15: Case Study: Porting chardet to Python 3	253
What Is Character Encoding Auto-Detection?.....	253

Why Auto-Detection Is Difficult.....	253
Auto-Encoding Algorithms.....	254
Introducing the chardet Module	254
UTF-n with a BOM	254
Escaped Encodings	254
Multibyte Encodings.....	255
Single-Byte Encodings.....	255
windows-1252	256
Running 2to3.....	256
A Short Digression Into Multi-File Modules	259
Fixing What 2to3 Can't	261
False Is Invalid Syntax	261
No Module Named Constants	262
Name 'file' Is Not Defined.....	263
Can't Use a String Pattern on a Bytes-Like Object.....	264
Can't Convert 'bytes' Object to str Implicitly	266
Unsupported Operand type(s) for +: 'int' and 'bytes'	269
ord() Expected String of Length 1, but int Found	270
Unorderable Types: int() >= str()	273
Global Name 'reduce' Is not Defined.....	275
Lessons Learned.....	277
Chapter 16: Packaging Python Libraries.....	279
Things Distutils Can't Do for You	280
Directory Structure.....	281
Writing Your Setup Script.....	282
Classifying Your Package	284
Examples of Good Package Classifiers	284
Checking Your Setup Script for Errors	287
Creating a Source Distribution	287
Creating a Graphical Installer.....	289
Building Installable Packages for Other Operating Systems	290
Adding Your Software to the Python Package Index	291
The Many Possible Futures of Python Packaging.....	292
Further Reading Online.....	293
Appendix A: Porting Code to Python 3 with 2to3	295
print Statement.....	295
Unicode String Literals.....	296
unicode() Global Function	296
long Datatype.....	297
<> Comparison	297
has_key() Dictionary Method.....	298

Dictionary Methods that Return Lists	299
Renamed or Reorganized Modules.....	299
http.....	300
urllib	300
dbm	301
xmlrpc.....	302
Other Modules	302
Relative Imports Within a Package.....	304
next() Iterator Method.....	305
filter() Global Function.....	306
map() Global Function	306
reduce() Global Function	307
apply() Global Function	308
intern() Global Function	308
exec Statement.....	309
execfile Statement	309
repr Literals (Backticks)	310
try...except Statement.....	310
raise Statement	312
throw Method on Generators.....	312
xrange() Global Function	313
raw_input() and input() Global Functions.....	314
func_* Function Attributes.....	314
xreadlines() I/O Method	315
lambda Functions that Take a Tuple Instead of Multiple Parameters.....	316
Special Method Attributes.....	317
__nonzero__ Special Method	317
Octal Literals	318
sys.maxint	318
callable() Global Function.....	319
zip() Global Function.....	319
StandardError Exception	319
types Module Constants	320
isinstance() Global Function.....	321
basestring Datatype.....	321
itertools Module.....	322
sys.exc_type, sys.exc_value, sys.exc_traceback	322
List Comprehensions Over Tuples.....	323
os.getcwd() Function.....	323
Metaclasses	323
Matters of Style	324
set() Literals (Explicit).....	324

buffer() Global Function (Explicit)	324
Whitespace Around Commas (Explicit)	325
Common Idioms (Explicit)	325
■ Appendix B: Special Method Names	327
Basics	327
Classes that Act Like Iterators	328
Computed Attributes	328
Classes that Act Like Functions.....	331
Classes that Act Like Sequences.....	332
Classes that Act Like Dictionaries.....	334
Classes that Act Like Numbers	335
Classes that Can Be Compared	339
Classes that Can Be Serialized	340
Classes that Can Be Used in a “with” Block	340
Really Esoteric Stuff.....	342
Further Reading Online.....	343
■ Appendix C: Where to Go From Here	345
■ Index	347

Foreword

Seven years ago, I'd have looked at you incredulously and probably laughed if you had told me I would be sitting here today writing the foreword to a book, much less the foreword to a programming book. Yet here I am.

Seven years ago, I was just a test engineer with some scripting skills and a systems administration background. I had little programming experience and even less passion for it.

One day, a soon-to-be-coworker of mine mentioned this “new” scripting language called Python. He said it was easy to learn and might add to my skill set. I was wary because programmers seemed to be so separated from my “real world” of tests and systems and users. But his description also made me curious, so I visited the nearest bookstore and bought the first book on the subject I found.

The book I bought was the original *Dive Into Python*, by Mark Pilgrim. I have to believe that I am not the only person who can say, without exaggeration, that Mark's book changed my life and career forever.

The combination of Mark's book, his passion for Python, his presentation of the material, and even the Python (the language itself) fundamentally altered the way I thought. The combination drove me not just to read “yet another book about tech stuff”; it drove me to code, to represent my ideas in a completely new and exciting way. Mark's passion for the language inspired me with a newfound passion.

Now, seven years later, I'm a contributor to the Python standard library—an active community member—and I teach the language to as many people as I can. I use it in my free time, I use it at my job, and I contribute to it in between my daughter's naps. *Dive Into Python*—and Python itself—changed me.

Python is neither the prettiest nor most flexible language out there. But it is clean, simple, and powerful. Its elegance lies in its simplicity and its practicality. Its flexibility enables you (or anyone) to get something—anything—done simply by “keeping out of your way.”

I've said for some time the beauty of Python is that it scales “up.” It is useful for someone who wants only to do some math or write a simple script. And it is equally useful for programmers who want to create large-scale systems, web frameworks, and multimillion dollar video-sharing sites.

Python has not been without its warts, though. Building a language is, at least in my mind, much like learning to program. It's an evolutionary process where you constantly have to question the decisions you've made and be willing to correct those decisions.

Python 3 admits to some of those mistakes with its new fixes, removing some of the old warts, while also possibly introducing some new ones. Python 3 shows a self-awareness and willingness to evolve in much-needed ways you don't see in a lot of things.

Python 3 does not redefine, fundamentally alter, or suddenly invalidate all the Python you knew before. Rather, it takes something that is time-proven and battle-worn and improves on it in rational, practical ways.

Python 3 doesn't represent the end of the evolution of the language. New features, syntax, and libraries continue to be added; and it will probably be added, tweaked, and removed for as long as Python carries on.

Python 3 is simply a cleaner, more evolved platform for you, the reader, to get things done with.

Like Python 3, *Dive Into Python 3* represents the evolution of something that was already very good becoming something even better. Mark's passion, wit, and engaging style are still there; and the material has been expanded, improved, and updated. But like Python 3 itself, version 3 of this series fundamentally remains the thing that originally gave me such a passion for programming.

Python's simplicity is infectious. The passion of its community, not to mention the passion with which the language is created and maintained, remains astounding.

I hope Mark's passion, and Python itself, inspires you as it did me seven years ago. I hope you find Python, and Python 3, to be as practical and powerful as the hundreds of thousands of programmers and companies that use it across the world.

Jesse Noller
Python Developer

About the Author



■ By day, Mark Pilgrim is a developer advocate for open source and open standards. By night, he is a husband and father who lives in North Carolina with his wife, his two sons, and his big, slobbery dog. He spends his copious free time sunbathing, skydiving, and making up autobiographical information.

About the Technical Reviewer



■ Simon Willison is a speaker, writer, developer and all-around web technology enthusiast. Simon works for Guardian News and Media as a technical architect for both guardian.co.uk and the recently launched Guardian Developer Network.

Before joining the Guardian Simon worked as a consultant for clients that included the BBC, Automattic, and GCap Media. He is a past member of Yahoo!'s Technology Development team (his projects included the initial prototype of FireEagle, Yahoo!'s location broker API). Prior to Yahoo!, he worked at the *Lawrence Journal-World*, an award winning local newspaper in Kansas.

Simon is a co-creator of the Django web framework, and a passionate advocate for Open Source and standards-based development. He maintains a popular web development weblog at <http://simonwillison.net>.

Acknowledgments

The author would like to thank his wife for her never-ending support and encouragement, without which this book would still be an item on an ever-growing wish list.

Thank you to Raymond Hettinger for relicensing his `alphanumeric` solver so I could use it as the basis for Chapter 8.

Thank you to Jesse Noller for patiently explaining so many things to me at PyCon 2009, so that I could explain them to everyone else.

Finally, thank you to the many people who gave me feedback during the public writing process, especially Giulio Piancastelli, Florian Wollenschein, and all the good people of `python.reddit.com`.

Installing Python

Welcome to Python 3. Let's dive in. In this chapter, you'll install the version of Python 3 that's right for you.

Which Python Is Right for You?

The first thing you need to do with Python is install it. Or do you?

If you're using an account on a hosted server, your Internet service provider (ISP) might have already installed Python 3. If you're running Linux at home, you might already have Python 3, too. Most popular GNU/Linux distributions come with Python 2 in the default installation; a small but growing number of distributions also include Python 3. (As you'll see in this chapter, you can have more than one version of Python installed on your computer.) Mac OS X includes a command-line version of Python 2, but (as of this writing) it does not include Python 3. Microsoft Windows does not come with any version of Python. But don't despair! You can point and click your way through installing Python, regardless of which operating system you have.

The easiest way to check for Python 3 on your Linux or Mac OS X system is to get to a command line. On Linux, look in your Applications menu for a program called Terminal. (It might be in a submenu such as Accessories or System.) On Mac OS X, there is an application called `Terminal.app` in your `/Application/Utilities/` folder.

Once you're at a command-line prompt, just type `python3` (all lowercase, no spaces) and see what happens. On my home Linux system, Python 3 is already installed, and this command gets me into the Python interactive shell, as shown in Listing 0-1.

Listing 0-1. Python Interactive Shell

```
mark@atlantis:~$ python3
Python 3.0.1+ (r301:69556, Apr 15 2009, 17:25:52)
[GCC 4.3.3] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

Type `exit()` and press Enter to exit the Python interactive shell.

My web hosting provider also runs Linux and provides command-line access, but as Listing 0-2 shows, my server does not have Python 3 installed. (Boo!)

Listing 0-2. Python Interactive Shell

```
mark@manganese:~$ python3
bash: python3: command not found
```

So back to the question that started this section: “Which Python is right for you?” The answer is simple: whichever one runs on the computer you already have.

Installing on Microsoft Windows

Windows comes in two architectures these days: 32-bit and 64-bit. Of course, there are lots of different *versions* of Windows—XP, Vista, and Windows 7—but Python runs on all of them. The more important distinction is 32-bit versus 64-bit. If you have no idea what architecture you’re running, it’s probably 32-bit.

Visit <http://python.org/download/> and download the appropriate Python 3 Windows installer for your architecture. Your choices will look something like these:

- Python 3.1 Windows installer (Windows binary—does not include source)
- Python 3.1 Windows AMD64 installer (Windows AMD64 binary—does not include source)

I don’t want to include direct download links here because minor updates of Python happen all the time and I don’t want to be responsible for you missing important updates. You should always install the most recent version of Python 3.x unless you have some esoteric reason not to.

Once your download is complete, double-click the `.msi` file. Windows will display a security alert because you’re about to be running executable code. The official Python installer is digitally signed by the Python Software Foundation, the nonprofit corporation that oversees Python development. Don’t accept imitations!

Click the Run button to launch the Python 3 installer.



The first question the installer will ask you is whether you want to install Python 3 for all users or just for you. The default choice is “Install for all users,” which is the best choice unless you have a good reason to choose otherwise. (One reason why you might want to choose “Install just for me” is that you are installing Python on your company’s computer and you don’t have administrative rights on your Windows account. But then why are you installing Python without permission from your company’s Windows administrator? Don’t get me in trouble here!)

Click the Next button to accept your choice of installation type.



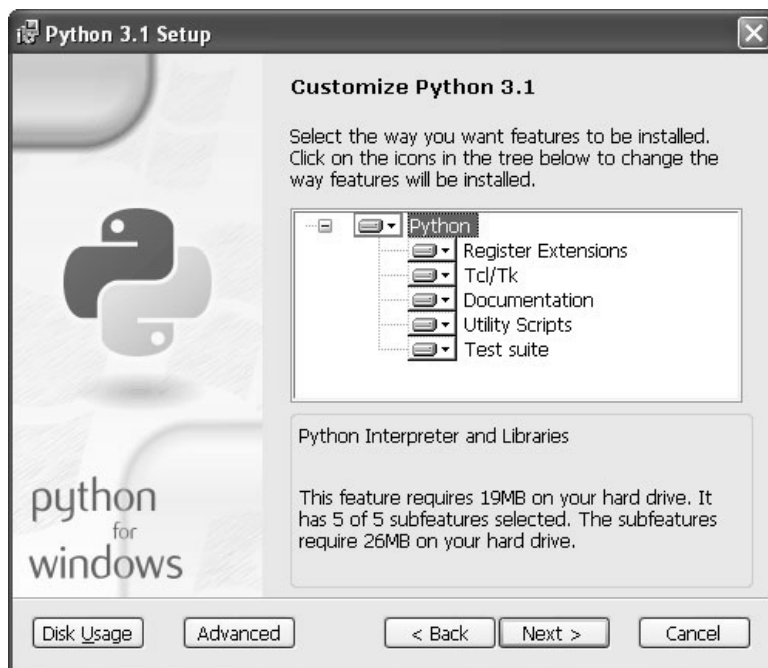
Next, the installer will prompt you to choose a destination directory. The default for all versions of Python 3.1.x is `C:\Python31\`, which should work well for most users unless you have a specific reason to change it. If you maintain a separate drive letter for installing applications, you can browse to it using the embedded controls or simply type the pathname in the box below. You are not limited to installing Python on the C: drive; you can install it on any drive, in any folder.

Click the Next button to accept your choice of destination directory.

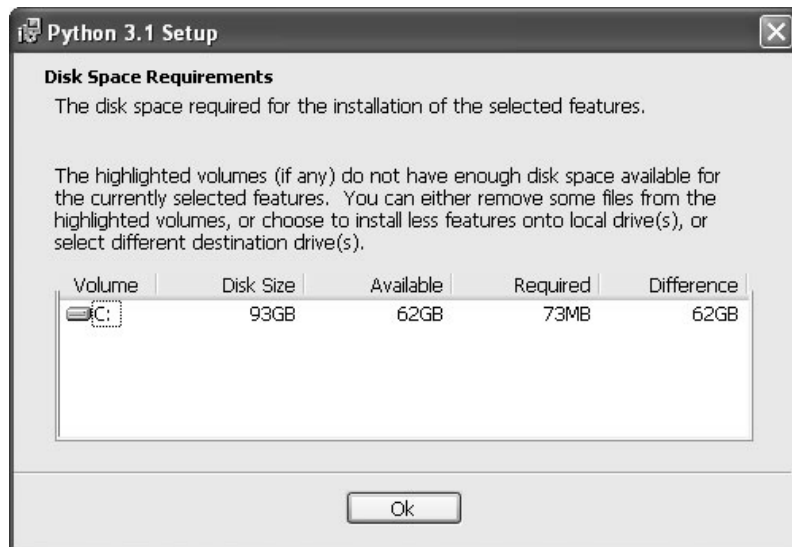


Although the next page looks complicated, it's not really difficult. Like many installers, you have the option not to install every single component of Python 3. If disk space is especially tight, you can exclude certain components.

- Register Extensions allows you to double-click Python scripts (.py files) and run them (recommended but not required). This option doesn't require any disk space, so there is little point in excluding it.
- Tcl/Tk is the graphics library used by the Python shell, which you will use throughout this book. I strongly recommend keeping this option.
- Documentation installs a help file that contains much of the information on <http://docs.python.org>. This option is recommended if you are on dialup or have limited Internet access.
- Utility Scripts includes the `2to3.py` script, which you'll learn about later in this book. It is required if you want to learn about migrating existing Python 2 code to Python 3. If you have no existing Python 2 code, you can skip this option.
- Test suite is a collection of scripts used to test the Python interpreter. You will not use it in this book, nor have I ever used it in the course of programming in Python. Completely optional.



If you don't know how much disk space you have, click the Disk Usage button. The installer will list your drive letters, compute how much space is available on each drive, and calculate how much would be left after installation.

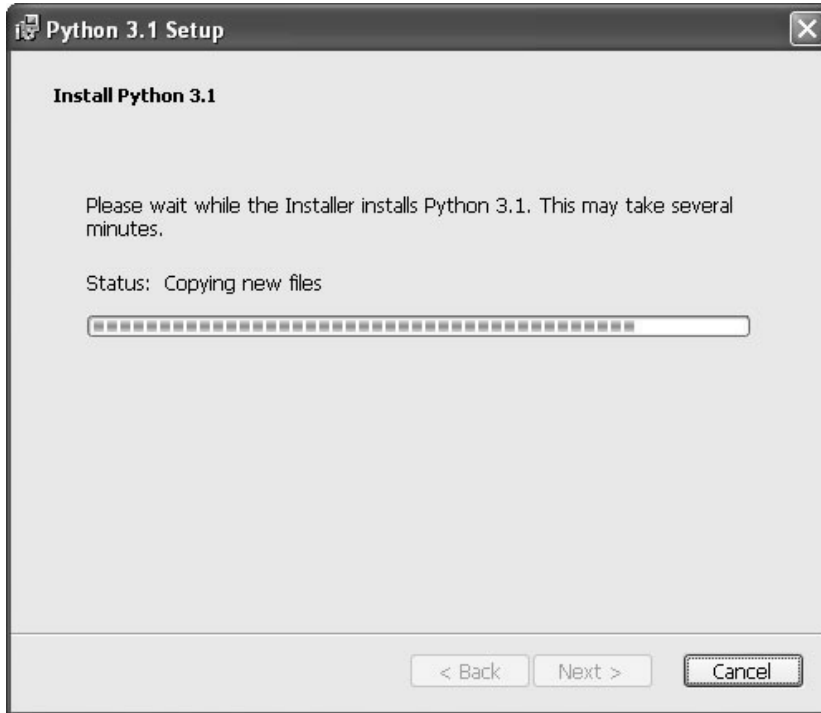


Click the OK button to return to the customization page. If you decide to exclude an option, select the drop-down button before the option and select "Entire feature will be unavailable". For example, excluding the Test suite will save you a whopping 7908KB of disk space.

Click the Next button to accept your choice of options.



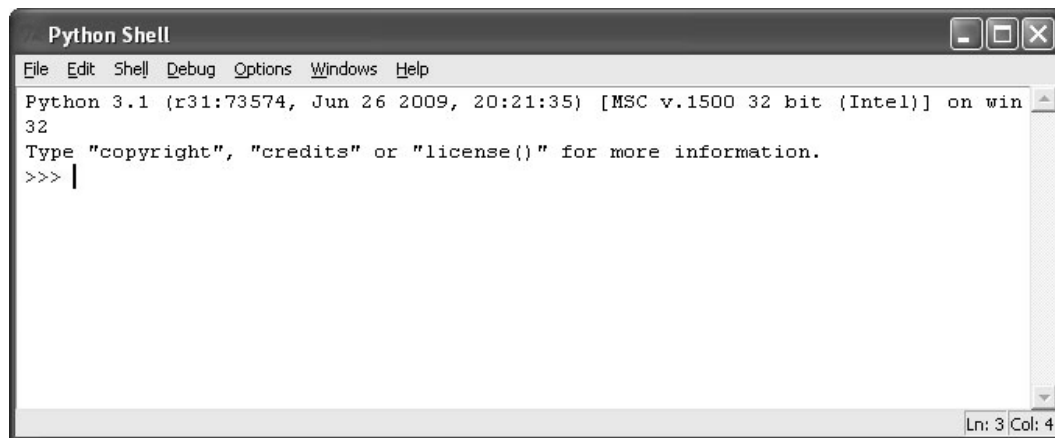
The installer will copy all the necessary files to your chosen destination directory. (This happens so quickly, I had to try it three times to even get a screenshot of it!)



Click the Finish button to exit the installer.



In your Start menu, there should be a new item called Python 3.1, in which you find a program called IDLE. Select this item to run the interactive Python shell.



```
Python Shell
File Edit Shell Debug Options Windows Help
Python 3.1 (r31:73574, Jun 26 2009, 20:21:35) [MSC v.1500 32 bit (Intel)] on win
32
Type "copyright", "credits" or "license()" for more information.
>>> |
Ln: 3 Col: 4
```

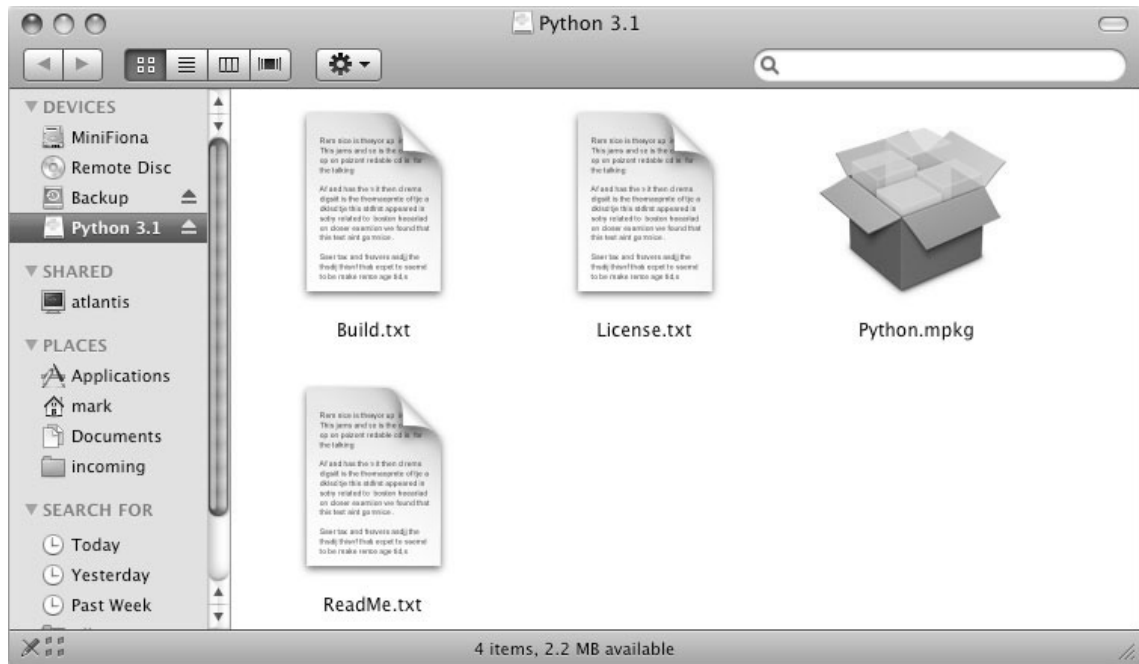
Installing on Mac OS X

All modern Macintosh computers use the Intel chip (as most Windows PCs do). Older Macs used PowerPC chips. You don't need to understand the difference because there's just one Mac Python installer for all Macs.

Visit <http://python.org/download/> and download the Mac installer. It will be called something like Python 3.1 Mac Installer Disk Image, although the version number might vary. Be sure to download version 3.x, not 2.x.

Your browser should automatically mount the disk image and open a Finder window to show you the contents. (If this doesn't happen, you'll need to find the disk image in your **Downloads** folder and double-click to mount it. It will be named something like `python-3.1.dmg`.) The disk image contains a number of text files (`Build.txt`, `License.txt`, `ReadMe.txt`) and the actual installer package, `Python.mpkg`.

Double-click the `Python.mpkg` installer package to launch the Mac Python installer.



The first page of the installer gives a brief description of Python itself. It then refers you to the `ReadMe.txt` file (which you didn't read, did you?) for more details.

Click the Continue button to move along.



The next page actually contains some important information: Python requires Mac OS X 10.3 or later. If you are still running Mac OS X 10.2, you should upgrade. Apple no longer provides security updates for your operating system, and your computer is probably at risk if you ever go online. Also, you can't run Python 3.