

Beginning Python

From Novice to Professional,
Second Edition



Magnus Lie Hetland

Beginning Python: From Novice to Professional, Second Edition

Copyright © 2008 by Magnus Lie Hetland

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 (paperback): 978-1-59059-982-2

ISBN-13 (electronic): 978-1-4302-0634-7

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

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: Frank Pohlmann

Technical Reviewers: Gregg Bolinger, Richard Taylor

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

Project Manager: Richard Dal Porto

Copy Editor: Marilyn Smith

Associate Production Director: Kari Brooks-Copony

Production Editor: Liz Berry

Compositor: Pat Christenson

Proofreader: April Eddy

Indexer: John Collin

Cover Designer: Kurt Krames

Manufacturing Director: Tom DeBolski

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 contact Apress directly at 2855 Telegraph Avenue, Suite 600, Berkeley, CA 94705. Phone 510-549-5930, fax 510-549-5939, 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.

The source code for this book is available to readers at <http://www.apress.com>.

Contents at a Glance

About the Author	xxiii
About the Technical Reviewer	xxv
Preface	xxvii
Introduction	xxix
■ CHAPTER 1 Instant Hacking: The Basics	1
■ CHAPTER 2 Lists and Tuples	31
■ CHAPTER 3 Working with Strings	53
■ CHAPTER 4 Dictionaries: When Indices Won't Do	69
■ CHAPTER 5 Conditionals, Loops, and Some Other Statements	83
■ CHAPTER 6 Abstraction	113
■ CHAPTER 7 More Abstraction	141
■ CHAPTER 8 Exceptions	161
■ CHAPTER 9 Magic Methods, Properties, and Iterators	175
■ CHAPTER 10 Batteries Included	209
■ CHAPTER 11 Files and Stuff	261
■ CHAPTER 12 Graphical User Interfaces	277
■ CHAPTER 13 Database Support	293
■ CHAPTER 14 Network Programming	305
■ CHAPTER 15 Python and the Web	321
■ CHAPTER 16 Testing, 1-2-3	349
■ CHAPTER 17 Extending Python	365
■ CHAPTER 18 Packaging Your Programs	383
■ CHAPTER 19 Playful Programming	393
■ CHAPTER 20 Project 1: Instant Markup	403
■ CHAPTER 21 Project 2: Painting a Pretty Picture	425
■ CHAPTER 22 Project 3: XML for All Occasions	435
■ CHAPTER 23 Project 4: In the News	453

CHAPTER 24	Project 5: A Virtual Tea Party	469
CHAPTER 25	Project 6: Remote Editing with CGI	489
CHAPTER 26	Project 7: Your Own Bulletin Board	499
CHAPTER 27	Project 8: File Sharing with XML-RPC	517
CHAPTER 28	Project 9: File Sharing II—Now with GUI!	537
CHAPTER 29	Project 10: Do-It-Yourself Arcade Game	547
APPENDIX A	The Short Version	569
APPENDIX B	Python Reference	579
APPENDIX C	Online Resources	595
APPENDIX D	Python 3.0	599
INDEX	607

Contents

About the Author	xxiii
About the Technical Reviewer	xxv
Preface	xxvii
Introduction	xxix
CHAPTER 1 Instant Hacking: The Basics	1
Installing Python	1
Windows	1
Linux and UNIX	3
Macintosh	5
Other Distributions	5
Keeping in Touch and Up-to-Date	7
The Interactive Interpreter	7
Algo . . . What?	9
Numbers and Expressions	9
Large Integers	11
Hexadecimals and Octals	12
Variables	13
Statements	13
Getting Input from the User	14
Functions	16
Modules	17
cmath and Complex Numbers	18
Back to the <code>__future__</code>	19
Saving and Executing Your Programs	19
Running Your Python Scripts from a Command Prompt	20
Making Your Scripts Behave Like Normal Programs	20
Comments	22

Strings	22
Single-Quoted Strings and Escaping Quotes	23
Concatenating Strings	24
String Representations, str and repr	24
input vs. raw_input	26
Long Strings, Raw Strings, and Unicode	26
A Quick Summary	29
New Functions in This Chapter	30
What Now?	30

CHAPTER 2	Lists and Tuples	31
	Sequence Overview	31
	Common Sequence Operations	32
	Indexing	32
	Slicing	34
	Adding Sequences	37
	Multiplication	37
	Membership	38
	Length, Minimum, and Maximum	40
	Lists: Python's Workhorse	40
	The list Function	40
	Basic List Operations	41
	List Methods	43
	Tuples: Immutable Sequences	49
	The tuple Function	50
	Basic Tuple Operations	50
	So What's the Point?	51
	A Quick Summary	51
	New Functions in This Chapter	52
	What Now?	52

CHAPTER 3	Working with Strings	53
	Basic String Operations	53
	String Formatting: The Short Version	53

String Formatting: The Long Version	56
Simple Conversion	57
Width and Precision	57
Signs, Alignment, and Zero-Padding	58
String Methods	60
find	60
join	61
lower	62
replace	63
split	63
strip	64
translate	64
A Quick Summary	66
New Functions in This Chapter	66
What Now?	67
 CHAPTER 4 Dictionaries: When Indices Won't Do	 69
Dictionary Uses	69
Creating and Using Dictionaries	70
The dict Function	71
Basic Dictionary Operations	71
String Formatting with Dictionaries	73
Dictionary Methods	74
A Quick Summary	81
New Functions in This Chapter	81
What Now?	81
 CHAPTER 5 Conditionals, Loops, and Some Other Statements	 83
More About print and import	83
Printing with Commas	83
Importing Something As Something Else	84
Assignment Magic	85
Sequence Unpacking	85
Chained Assignments	87
Augmented Assignments	87

Blocks: The Joy of Indentation	88
Conditions and Conditional Statements	88
So That's What Those Boolean Values Are For	89
Conditional Execution and the if Statement	90
else Clauses	90
elif Clauses	91
Nesting Blocks	91
More Complex Conditions	92
Assertions	97
Loops	97
while Loops	98
for Loops	99
Iterating Over Dictionaries	100
Some Iteration Utilities	100
Breaking Out of Loops	102
else Clauses in Loops	105
List Comprehension—Slightly Loopy	105
And Three for the Road	107
Nothing Happened!	107
Deleting with del	107
Executing and Evaluating Strings with exec and eval	108
A Quick Summary	111
New Functions in This Chapter	112
What Now?	112
 CHAPTER 6 Abstraction	 113
Laziness Is a Virtue	113
Abstraction and Structure	114
Creating Your Own Functions	115
Documenting Functions	116
Functions That Aren't Really Functions	117
The Magic of Parameters	117
Where Do the Values Come From?	118
Can I Change a Parameter?	118
Keyword Parameters and Defaults	123
Collecting Parameters	125
Reversing the Process	128
Parameter Practice	129
Scoping	131

Recursion	133
Two Classics: Factorial and Power	134
Another Classic: Binary Search	136
A Quick Summary	139
New Functions in This Chapter	140
What Now?	140
 CHAPTER 7 More Abstraction	 141
The Magic of Objects	141
Polymorphism	142
Encapsulation	145
Inheritance	147
Classes and Types	147
What Is a Class, Exactly?	147
Making Your Own Classes	148
Attributes, Functions, and Methods	150
Privacy Revisited	150
The Class Namespace	152
Specifying a Superclass	153
Investigating Inheritance	154
Multiple Superclasses	155
Interfaces and Introspection	156
Some Thoughts on Object-Oriented Design	157
A Quick Summary	158
New Functions in This Chapter	159
What Now?	159
 CHAPTER 8 Exceptions	 161
What Is an Exception?	161
Making Things Go Wrong . . . Your Way	161
The raise Statement	162
Custom Exception Classes	163
Catching Exceptions	163
Look, Ma, No Arguments!	164
More Than One except Clause	165
Catching Two Exceptions with One Block	166
Catching the Object	166
A Real Catchall	167

When All Is Well	168
And Finally	169
Exceptions and Functions	170
The Zen of Exceptions	171
A Quick Summary	173
New Functions in This Chapter	174
What Now?	174

■ CHAPTER 9 **Magic Methods, Properties, and Iterators** 175

Before We Begin	175
Constructors	176
Overriding Methods in General, and the Constructor in Particular	177
Calling the Unbound Superclass Constructor	179
Using the super Function	180
Item Access	182
The Basic Sequence and Mapping Protocol	182
Subclassing list, dict, and str	185
More Magic	187
Properties	187
The property Function	188
Static Methods and Class Methods	189
__getattr__, __setattr__, and Friends	191
Iterators	192
The Iterator Protocol	192
Making Sequences from Iterators	194
Generators	194
Making a Generator	195
A Recursive Generator	196
Generators in General	197
Generator Methods	198
Simulating Generators	199
The Eight Queens	200
Generators and Backtracking	200
The Problem	201
State Representation	202
Finding Conflicts	202
The Base Case	203
The Recursive Case	204
Wrapping It Up	205

A Quick Summary	206
New Functions in This Chapter	207
What Now?	207
CHAPTER 10 Batteries Included	209
Modules	209
Modules Are Programs	209
Modules Are Used to Define Things	211
Making Your Modules Available	214
Packages	217
Exploring Modules	218
What's in a Module?	218
Getting Help with help	219
Documentation	220
Use the Source	221
The Standard Library: A Few Favorites	221
sys	222
os	223
fileinput	225
Sets, Heaps, and Deques	227
time	232
random	234
shelve	238
re	242
Other Interesting Standard Modules	258
A Quick Summary	259
New Functions in This Chapter	260
What Now?	260
CHAPTER 11 Files and Stuff	261
Opening Files	261
File Modes	261
Buffering	263
The Basic File Methods	263
Reading and Writing	264
Piping Output	264
Reading and Writing Lines	266
Closing Files	267
Using the Basic File Methods	268

Iterating over File Contents	270
Doing It Byte by Byte	270
One Line at a Time	271
Reading Everything	271
Lazy Line Iteration with fileinput	272
File Iterators	272
A Quick Summary	274
New Functions in This Chapter	275
What Now?	275
 CHAPTER 12 Graphical User Interfaces	277
A Plethora of Platforms	277
Downloading and Installing wxPython	278
Building a Sample GUI Application	279
Getting Started	280
Windows and Components	281
Labels, Titles, and Positions	282
More Intelligent Layout	284
Event Handling	286
The Finished Program	286
But I'd Rather Use	288
Using Tkinter	289
Using Jython and Swing	290
Using Something Else	290
A Quick Summary	291
What Now?	291
 CHAPTER 13 Database Support	293
The Python Database API	294
Global Variables	294
Exceptions	295
Connections and Cursors	296
Types	297
SQLite and PySQLite	298
Getting Started	300
A Sample Database Application	300
A Quick Summary	303
New Functions in This Chapter	304
What Now?	304

CHAPTER 14	Network Programming	305
	A Handful of Networking Modules	305
	The socket Module	306
	The urllib and urllib2 Modules	308
	Other Modules	310
	SocketServer and Friends	310
	Multiple Connections	311
	Forking and Threading with SocketServer	313
	Asynchronous I/O with select and poll	313
	Twisted	316
	Downloading and Installing Twisted	317
	Writing a Twisted Server	317
	A Quick Summary	319
	New Functions in This Chapter	320
	What Now?	320
CHAPTER 15	Python and the Web	321
	Screen Scraping	321
	Tidy and XHTML Parsing	322
	Beautiful Soup	327
	Dynamic Web Pages with CGI	328
	Step 1. Preparing the Web Server	328
	Step 2. Adding the Pound Bang Line	329
	Step 3. Setting the File Permissions	329
	CGI Security Risks	330
	A Simple CGI Script	331
	Debugging with cgitb	331
	Using the cgi Module	333
	A Simple Form	334
	One Step Up: mod_python	336
	Installing mod_python	337
	CGI Handler	338
	PSP	339
	The Publisher	341
	Web Application Frameworks	343
	Web Services: Scraping Done Right	344
	RSS and Friends	345
	Remote Procedure Calls with XML-RPC	345
	SOAP	346

A Quick Summary.....	346
New Functions in This Chapter.....	347
What Now?.....	347
CHAPTER 16 Testing, 1-2-3.....	349
Test First, Code Later.....	349
Precise Requirement Specification	350
Planning for Change	351
The 1-2-3 (and 4) of Testing.....	352
Tools for Testing	352
doctest	353
unittest	355
Beyond Unit Tests.....	358
Source Code Checking with PyChecker and PyLint	359
Profiling.....	362
A Quick Summary.....	364
New Functions in This Chapter.....	364
What Now?.....	364
CHAPTER 17 Extending Python	365
The Best of Both Worlds	365
The Really Easy Way: Jython and IronPython	367
Writing C Extensions	369
A Swig of . . . SWIG.....	371
Hacking It on Your Own	375
A Quick Summary.....	380
New Functions in This Chapter.....	381
What Now?.....	381
CHAPTER 18 Packaging Your Programs	383
Distutils Basics	383
Wrapping Things Up.....	386
Building an Archive File	386
Creating a Windows Installer or an RPM Package.....	387
Compiling Extensions.....	388
Creating Executable Programs with py2exe	389

A Quick Summary	390
New Functions in This Chapter	391
What Now?	391
CHAPTER 19 Playful Programming	393
Why Playful?	393
The Jujitsu of Programming	393
Prototyping	394
Configuration	396
Extracting Constants	396
Configuration Files	396
Logging	399
If You Can't Be Bothered	400
If You Want to Learn More	400
A Quick Summary	401
What Now?	401
CHAPTER 20 Project 1: Instant Markup	403
What's the Problem?	403
Useful Tools	404
Preparations	405
First Implementation	406
Finding Blocks of Text	406
Adding Some Markup	407
Second Implementation	408
Handlers	409
A Handler Superclass	410
Rules	412
A Rule Superclass	413
Filters	413
The Parser	413
Constructing the Rules and Filters	415
Putting It All Together	418
Further Exploration	423
What Now?	424

CHAPTER 21	Project 2: Painting a Pretty Picture	425
	What's the Problem?	425
	Useful Tools	426
	Preparations	426
	First Implementation	427
	Drawing with ReportLab	427
	Constructing Some PolyLines	429
	Writing the Prototype	430
	Second Implementation	431
	Getting the Data	432
	Using the LinePlot Class	432
	Further Exploration	434
	What Now?	434
CHAPTER 22	Project 3: XML for All Occasions	435
	What's the Problem?	435
	Useful Tools	436
	Preparations	437
	First Implementation	438
	Creating a Simple Content Handler	439
	Creating HTML Pages	442
	Second Implementation	444
	A Dispatcher Mix-In Class	444
	Factoring Out the Header, Footer, and Default Handling	446
	Support for Directories	447
	The Event Handlers	448
	Further Exploration	451
	What Now?	452
CHAPTER 23	Project 4: In the News	453
	What's the Problem?	453
	Useful Tools	454
	Preparations	454
	First Implementation	455

Second Implementation.....	458
Further Exploration.....	467
What Now?.....	468
CHAPTER 24 Project 5: A Virtual Tea Party	469
What's the Problem?	469
Useful Tools	470
Preparations	470
First Implementation	471
The ChatServer Class	471
The ChatSession Class.....	473
Putting It Together.....	475
Second Implementation.....	477
Basic Command Interpretation	477
Rooms	478
Login and Logout Rooms.....	479
The Main Chat Room	479
The New Server	480
Further Exploration.....	486
What Now?.....	487
CHAPTER 25 Project 6: Remote Editing with CGI	489
What's the Problem?	489
Useful Tools	490
Preparations	490
First Implementation	490
Second Implementation.....	491
Creating the File Name Form	492
Writing the Editor Script.....	492
Writing the Save Script.....	494
Running the Editor.....	496
Further Exploration.....	497
What Now?.....	498

CHAPTER 26	Project 7: Your Own Bulletin Board	499
	What's the Problem?	499
	Useful Tools	500
	Preparations	500
	First Implementation	502
	Second Implementation	506
	Writing the Main Script	507
	Writing the View Script	508
	Writing the Edit Script	510
	Writing the Save Script	511
	Trying It Out	513
	Further Exploration	515
	What Now?	515
CHAPTER 27	Project 8: File Sharing with XML-RPC	517
	What's the Problem?	517
	Useful Tools	518
	Preparations	519
	First Implementation	519
	Implementing a Simple Node	520
	Trying Out the First Implementation	525
	Second Implementation	527
	Creating the Client Interface	527
	Raising Exceptions	528
	Validating File Names	529
	Trying Out the Second Implementation	534
	Further Exploration	534
	What Now?	535
CHAPTER 28	Project 9: File Sharing II—Now with GUI!	537
	What's the Problem?	537
	Useful Tools	537
	Preparations	538
	First Implementation	538
	Second Implementation	541
	Further Exploration	545
	What Now?	545

CHAPTER 29	Project 10: Do-It-Yourself Arcade Game	547
	What's the Problem?	547
	Useful Tools	548
	pygame	548
	pygame.locals	549
	pygame.display	549
	pygame.font	550
	pygame.sprite	550
	pygame.mouse	550
	pygame.event	550
	pygame.image	551
	Preparations	551
	First Implementation	551
	Second Implementation	556
	Further Exploration	567
	What Now?	567
APPENDIX A	The Short Version	569
	The Basics	569
	Functions	571
	Objects and Stuff	572
	Some Loose Ends	576
APPENDIX B	Python Reference	579
	Expressions	579
	Statements	589
	Simple Statements	589
	Compound Statements	592
APPENDIX C	Online Resources	595
	Python Distributions	595
	Python Documentation	596
	Useful Toolkits and Modules	596
	Newsgroups, Mailing Lists, and Blogs	597

■ APPENDIX D	Python 3.0	599
	Strings and I/O	599
	Strings, Bytes, and Encodings	599
	Console I/O	600
	New String Formatting	600
	Classes and Functions	601
	Function Annotation	601
	Abstract Base Classes	601
	Class Decorators and New Metaclass Syntax	601
	Keyword-Only Parameters	602
	Nonlocal Variables	602
	Iterables, Comprehensions, and Views	603
	Extended Iterable Unpacking	603
	Dictionary and Set Comprehension	603
	Dictionary Views	603
	Iterator Return Values	603
	Things That Have Gone	604
	Some Minor Issues	604
	The Standard Library	604
	Other Stuff	605
■ INDEX		607

About the Author



■ **MAGNUS LIE HETLAND** is an associate professor of algorithms at the Norwegian University of Science and Technology (NTNU). Even though he loves learning new programming languages—even quite obscure ones—Magnus has been a devoted Python fan and an active member of the Python community for many years, and is the author of the popular online tutorials “Instant Python” and “Instant Hacking.” His publications include the forerunner to this book, *Practical Python* (Apress, 2002), as well as several scientific papers. When he isn’t busy staring at a computer screen, he may be found reading (even while bicycling), acting (in a local theater group), or gaming (mostly role-playing games).

About the Technical Reviewer

■ **RICHARD TAYLOR** is a senior analyst at QinetiQ Ltd in the UK, where he specializes in open systems architectures for command and control systems. He has been developing in Python since about 1994, and has used Python to build many large-scale commercial and research applications. When not working, Richard indulges his keen interest in genealogy and open source software, and is a regular contributor to the GRAMPS (Genealogical Research and Analysis Management Programming System) project.

Preface

Here it is—a shiny new edition of *Beginning Python*. If you count its predecessor, *Practical Python*, this is actually the third edition, and a book I’ve been involved with for the better part of a decade. During this time, Python has seen many interesting changes, and I’ve done my best to update my introduction to the language. At the moment, Python is facing perhaps its most marked transition in a very long time: the introduction of version 3. As I write this, the final release isn’t out yet, but the features are clearly defined and working versions are available. One interesting challenge linked to this language revision is that it isn’t backward-compatible. In other words, it doesn’t simply add features that I could pick and choose from in my writing. It also changes the existing language, so that certain things that are true for Python 2.5 no longer hold.

Had it been clear that the entire Python community would instantly switch to the new version and update all its legacy code, this would hardly be a problem. Simply describe the new language! However, a lot of code written for older versions exists, and much will probably still be written, until version 3 is universally accepted as The Way To Go™.

So, how have I gotten myself out of this pickle? First of all, even though there are incompatible changes, *most* of the language remains the same. Therefore, if I wrote entirely about Python 2.5, it would be *mostly* correct for Python 3 (and even more so for its companion release, 2.6). As for the parts that will no longer be correct, I have been a bit conservative and assumed that full adoption of version 3 will take some time. I have based the book primarily on 2.5, and noted things that will change throughout the text. In addition, I’ve included Appendix D, which gives you an overview of the main changes. I think this will work out for most readers.

In writing this second edition, I have had a lot of help from several people. Just as with the previous two versions (the first edition, and, before it, *Practical Python*), Jason Gilmore got me started and played an important role in getting the project on the road. As it has moved along, Richard Dal Porto, Frank Pohlmann, and Dominic Shakeshaft have been instrumental in keeping it going. Richard Taylor has certainly played a crucial role in ensuring that the code is correct (and if it still isn’t, I’m the one to blame), and Marilyn Smith has done a great job tuning my writing. My thanks also go out to other Apress staff, including Liz Berry, Beth Christmas, Steve Anglin, and Tina Nielsen, as well as various readers who have provided errata and helpful suggestions, including Bob Helmbold and Waclaw Kusnierczyk. I am also, of course, still thankful to all those who helped in getting the first two incarnations of this book on the shelves.

Preface to the First Edition

A few years ago, Jason Gilmore approached me about writing a book for Apress. He had read my online Python tutorials and wanted me to write a book in a similar style. I was flattered, excited, and just a little nervous. The one thing that worried me the most was how much time it would take, and how much it would interfere with my studies (I was a Ph.D student at the time). It turned out to be quite an undertaking, and it took me a lot longer to finish than I had expected.

Luckily, it didn't interfere too much with my school work, and I managed to get my degree without any delays.

Last year, Jason contacted me again. Apress wanted an expanded and revised version of my book. Was I interested? At the time, I was busy settling into a new position as associate professor, while spending all my spare time portraying Peer Gynt, so again time became the major issue. Eventually (after things had settled down a bit, and I had a bit more time to spare), I agreed to do the book, and this (as I'm sure you've gathered) is the result. Most of the material is taken from the first version of the book, *Practical Python* (Apress, 2002). The existing material has been completely revised, based on recent changes in the Python language, and several new chapters have been added. Some of the old material has also been redistributed to accommodate the new structure. I've received a lot of positive feedback from readers about the first version. I hope I've been able to keep what people liked and to add more of the same.

Without the persistent help and encouragement from several people, this book would never have been written. My heartfelt thanks go out to all of them. In particular, I would like to thank the team that has worked directly with me in the process of writing the book: Jason Gilmore, for getting the project off the ground and steering it in the right direction; Beckie Stones, for keeping everything together; Jeremy Jones and Matt Moodie for their technical comments and insights; and Linda Marousek for being so patient with me. I'm also grateful to the rest of the team for making the process as smooth as it has been. But this book wouldn't be what it is without several people who worked with me on the previous version: I'd like to thank Jason Gilmore and Alex Martelli for their excellent technical editing (Jason on the entire book, and Alex on the first half) and for going above and beyond the call of duty in dispensing advice and suggestions; Erin Mulligan and Tory McLearn for holding my hand through the process and for nudging me along when that was needed; Nancy Rapoport for her help polishing my prose; and Grace Wong for providing answers when no one else could. Pete Shinnars gave me several helpful suggestions on the game in Project 10, for which I am very grateful. My morale has also been heavily boosted by several encouraging emails from satisfied readers—thanks! Finally, I would like to thank my family and friends, and my girlfriend Ranveig, for putting up with me while I was writing this book.

Introduction

A C program is like a fast dance on a newly waxed dance floor by people carrying razors.

—Waldi Ravens

C++: Hard to learn and built to stay that way.

—Anonymous

Java is, in many ways, C++- -.

—Michael Feldman

And now for something completely different . . .

—Monty Python's Flying Circus

I've started this introduction with a few quotes to set the tone for the book, which is rather informal. In the hope of making it an easy read, I've tried to approach the topic of Python programming with a healthy dose of humor, and true to the traditions of the Python community, much of this humor is related to Monty Python sketches. As a consequence, some of my examples may seem a bit silly; I hope you will bear with me. (And, yes, the name Python is derived from Monty Python, not from snakes belonging to the family *Pythonidae*.)

In this introduction, I give you a quick look at what Python is, why you should use it, who uses it, who this book's intended audience is, and how the book is organized.

So, what is Python, and why should you use it? To quote an official blurb (available from <http://python.org/doc/essays/blurb.html>), it is "an interpreted, object-oriented, high-level programming language with dynamic semantics." Many of these terms will become clear as you read this book, but the gist is that Python is a programming language that knows how to stay out of your way when you write your programs. It enables you to implement the functionality you want without any hassle, and lets you write programs that are clear and readable (much more so than programs in most other currently popular programming languages).

Even though Python might not be as fast as compiled languages such as C or C++, what you save in programming time will probably be worth using it, and in most programs, the speed difference won't be noticeable anyway. If you are a C programmer, you can easily implement the critical parts of your program in C at a later date, and have them interoperate with the Python parts. If you haven't done any programming before (and perhaps are a bit confused by my references to C and C++), Python's combination of simplicity and power makes it an ideal choice as a place to start.

So, who uses Python? Since Guido van Rossum created the language in the early 1990s, its following has grown steadily, and interest has increased markedly in the past few years. Python is used extensively for system administration tasks (it is, for example, a vital component of several Linux distributions), but it is also used to teach programming to complete beginners. The US National Aeronautics and Space Administration (NASA) uses Python both for development and as a scripting language in several of its systems. Industrial Light & Magic uses Python in its production of special effects for large-budget feature films. Yahoo! uses it (among other things) to manage its discussion groups. Google has used it to implement many components of its web crawler and search engine. Python is being used in such diverse areas as computer games and bioinformatics. Soon one might as well ask, “Who *isn’t* using Python?”

This book is for those of you who want to learn how to program in Python. It is intended to suit a wide audience, from neophyte programmer to advanced computer wiz. If you have never programmed before, you should start by reading Chapter 1 and continue until you find that things get too advanced for you (if, indeed, they do). Then you should start practicing and write some programs of your own. When the time is right, you can return to the book and proceed with the more intricate stuff.

If you already know how to program, some of the introductory material might not be new to you (although there will probably be some surprising details here and there). You could skim through the early chapters to get an idea of how Python works, or perhaps read through Appendix A, which is based on my online Python tutorial “Instant Python.” It will get you up to speed on the most important Python concepts. After getting the big picture, you could jump straight to Chapter 10 (which describes the Python standard libraries).

The last ten chapters present ten programming projects, which show off various capabilities of the Python language. These projects should be of interest to beginners and experts alike. Although some of the material in the later projects may be a bit difficult for an inexperienced programmer, following the projects in order (after reading the material in the first part of the book) should be possible.

The projects touch upon a wide range of topics, most of which will be very useful to you when writing programs of your own. You will learn how to do things that may seem completely out of reach to you at this point, such as creating a chat server, a peer-to-peer file sharing system, or a full-fledged graphical computer game. Although much of the material may seem hard at first glance, I think you will be surprised by how easy most of it really is. If you would like to download the source code, it’s available from the Source Code/Download section of the Apress web site (<http://www.apress.com>).

Well, that’s it. I always find long introductions boring myself, so I’ll let you continue with your Pythoneering, either in Chapter 1 or in Appendix A. Good luck, and happy hacking.



Instant Hacking: The Basics

It's time to start hacking.¹ In this chapter, you learn how to take control of your computer by speaking a language it understands: Python. Nothing here is particularly difficult, so if you know the basic principles of how your computer works, you should be able to follow the examples and try them out yourself. I'll go through the basics, starting with the excruciatingly simple, but because Python is such a powerful language, you'll soon be able to do pretty advanced things.

First, I show you how to get the software you need. Then I tell you a bit about algorithms and their main components. Throughout these sections, there are numerous small examples (most of them using only simple arithmetic) that you can try out in the Python interactive interpreter (covered in the section “The Interactive Interpreter” in this chapter). You learn about variables, functions, and modules, and after handling these topics, I show you how to write and run larger programs. Finally, I deal with strings, an important aspect of almost any Python program.

Installing Python

Before you can start programming, you need some new software. What follows is a short description of how to download and install Python. If you want to jump into the installation process without detailed guidance, you can simply visit <http://www.python.org/download> to get the most recent version of Python.

Windows

To install Python on a Windows machine, follow these steps:

1. Open a web browser and go to <http://www.python.org>.
2. Click the Download link.
3. You should see several links here, with names such as Python 2.5.x and Python 2.5.x Windows installer. Click the Windows installer link to download the installer file. (If you're running on an Itanium or AMD machine, you need to choose the appropriate installer.)

1. *Hacking* is not the same as *cracking*, which is a term describing computer crime. The two are often confused. Hacking basically means “having fun while programming.” For more information, see Eric Raymond's article “How to Become a Hacker” at <http://www.catb.org/~esr/faqs/hacker-howto.html>.

Note If you can't find the link mentioned in step 3, click the link with the highest version among those with names like Python 2.5.x. For Python 2.5, you could simply go to <http://www.python.org/2.5>. Follow the instructions for Windows users. This will entail downloading a file called `python-2.5.x.msi` (or something similar), where 2.5.x should be the version number of the newest release.

4. Store the Windows Installer file somewhere on your computer, such as `C:\download\python-2.5.x.msi`. (Just create a directory where you can find it later.)
5. Run the downloaded file by double-clicking it in Windows Explorer. This brings up the Python install wizard, which is really easy to use. Just accept the default settings, wait until the installation is finished, and you're ready to roll!

Assuming that the installation went well, you now have a new program in your Windows Start menu. Run the Python Integrated Development Environment (IDLE) by selecting **Start ► Programs ► Python² ► IDLE (Python GUI)**.

You should now see a window that looks like the one shown in Figure 1-1. If you feel a bit lost, simply select **Help ► IDLE Help** from the menu to get a simple description of the various menu items and basic usage. For more documentation on IDLE, check out <http://www.python.org/idle>. (Here you will also find more information on running IDLE on platforms other than Windows.) If you press F1, or select **Help ► Python Docs** from the menu, you will get the full Python documentation. (The document there of most use to you will probably be the Library Reference.) All the documentation is searchable.

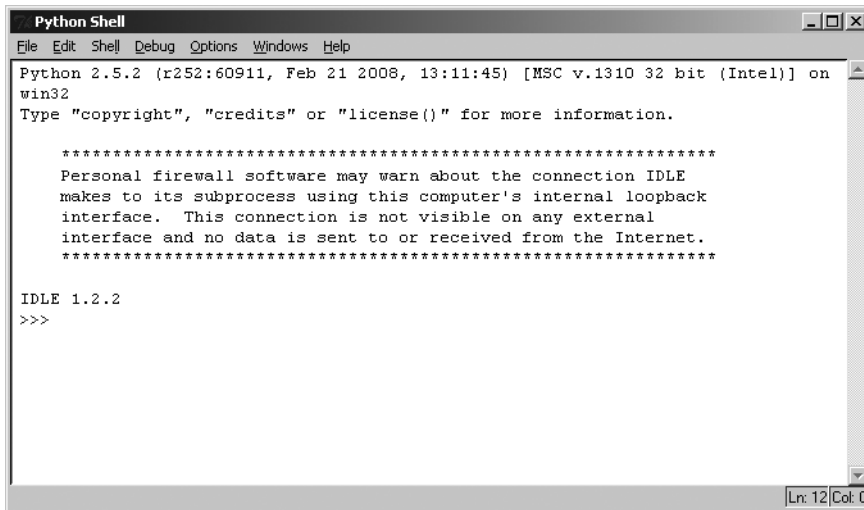


Figure 1-1. *The IDLE interactive Python shell*

2. This menu option will probably include your version number, as in Python 2.5.

Once you have the IDLE interactive Python shell running, you can continue with the section “The Interactive Interpreter,” later in this chapter.

WINDOWS INSTALLER

Python for Microsoft Windows is distributed as a Windows Installer file, and requires that your Windows version supports Windows Installer 2.0 (or later). If you don't have Windows Installer, it can be downloaded freely for Windows 95, 98, ME, NT 4.0, and 2000. Windows XP and later versions of Windows already have Windows Installer, and many older machines will, too. There are download instructions for the Installer on the Python download page.

Alternatively, you could go to the Microsoft download site, <http://www.microsoft.com/downloads>, and search for “Windows Installer” (or simply select it from the download menu). Choose the most recent version for your platform and follow the download and installation instructions.

If you're uncertain about whether you have Windows Installer, simply try executing step 5 of the previous installation instructions: double-click the MSI file. If you get the install wizard, everything is okay. See <http://www.python.org/2.5/msi.html> for advanced features of the Windows Installer related to Python installation.

Linux and UNIX

In most Linux and UNIX installations (including Mac OS X), a Python interpreter will already be present. You can check whether this is the case for you by running the `python` command at the prompt, as follows:

```
$ python
```

Running this command should start the interactive Python interpreter, with output similar to the following:

```
Python 2.5.1 (r251:54869, Apr 18 2007, 22:08:04)
[GCC 4.0.1 (Apple Computer, Inc. build 5367)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

Note To exit the interactive interpreter, use Ctrl-D (press the Ctrl key and while keeping that depressed, press D).

If there is no Python interpreter installed, you will probably get an error message similar to the following:

```
bash: python: command not found
```

In that case, you need to install Python yourself, as described in the following sections.

Using a Package Manager

Several package systems and installation mechanisms exist for Linux. If you're running a Linux system with some form of package manager, chances are you can get Python through it.

Note You will probably need to have administrator privileges (a root account) in order to install Python using a package manager in Linux.

For example, if you're running Debian Linux, you should be able to install Python with the following command:

```
$ apt-get install python
```

If you're running Gentoo Linux, you should be able to use Portage, like this:

```
$ emerge python
```

In both cases, `$` is, of course, the bash prompt.

Note Many other package managers out there have automatic download capabilities, including Yum, Synaptic (specific to Ubuntu Linux), and other Debian-style managers. You should probably be able to get recent versions of Python through these.

Compiling from Sources

If you don't have a package manager, or would rather not use it, you can compile Python yourself. This may be the method of choice if you are on a UNIX box but you don't have root access (installation privileges). This method is very flexible, and enables you to install Python wherever you want, including in your own home directory. To compile and install Python, follow these steps:

1. Go to the download page (refer to steps 1 and 2 in the instructions for installing Python on a Windows system).
2. Follow the instructions for downloading the sources.
3. Download the file with the extension `.tgz`. Store it in a temporary location. Assuming that you want to install Python in your home directory, you may want to put it in a directory such as `~/python`. Enter this directory (e.g., using `cd ~/python`).
4. Unpack the archive with the command `tar -xvzf Python-2.5.tgz` (where 2.5 is the version number of the downloaded source code). If your version of `tar` doesn't support the `z` option, you may want to uncompress the archive with `gunzip` first, and then use `tar -xvf` afterward. If there is something wrong with the archive, try downloading it again. Sometimes errors occur during download.