

Python[®] FOR DUMMIES[®]

by Stef Maruch and Aahz Maruch



WILEY

Wiley Publishing, Inc.

Python[®] FOR DUMMIES[®]

by Stef Maruch and Aahz Maruch



WILEY

Wiley Publishing, Inc.

Python® For Dummies®

Published by
Wiley Publishing, Inc.
111 River Street
Hoboken, NJ 07030-5774
www.wiley.com

Copyright © 2006 by Wiley Publishing, Inc., Indianapolis, Indiana

Published by Wiley Publishing, Inc., Indianapolis, Indiana

Published simultaneously in Canada

No part of this publication may be reproduced, stored in a retrieval system or transmitted in any form or by any means, electronic, mechanical, photocopying, recording, scanning or otherwise, except as permitted under Sections 107 or 108 of the 1976 United States Copyright Act, without either the prior written permission of the Publisher, or authorization through payment of the appropriate per-copy fee to the Copyright Clearance Center, 222 Rosewood Drive, Danvers, MA 01923, (978) 750-8400, fax (978) 646-8600. Requests to the Publisher for permission should be addressed to the Legal Department, Wiley Publishing, Inc., 10475 Crosspoint Blvd., Indianapolis, IN 46256, (317) 572-3447, fax (317) 572-4355, or online at <http://www.wiley.com/go/permissions>.

Trademarks: Wiley, the Wiley Publishing logo, For Dummies, the Dummies Man logo, A Reference for the Rest of Us!, The Dummies Way, Dummies Daily, The Fun and Easy Way, Dummies.com, and related trade dress are trademarks or registered trademarks of John Wiley & Sons, Inc. and/or its affiliates in the United States and other countries, and may not be used without written permission. All other trademarks are the property of their respective owners. Wiley Publishing, Inc., is not associated with any product or vendor mentioned in this book.

LIMIT OF LIABILITY/DISCLAIMER OF WARRANTY: THE PUBLISHER AND THE AUTHOR MAKE NO REPRESENTATIONS OR WARRANTIES WITH RESPECT TO THE ACCURACY OR COMPLETENESS OF THE CONTENTS OF THIS WORK AND SPECIFICALLY DISCLAIM ALL WARRANTIES, INCLUDING WITHOUT LIMITATION WARRANTIES OF FITNESS FOR A PARTICULAR PURPOSE. NO WARRANTY MAY BE CREATED OR EXTENDED BY SALES OR PROMOTIONAL MATERIALS. THE ADVICE AND STRATEGIES CONTAINED HEREIN MAY NOT BE SUITABLE FOR EVERY SITUATION. THIS WORK IS SOLD WITH THE UNDERSTANDING THAT THE PUBLISHER IS NOT ENGAGED IN RENDERING LEGAL, ACCOUNTING, OR OTHER PROFESSIONAL SERVICES. IF PROFESSIONAL ASSISTANCE IS REQUIRED, THE SERVICES OF A COMPETENT PROFESSIONAL PERSON SHOULD BE SOUGHT. NEITHER THE PUBLISHER NOR THE AUTHOR SHALL BE LIABLE FOR DAMAGES ARISING HEREFROM. THE FACT THAT AN ORGANIZATION OR WEBSITE IS REFERRED TO IN THIS WORK AS A CITATION AND/OR A POTENTIAL SOURCE OF FURTHER INFORMATION DOES NOT MEAN THAT THE AUTHOR OR THE PUBLISHER ENDORSES THE INFORMATION THE ORGANIZATION OR WEBSITE MAY PROVIDE OR RECOMMENDATIONS IT MAY MAKE. FURTHER, READERS SHOULD BE AWARE THAT INTERNET WEBSITES LISTED IN THIS WORK MAY HAVE CHANGED OR DISAPPEARED BETWEEN WHEN THIS WORK WAS WRITTEN AND WHEN IT IS READ. FULFILLMENT OF EACH COUPON OFFER IS THE SOLE RESPONSIBILITY OF THE OFFEROR.

For general information on our other products and services, please contact our Customer Care Department within the U.S. at 800-762-2974, outside the U.S. at 317-572-3993, or fax 317-572-4002.

For technical support, please visit www.wiley.com/techsupport.

Wiley also publishes its books in a variety of electronic formats. Some content that appears in print may not be available in electronic books.

Library of Congress Control Number: 2006924031

ISBN-13: 978-0-471-77864-6

ISBN-10: 0-471-77864-8

Manufactured in the United States of America

10 9 8 7 6 5 4 3 2 1

10/ST/QY/QW/IN



About the Authors

Stef Maruch got her hands on an original 128K Mac in 1984 and has been writing about computers ever since. She has over fifteen years' experience in instructional design, writing, and editing end-user computer manuals, including tutorials and user's guides for Apple Newton, HyperCard and HyperTalk, and DVD Studio Pro.

Aahz Maruch is a writer, trainer, and consultant who has been using Python for more than seven years. He has been using computers professionally for 20 years, and his background includes stints of high-end tech support, systems administration, and programming. Aahz is currently working as a programmer for a company with a Web-based application.

The authors can be reached at authors@pythonfood.com.

Dedication

Stef: I dedicate this book to my parents, Don and Betty Jones. You have always believed in me, even at times when I was quite improbable.

Aahz: I dedicate this book to the Python community. I hated programming until I learned Python (yes, for more than 20 years). I hope this book brings the joy of Python to many people.

This book is also dedicated to the Flying Spaghetti Monster.

Authors' Acknowledgments

Many people have helped us and supported us in writing this book. There are too many to mention all of them by name, so we want to start by thanking all the people we don't name here — all the family and friends and community who have sustained us.

Paula Anderson, Naomi Tilsen, Joyce Wermont, and Maggie Young provided much appreciated emotional support.

Thanks to our editors at Wiley:

- ✓ Acquisition editors Terri Varveris, Tiffany Franklin, and Kyle Looper, who shepherded these first-time *For Dummies* authors with just the right balance of patience and whip-cracking.
- ✓ Project editor Pat O'Brien, who provided invaluable assistance in e-mails that were often time-stamped with hours well past the time any less-dedicated person would have been in bed.
- ✓ Copy editor Andy Hollandbeck, who improved the book with his keen grasp of the beginner's mind, light-hearted prose, and Monty Python quotations.

High praise also to the production staff at Wiley, who are doing such great work with an extraordinarily complex and flexible book design.

Our technical editor, David Goodger, vastly improved our book with his edits and suggestions.

We feel fortunate to have our agents, David Fugate, who supported us expertly and patiently through the lengthy acquisition process, and Carole McClendon, who provided support at a critical juncture. Thanks also to the efficient staff at Waterside Productions.

Many people gave us advice and help while we were writing:

- ✓ Don and Betty Jones provided invaluable advice from the point of view of programming beginners
- ✓ Aahz's coworkers at Printra (<http://printra.net/>), especially Tony Lownds

The community of Python programmers on `comp.lang.python` and `tutor@python.org` not only helped Stef learn Python but also tirelessly work every day to promote Python and help make it accessible. This book wouldn't be possible without them.

The folks who maintain `www.python.org` and run the Python Software Foundation provide a critical service without which Python would be poorer.

Millions of people volunteer their time and efforts to make the Open Source movement a powerful force for good in the computer industry.

We also want to thank each other. Living together and writing a book is stressful, but we're glad we did this.

And, of course, none of it would be possible without Guido.

Publisher's Acknowledgments

We're proud of this book; please send us your comments through our online registration form located at www.dummies.com/register/.

Some of the people who helped bring this book to market include the following:

Acquisitions, Editorial, and Media Development

Project Editor: Pat O'Brien

Acquisitions Editor: Kyle Looper

Copy Editor: Andy Hollandbeck

Technical Editor: David Goodger

Editorial Manager: Kevin Kirschner

Media Development Specialists: Angela Denny, Kate Jenkins, Steven Kudirka, Kit Malone

Media Development Coordinator:

Laura Atkinson

Media Project Supervisor: Laura Moss

Media Development Manager: Laura VanWinkle

Editorial Assistant: Amanda Foxworth

Sr. Editorial Assistant: Cherie Case

Cartoons: Rich Tennant
(www.the5thwave.com)

Composition Services

Project Coordinator: Tera Knapp

Layout and Graphics: Claudia Bell, Denny Hager, Jake Mansfield, Barbara Moore, Barry Offringa, Heather Ryan

Proofreaders: Laura Albert, Susan Moritz, Techbooks

Indexer: Techbooks

Publishing and Editorial for Technology Dummies

Richard Swadley, Vice President and Executive Group Publisher

Andy Cummings, Vice President and Publisher

Mary Bednarek, Executive Acquisitions Director

Mary C. Corder, Editorial Director

Publishing for Consumer Dummies

Diane Graves Steele, Vice President and Publisher

Joyce Pepple, Acquisitions Director

Composition Services

Gerry Fahey, Vice President of Production Services

Debbie Stailey, Director of Composition Services

Contents at a Glance

<i>Introduction</i>	1
<i>Part I: Getting Started</i>	5
Chapter 1: Introducing Python	7
Chapter 2: Getting Your Hands on the Keyboard: Using Help, Interactive Mode, and IDLE.....	15
Chapter 3: Basic Elements and Syntax	39
Chapter 4: Grand Tour of the Python Language.....	59
Chapter 5: Working Like a Programmer.....	71
<i>Part II: Building Blocks</i>	87
Chapter 6: So This String Walks into a Bar.....	89
Chapter 7: Counting Your Way to Fun and Profit	111
Chapter 8: Processing Lists and Tuples	119
Chapter 9: Diving into Dictionaries	139
<i>Part III: Structures</i>	159
Chapter 10: Staying in Control.....	161
Chapter 11: Fun with Functions.....	183
Chapter 12: Building Applications with Modules and Packages	201
Chapter 13: Getting Classy	213
Chapter 14: Introducing New-Style Classes.....	235
Chapter 15: Feeling Exceptional	247
Chapter 16: Tackling Some Advanced Features.....	257
<i>Part IV: Libraries</i>	271
Chapter 17: Using Python's Primary Services	273
Chapter 18: Processing Text	301
Chapter 19: Digging into Disk Data.....	319
Chapter 20: Accessing the Internet.....	337
<i>Part V: The Part of Tens</i>	363
Chapter 21: Ten Critical Python Idioms.....	365
Chapter 22: Ten Great Resources	375

<i>Part VI: Appendixes</i>	381
Appendix A: Getting and Installing Python	383
Appendix B: Python Version Differences.....	391
<i>Index</i>	395

Table of Contents

<i>Introduction</i>	1
About This Book.....	1
Conventions Used in This Book	1
Foolish Assumptions	2
How This Book Is Organized.....	2
Part I: Getting Started	3
Part II: Building Blocks.....	3
Part III: Structures	3
Part IV: Libraries.....	3
Part V: The Part of Tens.....	3
Part VI: Appendixes.....	4
Icons Used in This Book.....	4
Where to Go from Here.....	4
<i>Part 1: Getting Started</i>	5
Chapter 1: Introducing Python	7
The Right Tool for the Job	7
Good uses of Python.....	7
Sometimes, Python isn't so hot	11
Cooking Up Programs.....	12
Training your assistant.....	13
Combining ingredients.....	13
Chapter 2: Getting Your Hands on the Keyboard: Using Help, Interactive Mode, and IDLE	15
Two Ways to Interact with Python.....	16
Going One-on-One in Interactive Mode	16
Starting interactive mode.....	17
Following the rules of engagement	18
Seeing information about a Python object.....	19
Seeing the result of the last expression.....	20
Manipulating strings and lists	21
Using interactive mode as a calculator	22
Working with built-in functions	23
Examining names.....	24
Writing multiline programs in interactive mode	25
Quitting interactive mode	26



- Getting Help27
 - Help in interactive mode27
 - Getting help in a Web browser28
- Using Scripts and Modules29
 - Running a script from the command line29
 - Importing a module in interactive mode.....30
 - Using Python’s standard modules in interactive mode.....32
- IDLE Musings34
 - Opening IDLE34
 - Typing statements and programs in the Python Shell34
 - Getting more help for IDLE.....35
 - Writing and editing code with IDLE’s text editor36
 - Briefly meet a few other IDLE commands36
 - Debugging in IDLE37

Chapter 3: Basic Elements and Syntax 39

- Making Names and Storing Values39
- Data Type Does Matter41
 - Numeric data.....42
 - Sequential data43
 - Dictionaries44
 - Sets44
 - Files45
 - Data types have methods.....45
- Operators Are Standing By47
 - Arithmetic operators47
 - Comparison operators.....47
 - Boolean operators.....48
 - Conditional operations49
 - Order, please!50
 - Special powers of the = symbol.....51
- If We May Comment51
 - It pays to be conventional.....51
 - Documenting your program.....52
- Oopsies! Understanding Error Messages53
- Deciphering Code Blocks53
 - Code block syntax54
 - Basic code blocks: Control structures and loops54
 - Code blocks that create a namespace56

Chapter 4: Grand Tour of the Python Language 59

- The spider.py Program59
- Examining a Python Program61
 - Setting up the structure.....62
 - Initializing the spider62
 - Running the spider63

Using Building Blocks65
 Function and method tidbits65
 Looping around66
 Collections of data67
 Naming names67
 Managing strings68
 Handling errors.....69

Chapter 5: Working Like a Programmer 71

The Three Ds71
 Documenting.....72
 Designing72
 Debugging.....73
 Maintaining Your Programs76
 Good Program Design Practices.....79
 Naming names80
 Following conventions80
 Don't forget to comment!.....81
 Debugging Strategies82
 Built-in functions82
 Print statements and traceback logs83
 Comments84
 Using a debugger84

Part II: Building Blocks87

Chapter 6: So This String Walks into a Bar 89

Stringing Them Along89
 Just the quotes, ma'am.....90
 Ways to escape91
 Raw strings.....92
 Being wordy92
 How a string looks inside Python.....93
 "Please repeat": String operators93
 A few more methods for working with strings96
 Cat's Cradle: Indexing and Slicing98
 Basic syntax98
 Figuring out the tricks99
 Changing the immutable string101
 Interpolating Between the Lines101
 Using the interpolation operator101
 A formatting example using string methods.....105
 Unraveling Unicode.....105
 Creating a Unicode string.....106
 A twisty maze of codes107
 Encoding, decoding, and other Unicode methods.....108

Chapter 7: Counting Your Way to Fun and Profit	111
Integrating Integers	111
Why Python has two kinds of integers	111
Avoiding unexpected results with integer division	112
Floating Along	113
Automatic conversion	113
Formatting floats	113
Size limits on floats	114
Imagining Complex Numbers	114
Using Math Modules	115
Turning Python into a Calculator with decimal	116
Representing numbers by using the decimal module	116
Viewing and changing parameters	117
Chapter 8: Processing Lists and Tuples	119
Introducing Lists and Tuples	119
What a list is	120
What a tuple is	120
Manipulating Sequence Objects	122
Comparing sequence objects	122
Operating on sequence objects	123
Listcraft: Methods, Indexes, and Slices	124
Functions that work with or create lists	124
Methods of lists	124
List indexing and slicing	126
Steering Clear of List Gotcha's	130
Simultaneous test/add/delete	131
List references that unexpectedly change	131
Disappearing lists	132
That tricky asterisk	132
Performance problems	133
Building Lists, Stacks, and Queues	133
Building lists incrementally	134
Stacking and queuing with lists	135
Taking Tuples in Hand	136
Converting another object to a tuple	136
Tuple packing	137
Tuple unpacking	137
Using a tuple to swap values	138
Chapter 9: Diving into Dictionaries	139
Defining the Dictionary	140
Creating a dictionary	140
Order in the dict	141

Doodling Around with Dicts142
 Popular dict operations142
 Finding out about dict methods144
 Building Dictionaries149
 Converting other data types into a dict149
 Updating a dictionary151
 When Only a Dict Will Do151
 Storing and retrieving values151
 Using a dict as a cache152
 Dealing with duplicate keys153
 Setting Them Up154
 What sets are and aren't154
 Membership testing with sets154
 Finding set elements155
 Immutable or frozen sets156

Part III: Structures 159

Chapter 10: Staying in Control161

Things to Know about Control Structures161
 All about Conditions and Comparisons162
 The value of truth162
 Boolean operators164
 Comparison operators165
 Feeling Iffy167
 Writing an if statement167
 Adding another condition to an if block168
 Adding an else statement to an if block169
 Combining tests170
 Staying in the Loop170
 For a Good Time171
 Whiling Away173
 Choosing Your Loop174
 What for's for175
 Why to while176
 Loopy Statements and Functions177
 Useful looping statements177
 Loopy functions179

Chapter 11: Fun with Functions 183

I Love Chunky Code183
 Calling a function184
 Defining a Function185
 Giving another name to a function186
 Returning values from a function188

Argument Clinic: Passing Data	188
Introducing parameters and arguments	189
Specifying arguments when you call a function	189
Specifying arguments with keywords and default values	190
Avoiding the quirks of default values	191
Specifying a function with an arbitrary number of arguments	194
Unpacking arguments	195
What's in a Namespace.....	196
Discovering where Python looks for names	196
Understanding function namespaces	196
Think globally, act locally.....	199
Sorting out module namespaces	200

Chapter 12: Building Applications with Modules and Packages . . .201

Modular Living: Storing Your Code in Files	201
Importing a module or its contents	202
Giving a local name to a module	204
Rules for writing and naming modules	205
Module, module, where is the module?.....	205
Finding what's in standard modules	206
Wrapping It Up in a Package	206
The purposes of packages	207
Requirements for packages.....	208
Importing items from packages	209

Chapter 13: Getting Classy213

Alley-OOP! Some Object-Oriented Programming Concepts.....	214
Objects.....	214
Inheriting, overriding, and extending	215
Polymorphism and duck-typing	215
Now Class, for Instance	216
Classes, modules, and functions	216
A class is like a template	216
An instance is a copy made from the template	218
All about class and instance attributes	218
Making and Calling Classes.....	219
Creating a class.....	219
Creating an instance.....	222
Calling a method via an instance.....	222
Getting Inside the Factory: How Class and Instance Namespaces Interact.....	224
Changing the values of class and instance attributes	224
Adding an attribute to an instance	225
Class and Instance Conventions	226

Inheriting the Farm: Overriding and Extending Classes	227
Creating a subclass	227
Overriding superclass methods	227
Extending superclass methods.....	228
Using multiple inheritance	229
Namespace searching in classes and superclasses	230
Operator interception and overloading	232
When to Go to Class.....	234
Chapter 14: Introducing New-Style Classes	235
An Object's Object: Intro to New-Style Classes	235
Everything comes from object	237
Methods of inheritance	237
New Improved Class Features	238
Getcher attributes!.....	238
Hey, baby, what's your type?	239
Calling the right superclass method.....	239
That's my property!.....	240
When only class matters	241
Cutting through the static about methods	243
Don't use the slots machine.....	244
Island of Dr. MRO.....	244
Exploding Your Head with Metaclasses	245
Roles.....	245
Applications	246
Chapter 15: Feeling Exceptional	247
All about Special Handling	247
Trying Things Out	248
Using try/except statements.....	249
Using try/finally statements.....	252
try/except/else/finally: Together at last	253
Raising Your Code to New Levels.....	254
Making Your Program Exceptional.....	255
Writing a base class for your exceptions	255
Developing an exception hierarchy	256
Chapter 16: Tackling Some Advanced Features	257
What's That Idiom?	257
What to Do Next: Iterators and Generators	258
The itertools library.....	258
Generators: yield for faster processing	260

Expression and Comprehension: Listcomps and Genexps.....	261
List comprehensions.....	262
Generator expressions.....	263
With What, Your Bare Hands? (The Power of ‘with’ Statements).....	264
Making Exceptions for Yourself.....	265
Under One Condition.....	266
Decorating Your Code.....	267
Focusing on Functions.....	268
Mary had a little lambda	269
Mapping it out.....	269
Applying filters	270
Reductionism	270

Part IV: Libraries271

Chapter 17: Using Python’s Primary Services273

Python: Batteries Included	273
You Get All This! — The <code>__builtin__</code> Module	274
Seeing what’s inside Python objects.....	275
Reading and writing files	275
Working with attributes.....	278
Finding largest and smallest items.....	279
Getting input from users.....	279
Finding an object’s type.....	280
Reloading a module.....	280
Evaluating a string.....	281
But Wait, There’s More — The <code>sys</code> Module.....	281
Solving OS Incompatibility — The <code>os</code> and <code>subprocess</code> Modules	282
Working with the <code>os</code> module	283
Subprocessing.....	287
Staying on Time with the <code>datetime</code> and <code>time</code> Modules.....	288
Using the <code>datetime</code> module	289
Taking your time.....	291
Checking with the <code>doctest</code> Module.....	293
Introducing the <code>doctest</code> module.....	293
Adding interactive code to a module’s docstrings	294
Adding <code>doctest</code> code to a module	294
Testing a module	295
Keeping Track with the <code>logging</code> Module.....	296
Getting started with basic logging	296
Changing the configuration of a logger	297
Designing your own logging system	298
Seeing a few logging functions.....	299

Chapter 18: Processing Text	301
A Million Ways to re, You Know That There Are	301
Writing a basic regex	302
Setting up a basic regex search	302
Regular expression characters and codes	303
Using the regular expression module	307
Strings Disguised as Files	313
Creating a StringIO object	314
StringIO special methods	314
Paragraph Dumplings: Filling and Wrapping Text	315
Removing indentation from strings	315
Wrapping text by splitting it up	315
Wrapping text by adding newline characters	316
Creating a TextWrapper object	317
Chapter 19: Digging into Disk Data	319
Shell Game: Copying and Moving Files	319
Zipping and Unzipping	321
zipfile	321
gzip	325
Sussing Out SQL Databases	326
Installing SQLite and sqlite3	327
Setting up a SQLite database	327
Working with a SQLite database	327
Pickling Your Data (And Relishing the Outcome)	330
Pickling an object	331
Unpickling an object	333
Using shelve with DBM-style databases	333
Storing pickles on a shelf	334
Creating a shelve object	334
Properties of a shelve object	336
Chapter 20: Accessing the Internet	337
Downloading Web Data	337
Opening a URL	337
Finding information about a URL	339
Processing special characters in a URL	339
Submitting form data	340
Taming the Wild URL	341
Getting Hip with Hypertext	342
Of parsers, formatters, and writers	342
Setting up a read-and-output process	343
Outputting the links of a Web page	343
Getting help for messy HTML	344

The Great XML.....	344
The ElementTree XML implementation.....	344
Other useful XML modules.....	348
MIME-ing Success: Managing E-Mail Messages	349
Representing an e-mail message in Python.....	349
Creating e-mail and MIME objects.....	349
Generating MIME documents from message structures	352
Reading e-mail messages.....	353
Using e-mail utilities	354
Simply SMTP	355
CGI: Gateway to the Web.....	356
Setting up CGI output in Python.....	356
Reading data from Web input	357
Setting up and installing a CGI script.....	358
Debugging CGI scripts	359

***Part V: The Part of Tens*363**

Chapter 21: Ten Critical Python Idioms365

Collecting Globs and Globs of Files	365
Rolling Dice and Shuffling Cards	366
Making a saving throw	366
Playing the dealer.....	366
Uniquely Ordered Lists	367
Reversing Your Way to Success.....	368
Exceptional Type-Testing.....	368
Classes Just for Data.....	369
Getting Close Enough with difflib	370
DSU! DSU! Rah rah DSU!.....	370
Simplifying Choices Using Dicts.....	372
Singles Going Steady.....	372
Using a singleton object	372
Calling in the Borg Pattern	373

Chapter 22: Ten Great Resources375

The Mothership: www.python.org	375
We're glad you asked that: Python FAQs.....	375
Official documentation	376
tutor@python.org and help@python.org	377
The comp.lang.python Newsgroup.....	377
Cheese Shop: Online Collection of Python Modules	377
Random Access Reference at wiki.python.org	378
The Python Cookbook Web Site.....	379

The Latest News	379
Dr. Dobbs' Python-URL	379
Daily Python URL.....	379
comp.lang.python.announce	380
Being a PUG-nosed PIGgie: Local User Groups.....	380
<i>Part VI: Appendixes</i>	381
Appendix A: Getting and Installing Python	383
Operating Systems	383
Windows	383
Mac OS X.....	386
UNIX and Linux	388
Using Embedded Python.....	389
Appendix B: Python Version Differences	391
Python 2.5	391
Python 2.4	392
Python 2.3	392
Python 2.2	393
Python 2.1	394
Python 2.0	394
<i>Index</i>	395

Introduction

Congratulations! You're ready to discover the easiest-to-read powerful programming language — or maybe the most powerful, easy-to-read programming language. That's Python, of course.

With *Python For Dummies*, you can ferret out just a little or a lot. And with Python, you can write a little program that picks a random quote from a file, or you can write a set of programs that runs a complex business.

This book is for you whether you're a student, you're a hobbyist, you need to understand more about what your programmer co-workers are talking about, or you're taking the first steps on a new career path.

Python For Dummies gives you everything you need to get to an advanced-beginner level of Python programming. And it points you to other resources so you can take your Python programming skills even further.

About This Book

Python For Dummies is a reference book, which means you can read it in any order, and you don't have to read every chapter or section. However, to some extent, later chapters about more complex Python features rely on information introduced in earlier chapters. So if you don't understand something you see in a later chapter, go to Chapter 3, or go to the chapter on that feature to find out more. You can also look in the index to find a term or feature you want to know more about.

Conventions Used in This Book

This book contains Python code examples. All code examples are in `monospaced font` so they are easy to recognize. Anything that you need to type is also indicated in `monospaced font` so you know exactly which commas should be typed and which commas are part of the surrounding sentence.

Python interactive mode examples include this prompt: `>>>`. If you don't see the prompt, you can assume the code was written in a text editor.

Foolish Assumptions

We make the following assumptions about readers of this book:

✔ **You know how to use your computer and its operating system.**

It's helpful but not necessary to know how to set environment variables on your computer. It's also helpful to have a Web browser with access to the Internet.

✔ **You have and know how to use a text editor that can produce plain ASCII text or files that end with the `.txt` extension.**

If you don't have a text editor that can do this, we include instructions for setting up Python's IDLE programming environment to work with the examples in this book.

✔ **You have had a minimal amount of exposure to programming.**

We really do mean *minimal*. If you had a programming class in high school, or wrote a few BASIC programs at one time, or even if you have used HTML tags, that counts.

If you have absolutely no experience with programming, you can still find out plenty from this book, but we recommend that you also look at a book or Web tutorial designed to introduce programming to beginners. You'll benefit from the extended explanations of some concepts that we don't have the space to discuss in detail here.

✔ **You might have done some programming in another language.**

Programming knowledge is not required for this book, but people who have programmed in other languages have their own sets of issues when transitioning to Python, and we provide some material for such people.

✔ **You know little to nothing about Python.**

If you know Python, this book will still be helpful as a reference or a source of tips and tricks you may not be aware of.

How This Book Is Organized

This book gives you an overview of Python; the lowdown about all of its major parts, structures, and libraries; and a glimpse into some more advanced features. You also find out where to go to discover more.

Part I: Getting Started

In this part, we introduce Python and situate it among the myriad other programming languages available. Python is good for some things and not for others; you find out which is which. We provide a hands-on introduction to some of Python's abilities, using its helpful interactive mode and its IDLE programming environment. We briefly describe each of Python's basic building blocks and show how all these blocks come together by dissecting a working program. We sketch an overview of how professional programmers design programs and debug code and show you how to put these practices to work to make your own programming life easier.

Part II: Building Blocks

Python has six basic data types and many ways to work with each of them. In this part, we describe how to work with strings (chunks of text), numbers, lists and tuples (both of which store multiple data elements), dictionaries (which associate one element with another), and sets (which always contain unique elements, never duplicates).

Part III: Structures

Python code usually comes in chunks, both small and big, and each chunk does a particular thing. This part also includes a brief introduction to some advanced features and the new features of Python 2.5.

Part IV: Libraries

Python comes with everything you need to write a very powerful program, and other people have already solved lots of programming conundrums for you. Its libraries include primary services such as communication with the operating system, text processing tools, various ways of reading and writing information to disk, and Internet access methods.

Part V: The Part of Tens

All *For Dummies* books include The Part of Tens. In this part, we give you ten useful but not-so-obvious programming idioms and ten resources where you can find out more about Python.

Part VI: Appendixes

Here you find instructions on how to install Python and its documentation, as well as a list of new features introduced with each new version of Python since 2.0.

Icons Used in This Book



Icons appear throughout the book to indicate special material. Here's what they mean:

A Tip explains how to do something a little bit more easily and efficiently.



A Warning gives you a heads-up about tricky stuff or common mistakes that might cause data loss or some other sort of headache. It's best to read Warnings to make sure a tricky feature doesn't "getcha."



A Technical Stuff icon flags text that's of interest to readers who like to know about the inner workings or history of a subject. You don't need to read Technical Stuff material. After you've internalized a little about a subject, reading this text might help you understand it from a different angle.



Remember icons highlight important concepts or pieces of information to keep in mind.

Where to Go from Here

If you want an overview of Python's history and what it can do, go to Chapter 1. If you're new to Python and want to start working with it right away, go to Chapter 2. If you want a brief overview of all of Python's building blocks, go to Chapter 3. If you know some Python and you want a refresher or additional info on some of its tools, go to the specific chapters you're interested in.

Part I

Getting Started

The 5th Wave

By Rich Tennant



“The engineers lived on Jolt and cheese sticks putting this product together, but if you wanted to just use ‘cola and cheese sticks’ in the Users Documentation, that’s okay too. We’re pretty loose around here.”

In this part . . .

You get an overview of the Python programming language, an introduction to its interactive and developer environment, and a walkthrough of the building blocks that make up Python programs.

Chapter 1 describes the history of Python and all the exciting things it's being used for today. You find out why computers are both the fastest and dumbest things around. Best of all, you discover why it's called *Python* anyway.

Chapter 2 lets you talk to Python via its interactive mode and IDLE environment. You write a few basic programs and find out how to get Python to carry out commands for you, how to get Python to tell you things, and how to import tools that let you do even more.

Chapter 3 introduces you to Python's data types and code blocks, the chunks you use to build programs.

Chapter 4 shows you a working program. You see how all the chunks of a Python program talk to each other, and you find out something about the design philosophies behind Python programs.

Chapter 5 lets you try on a programmer's hat to understand how programmers work and why they make the design decisions they do. (Unfortunately, it doesn't explain the relevance of caffeinated sodas to this process — you'll have to figure that out for yourself.) There's also a very useful section on strategies for debugging programs, which is a huge part of every programmer's job.

Chapter 1

Introducing Python

In This Chapter

- ▶ The history of Python
 - ▶ What people use Python for
 - ▶ Useful concepts for Python programming
-

Welcome to Python! If you're the type of person who wants to know what you're getting into, this chapter is for you. We give you a quick history of Python and its community of developers. You find out what Python is and isn't good for (the "is" section is much longer than the "isn't" section) and the most important principles of good Python programming. If you're new to programming, you'll see how it's very similar to a task you're probably familiar with.

The Right Tool for the Job

Python is a general-purpose, high-level language that can be extended and embedded (included in applications as a tool for writing macros). That makes Python a smart choice for many programming problems, both small and large, and not so good for a couple of computing tasks.

Good uses of Python

Python is ideal for projects that require quick development. It supports multiple programming philosophies, so it's good for programs that require flexibility. The many packages and modules already written for Python provide versatility and save you time.

The story of Python

Guido van Rossum created Python and is affectionately bestowed with the title “Benevolent Dictator For Life” by the Python community. In the late 1980s, Guido liked features of several programming languages, but none of them had all the features he wanted. Specifically, he wanted a language that had the following features:

- ✓ **Scripting language:** A *script* is a program that controls other programs. Scripting languages are good for quick development and prototyping because they’re good at passing messages from one component to another and at handling fiddly stuff like memory management so that the programmer doesn’t have to. Python has grown beyond scripting languages, which are used mostly for small applications. The Python community prefers to call Python a *dynamic programming language*.
 - ✓ **Indentation for statement grouping:** Python specifies that several statements are part of a single group by indenting them. The indented group is called a *code block*. Other languages use different syntax or punctuation for statement grouping. For example, the C programming language uses { to begin an instruction and } to end it. Indentation is considered good practice in other languages also, but Python was one of the first to *enforce* indentation. Indentation makes code easier to read, and code blocks set off with indentation have fewer begin/end words and punctuation to accidentally leave out (which means fewer bugs).
 - ✓ **High-level data types:** Computers store everything in 1s and 0s, but humans need to work with data in more complex forms, such as text. A language that supports such complex data is said to have *high-level data types*. A high-level data type is easy to
- manipulate. For example, Python strings can be searched, sliced, joined, split, set to upper- or lowercase, or have white space removed. High-level data types in Python, such as lists and dicts (which can store other data types), encompass much more functionality than in other languages.
- ✓ **Extensibility:** An extensible programming language can be added to. These languages are very powerful because additions make them suitable for multiple applications and operating systems. Extensions can add data types or concepts, modules, and plug-ins. Python is extensible in several ways. A core group of programmers works on modifying and improving the language, while hundreds of other programmers write modules for specific purposes.
 - ✓ **Interpreted:** Interpreted languages run directly from source code that humans generate (whereas programs written in *compiled languages*, like C++, must be translated to machine code before they can run). Interpreted languages run more slowly because the translation takes place on the fly, but development and debugging is faster because you don’t have to wait for the compiler. Interpreted languages are easier to run on multiple operating systems. In the case of Python, it’s easy to write code that works on multiple operating systems — with no need to make modifications.
- People argue over whether Python is an interpreted or compiled language. Although Python works like an interpreted language in many ways, its code is compiled before execution (like Java), and many of its capabilities run at full machine speed because they’re written in C — leaving you free to focus on making your application work.