ZED SHAW'S HARD WAY SERIES

Learn PYTHON the HARD WAY

THIRD EDITION

A Very Simple Introduction to the Terrifyingly Beautiful World of Computers and Code

ZED A. SHAW

LEARN PYTHON THE HARD WAY

A Very Simple Introduction
to the Terrifyingly Beautiful World
of Computers and Code

Third Edition

Zed A. Shaw

Addison-Wesley

Upper Saddle River, NJ • Boston • Indianapolis • San Francisco
New York • Toronto • Montreal • London • Munich • Paris • Madrid
Capetown • Sydney • Tokyo • Singapore • Mexico City

Library of Congress Cataloging-in-Publication Data
Shaw, Zed.
Learn Python the hard way : a very simple introduction to the terrifyingly beautiful world of computers and code / Zed A. Shaw.—Third edition.
pages cm
Includes index.
ISBN 978-0-321-88491-6 (paperback : alkaline paper)
1. Python (Computer program language) 2. Python (Computer program language)—Problems, exercises, etc. 3. Computer programming—Problems, exercises, etc. I. Title.
QA76.73.P98S53 2014
005.13'3—dc23

2013029738

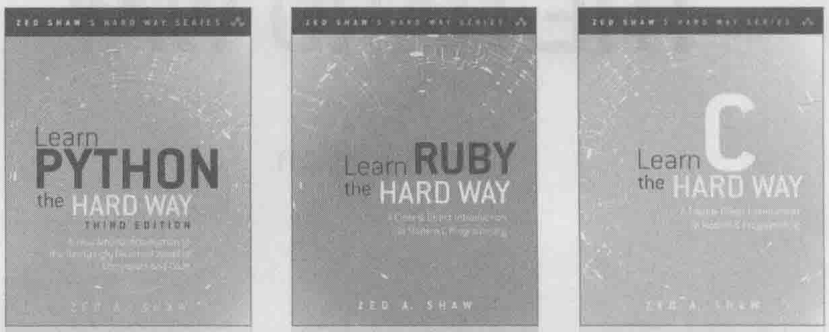
ISBN-13: 978-0-321-88491-6
ISBN-10: 0-321-88491-4

Text printed in the United States on recycled paper at RR Donnelley in Crawfordsville, Indiana.
Eighth Printing, November 2015

LEARN PYTHON THE HARD WAY

Third Edition

Zed Shaw's Hard Way Series



Addison-Wesley

Visit **informit.com/hardway** for a complete list of available publications.

Zed Shaw's Hard Way Series emphasizes instruction and *making* things as the best way to get started in many computer science topics. Each book in the series is designed around short, understandable exercises that take you through a course of instruction that creates working software. All exercises are thoroughly tested to verify they work with real students, thus increasing your chance of success. The accompanying video walks you through the code in each exercise. Zed adds a bit of humor and inside jokes to make you laugh while you're learning.

Make sure to connect with us!
informit.com/socialconnect

informIT.com the trusted technology learning source | Addison-Wesley | Safari Books Online

Contents

Preface 1
Acknowledgments 1
The Hard Way Is Easier 1
Reading and Writing 2
Attention to Detail 2
Spotting Differences 2
Do Not Copy-Paste 2
Using the Included Videos 3
A Word of Advice for "Visual Learners" 3
A Note on Practice and Persistence 3
A Warning for the Smarties 4
Exercise 0 The Setup 6
Mac OSX 6
OSX: What You Should See 7
Windows 7
Windows: What You Should See 8
Linux 9
Linux: What You Should See 10
Warnings for Beginners 10
Exercise 1 A Good First Program 12
What You Should See 14
Study Drills 15
Common Student Questions 16
Exercise 2 Comments and Pound Characters 18
What You Should See 18
Study Drills 18
Common Student Questions 19
Exercise 3 Numbers and Math 20
What You Should See 21
Study Drills 21
Common Student Questions 22

Exercise 4 Variables and Names 24
What You Should See 25
Study Drills 25
Common Student Questions 25
Exercise 5 More Variables and Printing 28
What You Should See 28
Study Drills 29
Common Student Questions 29
Exercise 6 Strings and Text 30
What You Should See 31
Study Drills 31
Common Student Questions 31
Exercise 7 More Printing 32
What You Should See 32
Study Drills 32
Common Student Questions 33
Exercise 8 Printing, Printing 34
What You Should See 34
Study Drills 34
Common Student Questions 34
Exercise 9 Printing, Printing, Printing 36
What You Should See 36
Study Drills 36
Common Student Questions 37
Exercise 10 What Was That? 38
What You Should See 39
Escape Sequences 39
Study Drills 40
Common Student Questions 40
Exercise 11 Asking Questions 42
What You Should See 42
Study Drills 43
Common Student Questions 43

Exercise 12 Prompting People 44
What You Should See 44
Study Drills 44
Common Student Questions 45

Exercise 13 Parameters, Unpacking, Variables 46
Hold Up! Features Have Another Name 46
What You Should See 47
Study Drills 48
Common Student Questions 48

Exercise 14 Prompting and Passing 50
What You Should See 50
Study Drills 51
Common Student Questions 51

Exercise 15 Reading Files 54
What You Should See 55
Study Drills 55
Common Student Questions 56

Exercise 16 Reading and Writing Files 58
What You Should See 59
Study Drills 59
Common Student Questions 60

Exercise 17 More Files 62
What You Should See 63
Study Drills 63
Common Student Questions 63

Exercise 18 Names, Variables, Code, Functions 66
What You Should See 67
Study Drills 68
Common Student Questions 68

Exercise 19 Functions and Variables 70
What You Should See 71
Study Drills 71
Common Student Questions 71

Exercise 20 Functions and Files 74
What You Should See 75
Study Drills 75
Common Student Questions 75
Exercise 21 Functions Can Return Something 78
What You Should See 79
Study Drills 79
Common Student Questions 80
Exercise 22 What Do You Know So Far? 81
What You Are Learning 81
Exercise 23 Read Some Code 82
Exercise 24 More Practice 84
What You Should See 85
Study Drills 85
Common Student Questions 85
Exercise 25 Even More Practice 86
What You Should See 87
Study Drills 88
Common Student Questions 89
Exercise 26 Congratulations, Take a Test! 90
Common Student Questions 90
Exercise 27 Memorizing Logic 92
The Truth Terms 92
The Truth Tables 93
Common Student Questions 94
Exercise 28 Boolean Practice 96
What You Should See 98
Study Drills 98
Common Student Questions 98
Exercise 29 What If 100
What You Should See 100
Study Drills 101
Common Student Questions 101

Exercise 30 Else and If. 102
What You Should See 103
Study Drills 103
Common Student Questions 103
Exercise 31 Making Decisions. 104
What You Should See 105
Study Drills 105
Common Student Questions 105
Exercise 32 Loops and Lists. 106
What You Should See 107
Study Drills 108
Common Student Questions 108
Exercise 33 While-Loops 110
What You Should See 111
Study Drills 111
Common Student Questions 112
Exercise 34 Accessing Elements of Lists 114
Study Drills 115
Exercise 35 Branches and Functions. 116
What You Should See 117
Study Drills 118
Common Student Questions 118
Exercise 36 Designing and Debugging 120
Rules for If-Statements 120
Rules for Loops 120
Tips for Debugging 121
Homework. 121
Exercise 37 Symbol Review. 122
Keywords. 122
Data Types 123
String Escape Sequences. 124
String Formats. 124
Operators 125

Reading Code 126
Study Drills 127
Common Student Questions 127
Exercise 38 Doing Things to Lists 128
What You Should See 129
Study Drills 130
Common Student Questions 130
Exercise 39 Dictionaries, Oh Lovely Dictionaries 132
What You Should See 134
Study Drills 135
Common Student Questions 135
Exercise 40 Modules, Classes, and Objects 138
Modules Are Like Dictionaries 138
Classes Are Like Modules 139
Objects Are Like Mini-Imports 140
Getting Things from Things 141
A First-Class Example 141
What You Should See 142
Study Drills 142
Common Student Questions 143
Exercise 41 Learning to Speak Object Oriented 144
Word Drills 144
Phrase Drills 144
Combined Drills 145
A Reading Test 145
Practice English to Code 147
Reading More Code 148
Common Student Questions 148
Exercise 42 Is-A, Has-A, Objects, and Classes 150
How This Looks in Code 151
About class Name(object) 153
Study Drills 153
Common Student Questions 154

Exercise 43 Basic Object-Oriented Analysis and Design 156
The Analysis of a Simple Game Engine 157
Write or Draw about the Problem 157
Extract Key Concepts and Research Them 158
Create a Class Hierarchy and Object Map for the Concepts 158
Code the Classes and a Test to Run Them 159
Repeat and Refine 161
Top Down vs. Bottom Up 161
The Code for "Gothons from Planet Percal #25" 162
What You Should See 167
Study Drills 168
Common Student Questions 168
Exercise 44 Inheritance vs. Composition 170
What is Inheritance? 170
Implicit Inheritance 171
Override Explicitly 172
Alter Before or After 172
All Three Combined 174
The Reason for super() 175
Using super() with __init__ 175
Composition 176
When to Use Inheritance or Composition 177
Study Drills 177
Common Student Questions 178
Exercise 45 You Make a Game 180
Evaluating Your Game 180
Function Style 181
Class Style 181
Code Style 182
Good Comments 182
Evaluate Your Game 183
Exercise 46 A Project Skeleton 184
Installing Python Packages 184
Creating the Skeleton Project Directory 185

Final Directory Structure 186
Testing Your Setup 187
Using the Skeleton 188
Required Quiz 188
Common Student Questions 189
Exercise 47 Automated Testing 190
Writing a Test Case 190
Testing Guidelines 192
What You Should See 192
Study Drills 193
Common Student Questions 193
Exercise 48 Advanced User Input 194
Our Game Lexicon 194
Breaking Up a Sentence 195
Lexicon Tuples 195
Scanning Input 195
Exceptions and Numbers 196
What You Should Test 196
Design Hints 198
Study Drills 198
Common Student Questions 198
Exercise 49 Making Sentences 200
Match and Peek 200
The Sentence Grammar 201
A Word on Exceptions 203
What You Should Test 204
Study Drills 204
Common Student Questions 204
Exercise 50 Your First Website 206
Installing lpthw.web 206
Make a Simple "Hello World" Project 207
What's Going On? 208
Fixing Errors 209

Create Basic Templates 209
Study Drills 211
Common Student Questions 211

Exercise 51 Getting Input from a Browser 214
How the Web Works 214
How Forms Work 216
Creating HTML Forms 218
Creating a Layout Template 220
Writing Automated Tests for Forms 221
Study Drills 223
Common Student Questions 224

Exercise 52 The Start of Your Web Game 226
Refactoring the Exercise 43 Game 226
Sessions and Tracking Users 231
Creating an Engine 232
Your Final Exam 235
Common Student Questions 236

Next Steps 237
How to Learn Any Programming Language 238

Advice from an Old Programmer 241

Appendix Command Line Crash Course 243
Introduction: Shut Up and Shell 243
How to Use This Appendix 243
You Will Be Memorizing Things 244
Exercise 1: The Setup 245
Do This 245
You Learned This 246
Do More 246
Exercise 2: Paths, Folders, Directories (pwd) 248
Do This 248
You Learned This 249
Do More 249
Exercise 3: If You Get Lost 250

Do This 250
You Learned This 250
Exercise 4: Make a Directory (mkdir) 250
Do This 250
You Learned This 252
Do More 252
Exercise 5: Change Directory (cd) 252
Do This 252
You Learned This 255
Do More 255
Exercise 6: List Directory (ls) 256
Do This 256
You Learned This 259
Do More 260
Exercise 7: Remove Directory (rmdir) 260
Do This 260
You Learned This 262
Do More 262
Exercise 8: Move Around (pushd, popd) 262
Do This 263
You Learned This 264
Do More 265
Exercise 9: Make Empty Files (Touch, New-Item) 265
Do This 265
You Learned This 266
Do More 266
Exercise 10: Copy a File (cp) 266
Do This 266
You Learned This 268
Do More 269
Exercise 11: Move a File (mv) 269
Do This 269
You Learned This 271
Do More 271

Exercise 12: View a File (less, MORE) 271
Do This 271
You Learned This 272
Do More 272
Exercise 13: Stream a File (cat) 272
Do This 272
You Learned This 273
Do More 273
Exercise 14: Remove a File (rm) 273
Do This 273
You Learned This 275
Do More 275
Exercise 15: Exit Your Terminal (exit) 275
Do This 275
You Learned This 276
Do More 276
Command Line Next Steps 276
Unix Bash References 276
PowerShell References 277

Index 279

Preface

This simple book is meant to get you started in programming. The title says it's the hard way to learn to write code, but it's actually not. It's only the "hard" way because it uses a technique called *instruction*. Instruction is where I tell you to do a sequence of controlled exercises designed to build a skill through repetition. This technique works very well with beginners who know nothing and need to acquire basic skills before they can understand more complex topics. It's used in everything from martial arts to music to even basic math and reading skills.

This book instructs you in Python by slowly building and establishing skills through techniques like practice and memorization, then applying them to increasingly difficult problems. By the end of the book, you will have the tools needed to begin learning more complex programming topics. I like to tell people that my book gives you your "programming black belt." What this means is that you know the basics well enough to now start learning programming.

If you work hard, take your time, and build these skills, you will learn to code.

Acknowledgments

I would like to thank Angela for helping me with the first two versions of this book. Without her, I probably wouldn't have bothered to finish it at all. She did the copy editing of the first draft and supported me immensely while I wrote it.

I'd also like to thank Greg Newman for doing the cover art for the first two editions, Brian Shumate for early website designs, and all the people who read previous editions of this book and took the time to send me feedback and corrections.

Thank you.

The Hard Way Is Easier

With the help of this book, you will do the incredibly simple things that all programmers do to learn a programming language:

1. Go through each exercise.
2. Type in each sample *exactly*.
3. Make it run.

That's it. This will be *very* difficult at first, but stick with it. If you go through this book and do each exercise for one or two hours a night, you will have a good foundation for moving on to another