CONTENTS IN DETAIL

ACKNOWLEDGMENTS xix

INTRODUCTION xxi

Who Should Read This Book and Why . . . xxii
About This Book . . . xxii
Your Programming Journey . . . xxiv

PART I: GETTING STARTED 1

1
DEALING WITH ERRORS AND ASKING FOR HELP 3

How to Understand Python Error Messages . . . 4
    Examining Tracebacks . . . 4
    Searching for Error Messages . . . 7
Preventing Errors with Linters . . . 8
How to Ask for Programming Help . . . 9
    Limit Back and Forth by Providing Your Information Upfront . . . 10
    State Your Question in the Form of an Actual Question . . . 10
    Ask Your Question on the Appropriate Website . . . 10
    Summarize Your Question in the Headline . . . 11
    Explain What You Want the Code to Do . . . 11
    Include the Full Error Message . . . 11
    Share Your Complete Code . . . 11
    Make Your Code Readable with Proper Formatting . . . 12
    Tell Your Helper What You've Already Tried . . . 13
    Describe Your Setup . . . 13
Examples of Asking a Question . . . 14
Summary . . . 14

2
ENVIRONMENT SETUP AND THE COMMAND LINE 17

The Filesystem . . . 18
    Paths in Python . . . 18
    The Home Directory . . . 19
    The Current Working Directory . . . 19
    Absolute vs. Relative Paths . . . 20
Programs and Processes . . . 21
The Command Line . . . 22
    Opening a Terminal Window . . . 23
    Running Programs from the Command Line . . . 23
    Using Command Line Arguments . . . 24
    Running Python Code from the Command Line with -c . . . 26
    Running Python Programs from the Command Line . . . 26

Running the py.exe Program . . . 26
Running Commands from a Python Program . . . 27
Minimizing Typing with Tab Completion . . . 27
Viewing the Command History . . . 28
Working with Common Commands . . . 28
Environment Variables and PATH . . . 35
Viewing Environment Variables . . . 36
Working with the PATH Environment Variable . . . 36
Changing the Command Line's PATH Environment Variable . . . 37
Permanently Adding Folders to PATH on Windows . . . 38
Permanently Adding Folders to PATH on macOS and Linux . . . 39
Running Python Programs Without the Command Line . . . 39
Running Python Programs on Windows . . . 40
Running Python Programs on macOS . . . 41
Running Python Programs on Ubuntu Linux . . . 41
Summary . . . 42

PART III: BEST PRACTICES, TOOLS, AND TECHNIQUES 43

3
CODE FORMATTING WITH BLACK 45

How to Lose Friends and Alienate Co-Workers . . . 46
Style Guides and PEP 8 . . . 46
Horizontal Spacing . . . 47
Use Space Characters for Indentation . . . 47
Spacing Within a Line . . . 48
Vertical Spacing . . . 51
A Vertical Spacing Example . . . 51
Vertical Spacing Best Practices . . . 52
Black: The Uncompromising Code Formatter . . . 53
Installing Black . . . 54
Running Black from the Command Line . . . 54
Disabling Black for Parts of Your Code . . . 57
Summary . . . 58

4
CHOOSING UNDERSTANDABLE NAMES 59

Casing Styles . . . 60
PEP 8's Naming Conventions . . . 61
Appropriate Name Length . . . 61
Too Short Names . . . 61
Too Long Names . . . 63
Make Names Searchable . . . 64
Avoid Jokes, Puns, and Cultural References . . . 64
Don't Overwrite Built-in Names . . . 65
The Worst Possible Variable Names Ever . . . 66
Summary . . . 67

5 FINDING CODE SMELLS 69

Duplicate Code . . . 70
Magic Numbers . . . 71
Commented-Out Code and Dead Code . . . 74
Print Debugging . . . 75
Variables with Numeric Suffixes . . . 76
Classes That Should Just Be Functions or Modules . . . 77
List Comprehensions Within List Comprehensions . . . 77
Empty except Blocks and Poor Error Messages . . . 79
Code Smell Myths . . . 80
    Myth: Functions Should Have Only One return Statement at the End . . . 80
    Myth: Functions Should Have at Most One try Statement . . . 81
    Myth: Flag Arguments Are Bad . . . 82
    Myth: Global Variables Are Bad . . . 82
    Myth: Comments Are Unnecessary . . . 83
Summary . . . 84

6 WRITING PYTHONIC CODE 87

The Zen of Python . . . 88
Learning to Love Significant Indentation . . . 91
Commonly Misused Syntax . . . 92
    Use enumerate() Instead of range() . . . 92
    Use the with Statement Instead of open() and close() . . . 93
    Use is to Compare with None Instead of == . . . 94
Formatting Strings . . . 95
    Use Raw Strings If Your String Has Many Backslashes . . . 95
    Format Strings with F-Strings . . . 96
Making Shallow Copies of Lists . . . 97
Pythonic Ways to Use Dictionaries . . . 98
    Use get() and setdefault() with Dictionaries . . . 98
    Use collections.defaultdict for Default Values . . . 99
    Use Dictionaries Instead of a switch Statement . . . 100
Conditional Expressions: Python's "Ugly" Ternary Operator . . . 101
Working with Variable Values . . . 103
    Chaining Assignment and Comparison Operators . . . 103
    Checking Whether a Variable Is One of Many Values . . . 103
Summary . . . 104

7 PROGRAMMING JARGON 107

Definitions . . . 108
    Python the Language and Python the Interpreter . . . 108
    Garbage Collection . . . 109
    Literals . . . 109
    Keywords . . . 110
    Objects, Values, Instances, and Identities . . . 111
    Items . . . 114

Mutable and Immutable 114
Indexes, Keys, and Hashes. 117
Containers, Sequences, Mapping, and Set Types 119
Dunder Methods and Magic Methods 120
Modules and Packages 120
Callables and First-Class Objects 121
Commonly Confused Terms 122
Statements vs. Expressions 122
Block vs. Clause vs. Body. 123
Variable vs. Attribute 124
Function vs. Method 124
Iterable vs. Iterator 125
Syntax vs. Runtime vs. Semantic Errors. 126
Parameters vs. Arguments 128
Type Coercion vs. Type Casting 128
Properties vs. Attributes 128
Bytecode vs. Machine Code. 129
Script vs. Program, Scripting Language vs. Programming Language 129
Library vs. Framework vs. SDK vs. Engine vs. API 130
Summary 131
Further Reading 131

8
COMMON PYTHON GOTCHAS 133

Don't Add or Delete Items from a List While Looping Over It 134
Don't Copy Mutable Values Without copy.copy() and copy.deepcopy(). 140
Don't Use Mutable Values for Default Arguments 142
Don't Build Strings with String Concatenation 144
Don't Expect sort() to Sort Alphabetically. 146
Don't Assume Floating-Point Numbers Are Perfectly Accurate. 147
Don't Chain Inequality != Operators. 149
Don't Forget the Comma in Single-Item Tuples 150
Summary 150

9
ESOTERIC PYTHON ODDITIES 153

Why 256 Is 256 but 257 Is Not 257. 154
String Interning 155
Python's Fake Increment and Decrement Operators 156
All of Nothing 157
Boolean Values Are Integer Values 158
Chaining Multiple Kinds of Operators. 159
Python's Antigravity Feature. 160
Summary 160

10
WRITING EFFECTIVE FUNCTIONS 161

Function Names. 162
Function Size Trade-Offs 162

Function Parameters and Arguments 165
Default Arguments 165
Using * and ** to Pass Arguments to Functions 166
Using * to Create Variadic Functions 167
Using ** to Create Variadic Functions 169
Using * and ** to Create Wrapper Functions 171
Functional Programming 172
Side Effects 172
Higher-Order Functions 174
Lambda Functions 174
Mapping and Filtering with List Comprehensions 175
Return Values Should Always Have the Same Data Type 177
Raising Exceptions vs. Returning Error Codes 178
Summary 179

11
COMMENTS, DOCSTRINGS, AND TYPE HINTS 181

Comments 182
Comment Style 183
Inline Comments 184
Explanatory Comments 184
Summary Comments 185
"Lessons Learned" Comments 185
Legal Comments 186
Professional Comments 186
Codetags and TODO Comments 187
Magic Comments and Source File Encoding 187
Docstrings 188
Type Hints 190
Using Static Analyzers 192
Setting Type Hints for Multiple Types 194
Setting Type Hints for Lists, Dictionaries, and More 195
Backporting Type Hints with Comments 196
Summary 197

12
ORGANIZING YOUR CODE PROJECTS WITH GIT 199

Git Commits and Repos 200
Using Cookiecutter to Create New Python Projects 200
Installing Git 202
Configuring Your Git Username and Email 203
Installing GUI Git Tools 203
The Git Workflow 204
How Git Keeps Track of File Status 204
Why Stage Files? 206
Creating a Git Repo on Your Computer 206
Adding Files for Git to Track 208
Ignoring Files in the Repo 209
Committing Changes 210
Deleting Files from the Repo 214
Renaming and Moving Files in the Repo 215

Viewing the Commit Log 216
Recovering Old Changes 217
    Undoing Uncommitted Local Changes 218
    Unstaging a Staged File 218
    Rolling Back the Most Recent Commits 218
    Rolling Back to a Specific Commit for a Single File 219
    Rewriting the Commit History 220
GitHub and the git push Command 221
    Pushing an Existing Repository to GitHub 222
    Cloning a Repo from an Existing GitHub Repo 222
Summary 223

13
MEASURING PERFORMANCE AND BIG O ALGORITHM ANALYSIS **225**

The timeit Module 226
The cProfile Profiler 228
Big O Algorithm Analysis 230
Big O Orders 230
    A Bookshelf Metaphor for Big O Orders 231
    Big O Measures the Worst-Case Scenario 235
Determining the Big O Order of Your Code 237
    Why Lower Orders and Coefficients Don't Matter 238
    Big O Analysis Examples 239
    The Big O Order of Common Function Calls 242
    Analyzing Big O at a Glance 243
    Big O Doesn't Matter When n Is Small, and n Is Usually Small 244
Summary 244

14
PRACTICE PROJECTS **247**

The Tower of Hanoi 248
    The Output 249
    The Source Code 250
    Writing the Code 252
Four-in-a-Row 259
    The Output 259
    The Source Code 260
    Writing the Code 264
Summary 271

PART III: OBJECT-ORIENTED PYTHON **273**

15
OBJECT-ORIENTED PROGRAMMING AND CLASSES **275**

Real-World Analogy: Filling Out a Form 276
Creating Objects from Classes 278

Creating a Simple Class: WizCoin 279
    Methods, __init__(), and self 280
    Attributes 282
    Private Attributes and Private Methods 282
The type() Function and __qualname__ Attribute 284
Non-OOP vs. OOP Examples: Tic-Tac-Toe 285
Designing Classes for the Real World Is Hard 290
Summary 291

16
OBJECT-ORIENTED PROGRAMMING AND INHERITANCE 293

How Inheritance Works 294
    Overriding Methods 296
    The super() Function 297
    Favor Composition Over Inheritance 299
    Inheritance's Downside 301
The isinstance() and issubclass() Functions 303
Class Methods 304
Class Attributes 306
Static Methods 306
When to Use Class and Static Object-Oriented Features 307
Object-Oriented Buzzwords 307
    Encapsulation 307
    Polymorphism 308
When Not to Use Inheritance 308
Multiple Inheritance 309
Method Resolution Order 311
Summary 312

17
PYTHONIC OOP: PROPERTIES AND DUNDER METHODS 315

Properties 316
    Turning an Attribute into a Property 316
    Using Setters to Validate Data 319
    Read-Only Properties 320
    When to Use Properties 322
Python's Dunder Methods 322
    String Representation Dunder Methods 323
    Numeric Dunder Methods 325
    Reflected Numeric Dunder Methods 328
    In-Place Augmented Assignment Dunder Methods 330
    Comparison Dunder Methods 332
Summary 337

INDEX 339