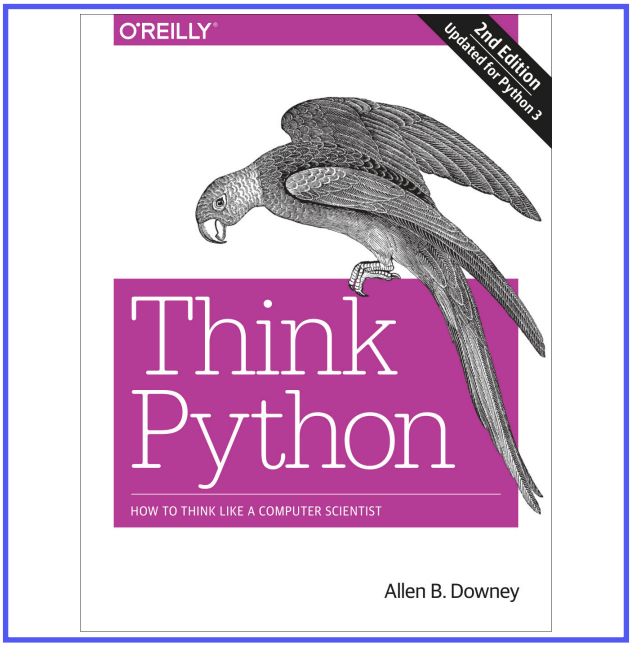


ebookbig.com

or scan the QR code to access the digital study portal.

Composition 23
Adding New Functions 23
Definitions and Uses 25
Flow of Execution 25
Parameters and Arguments 26
Variables and Parameters Are Local 27
Stack Diagrams 28
Fruitful Functions and Void Functions 29
Why Functions? 30
Debugging 30
Glossary 31
Exercises 32

4. Case Study: Interface Design. 35
The turtle Module 35
Simple Repetition 37
Exercises 38
Encapsulation 38
Generalization 39
Interface Design 40
Refactoring 41
A Development Plan 42
docstring 43
Debugging 43
Glossary 44
Exercises 44

5. Conditionals and Recursion. 47
Floor Division and Modulus 47
Boolean Expressions 48
Logical Operators 49
Conditional Execution 49
Alternative Execution 49
Chained Conditionals 50
Nested Conditionals 50
Recursion 51
Stack Diagrams for Recursive Functions 53
Infinite Recursion 53
Keyboard Input 54
Debugging 55
Glossary 56
Exercises 57

6. Fruitful Functions. 61
Return Values 61
Incremental Development 62
Composition 64
Boolean Functions 65
More Recursion 66
Leap of Faith 68
One More Example 68
Checking Types 69
Debugging 70
Glossary 71
Exercises 72

7. Iteration. 75
Reassignment 75
Updating Variables 76
The while Statement 77
break 78
Square Roots 79
Algorithms 81
Debugging 81
Glossary 82
Exercises 82

8. Strings. 85
A String Is a Sequence 85
len 86
Traversal with a for Loop 86
String Slices 87
Strings Are Immutable 88
Searching 89
Looping and Counting 89
String Methods 90
The in Operator 91
String Comparison 92
Debugging 92
Glossary 94
Exercises 95

9. Case Study: Word Play. 99
Reading Word Lists 99
Exercises 100

Search 101
Looping with Indices 103
Debugging 104
Glossary 105
Exercises 105

10. Lists. 107
A List Is a Sequence 107
Lists Are Mutable 108
Traversing a List 109
List Operations 110
List Slices 110
List Methods 111
Map, Filter and Reduce 111
Deleting Elements 113
Lists and Strings 113
Objects and Values 114
Aliasing 115
List Arguments 116
Debugging 118
Glossary 119
Exercises 120

11. Dictionaries. 125
A Dictionary Is a Mapping 125
Dictionary as a Collection of Counters 127
Looping and Dictionaries 128
Reverse Lookup 129
Dictionaries and Lists 130
Memos 131
Global Variables 133
Debugging 134
Glossary 135
Exercises 137

12. Tuples. 139
Tuples Are Immutable 139
Tuple Assignment 141
Tuples as Return Values 141
Variable-Length Argument Tuples 142
Lists and Tuples 143
Dictionaries and Tuples 144

Sequences of Sequences 146
Debugging 147
Glossary 148
Exercises 148

13. Case Study: Data Structure Selection. 151
Word Frequency Analysis 151
Random Numbers 152
Word Histogram 153
Most Common Words 155
Optional Parameters 155
Dictionary Subtraction 156
Random Words 157
Markov Analysis 158
Data Structures 159
Debugging 161
Glossary 162
Exercises 163

14. Files. 165
Persistence 165
Reading and Writing 166
Format Operator 166
Filenames and Paths 167
Catching Exceptions 169
Databases 169
Pickling 170
Pipes 171
Writing Modules 172
Debugging 173
Glossary 174
Exercises 175

15. Classes and Objects. 177
Programmer-Defined Types 177
Attributes 178
Rectangles 179
Instances as Return Values 181
Objects Are Mutable 181
Copying 182
Debugging 183
Glossary 184

Exercises 185

16. Classes and Functions. 187
Time 187
Pure Functions 188
Modifiers 189
Prototyping versus Planning 190
Debugging 192
Glossary 192
Exercises 193

17. Classes and Methods. 195
Object-Oriented Features 195
Printing Objects 196
Another Example 198
A More Complicated Example 198
The init Method 199
The __str__ Method 200
Operator Overloading 200
Type-Based Dispatch 201
Polymorphism 202
Interface and Implementation 203
Debugging 204
Glossary 204
Exercises 205

18. Inheritance. 207
Card Objects 207
Class Attributes 208
Comparing Cards 210
Decks 211
Printing the Deck 211
Add, Remove, Shuffle and Sort 212
Inheritance 213
Class Diagrams 214
Data Encapsulation 215
Debugging 217
Glossary 218
Exercises 219

19. The Goodies. 223
Conditional Expressions 223

List Comprehensions 224
Generator Expressions 225
any and all 226
Sets 226
Counters 228
defaultdict 229
Named Tuples 230
Gathering Keyword Args 232
Glossary 233
Exercises 233

20. Debugging. 235
Syntax Errors 235
I keep making changes and it makes no difference. 237
Runtime Errors 237
My program does absolutely nothing. 237
My program hangs. 238
When I run the program I get an exception. 239
I added so many print statements I get inundated with output. 240
Semantic Errors 241
My program doesn't work. 241
I've got a big hairy expression and it doesn't do what I expect. 242
I've got a function that doesn't return what I expect. 243
I'm really, really stuck and I need help. 243
No, I really need help. 243

21. Analysis of Algorithms. 245
Order of Growth 246
Analysis of Basic Python Operations 248
Analysis of Search Algorithms 250
Hashtables 251
Glossary 255

Index. 257

Preface

The Strange History of This Book

In January 1999 I was preparing to teach an introductory programming class in Java. I had taught it three times and I was getting frustrated. The failure rate in the class was too high and, even for students who succeeded, the overall level of achievement was too low.

One of the problems I saw was the books. They were too big, with too much unnecessary detail about Java, and not enough high-level guidance about how to program. And they all suffered from the trapdoor effect: they would start out easy, proceed gradually, and then somewhere around Chapter 5 the bottom would fall out. The students would get too much new material, too fast, and I would spend the rest of the semester picking up the pieces.

Two weeks before the first day of classes, I decided to write my own book. My goals were:

- Keep it short. It is better for students to read 10 pages than not read 50 pages.
- Be careful with vocabulary. I tried to minimize jargon and define each term at first use.
- Build gradually. To avoid trapdoors, I took the most difficult topics and split them into a series of small steps.
- Focus on programming, not the programming language. I included the minimum useful subset of Java and left out the rest.

I needed a title, so on a whim I chose *How to Think Like a Computer Scientist*.

My first version was rough, but it worked. Students did the reading, and they understood enough that I could spend class time on the hard topics, the interesting topics and (most important) letting the students practice.

I released the book under the GNU Free Documentation License, which allows users to copy, modify, and distribute the book.

What happened next is the cool part. Jeff Elkner, a high school teacher in Virginia, adopted my book and translated it into Python. He sent me a copy of his translation, and I had the unusual experience of learning Python by reading my own book. As Green Tea Press, I published the first Python version in 2001.

In 2003 I started teaching at Olin College and I got to teach Python for the first time. The contrast with Java was striking. Students struggled less, learned more, worked on more interesting projects, and generally had a lot more fun.

Since then I've continued to develop the book, correcting errors, improving some of the examples and adding material, especially exercises.

The result is this book, now with the less grandiose title *Think Python*. Some of the changes are:

- I added a section about debugging at the end of each chapter. These sections present general techniques for finding and avoiding bugs, and warnings about Python pitfalls.
- I added more exercises, ranging from short tests of understanding to a few substantial projects. Most exercises include a link to my solution.
- I added a series of case studies—longer examples with exercises, solutions, and discussion.
- I expanded the discussion of program development plans and basic design patterns.
- I added appendices about debugging and analysis of algorithms.

The second edition of *Think Python* has these new features:

- The book and all supporting code have been updated to Python 3.
- I added a few sections, and more details on the Web, to help beginners get started running Python in a browser, so you don't have to deal with installing Python until you want to.
- For "The turtle Module" on page 35 I switched from my own turtle graphics package, called Swampy, to a more standard Python module, `turtle`, which is easier to install and more powerful.
- I added a new chapter called "The Goodies", which introduces some additional Python features that are not strictly necessary, but sometimes handy.