CONTENTS

PREFACE  xxv

PART 1 THINKING ABOUT COMPUTING  1

Chapter 0  The Study of Computer Science  3

0.1  Why Computer Science?  3
Importance of Computer Science  3
Computer "Science"  4
Computer Science Through Computer Programming  5

0.2  The Difficulty and Promise of Programming  6
Difficulty 1: Two Things at Once  6
Difficulty 2: What is a Good Program?  8
The Promise of a Computer Program  10

0.3  Choosing a Computer Language  10
Different Computer Languages  10
Why Python?  11
Is Python the Best Language?  12

0.4  What is Computation?  12

0.5  What is a Computer?  13
Computation in Nature  13
The Human Computer  16

0.6  The Modern, Electronic Computer  17
It's the Switch!  17
The Transistor  19

0.7  A High-Level Look at a Modern Computer  23

0.8  Representing Data  25
Binary Data  25
Working with Binary  26
Limits  27
Representing Letters 28
Representing Other Data 29
What Does a Number Represent? 30
How to Talk About Quantities of Data 31
How Much Data is That? 31
0.9 Overview of Coming Chapters 32
Summary 33

PART 2 STARTING TO PROGRAM 35

Chapter 1 Beginnings 37
1.1 Practice, Practice, Practice 37
1.2 QUICKSTART, the Circumference Program 38
   Examining the Code 40
1.3 An Interactive Session 41
1.4 Parts of a Program 43
   Modules 43
   Statements and Expressions 43
   Whitespace 45
   Comments 46
   Special Python Elements: Tokens 46
   Naming Objects 48
1.5 Variables 49
   Variable Creation and Assignment 50
1.6 Objects and Types 52
   Numbers 55
   Other Built-In Types 57
   Object Types: Not Variable Types 58
   Constructing New Values 59
1.7 Operators 61
   Integer Operators 61
   Floating Point Operators 62
   Mixed Operations 62
   Order of Operations and Parentheses 63
   Augmented Assignment Operators: A Shortcut! 64
1.8 Your First Module, Math 66
1.9 Developing an Algorithm 67
1.10 Conclusion 70
1.11 Visual Vignette: Turtle Graphics 70
  Exercises 72
  Programming Projects 76

Chapter 2 Control 79
2.1 The Selection Statement for Decisions: if 79
  Booleans for Decisions 80
  The if Statement 81
  Example: What Lead is Safe in Basketball? 84
  Repetition 88
  Example: Finding Perfect Numbers 92
  Example: Classifying Numbers 96
2.2 In-Depth Control 99
  True and False: Booleans 99
  Boolean Variables 100
  Relational Operators 100
  Boolean Operators 105
  Precedence 106
  Boolean Operators Example 106
  Another Word on Assignments 110
  The Selection Statement for Decisions 112
  More on Python Decision Statements 113
  Repetition: The while Statement 116
  Sentinel Loop 125
  Summary of Repetition 126
  More on the for Statement 126
  Nesting 129
  Hailstone Sequence Example 129
2.3 Plotting Data with Pylab 132
  First Plot and Using a List 132
  More Interesting Plot: a Sine Wave 134
2.4 Computer Science Perspectives: Minimal Universal
  Computing 136
  Summary 138
  Exercises 138
  Programming Projects 144

Chapter 3 Algorithms and Program Development 147
3.1 What is an Algorithm? 147
  Example Algorithms 148
3.2 Algorithm Features 149
   Algorithm Versus Program 149
   Detailed 150
   Effective 150
   Specify Behavior 151
   General-Purpose Algorithms 151
   Can We Really Do All That? 152

3.3 What is a Program? 152
   Readability 152
   Robust 155
   Correctness 156

3.4 Strategies for Program Design 157
   Engage and Commit 157
   Understand, Then Visualize 158
   Think Before You Program 159
   Experiment 159
   Simplify 160
   Stop and Think 161
   Relax: Give Yourself a Break 161

3.5 A Simple Example 161
   Build the Skeleton 162
   Output 162
   Input 163
   Doing the Calculation 166
   Summary 171
   Exercises 171

PART 3 ORGANIZING: DATA STRUCTURES AND FUNCTIONS 173

Chapter 4   Working with Strings 175

4.1 The String Type 175
   The Triple Quote String 176
   Non Printing Characters 177
   String Representation 177
   Strings as a Sequence 178
   More Indexing and Slicing 179

4.2 String Operations 183
   Concatenation (+) and Repetition (*) 184
   Determining When + Indicates Addition or Concatenation? 185
   Comparison Operators 186
<table>
<thead>
<tr>
<th>CONTENTS</th>
<th>xi</th>
</tr>
</thead>
<tbody>
<tr>
<td>The in Operator</td>
<td>187</td>
</tr>
<tr>
<td>String Collections are Immutable</td>
<td>188</td>
</tr>
<tr>
<td>4.3 A Preview of Functions and Methods</td>
<td>189</td>
</tr>
<tr>
<td>First Cut: What is a Function?</td>
<td>189</td>
</tr>
<tr>
<td>A String Method</td>
<td>191</td>
</tr>
<tr>
<td>Determining Method Names and Method Arguments</td>
<td>193</td>
</tr>
<tr>
<td>String Methods</td>
<td>195</td>
</tr>
<tr>
<td>String Functions</td>
<td>196</td>
</tr>
<tr>
<td>4.4 Formatted Output for Strings</td>
<td>196</td>
</tr>
<tr>
<td>Descriptor Codes</td>
<td>197</td>
</tr>
<tr>
<td>Width Descriptor</td>
<td>198</td>
</tr>
<tr>
<td>Floating-Point Precision Descriptor</td>
<td>199</td>
</tr>
<tr>
<td>4.5 Control and Strings</td>
<td>200</td>
</tr>
<tr>
<td>4.6 Working with Strings</td>
<td>203</td>
</tr>
<tr>
<td>Example: Reordering a Person's Name</td>
<td>203</td>
</tr>
<tr>
<td>Palindromes</td>
<td>205</td>
</tr>
<tr>
<td>4.7 Example: Counting Poker Hands</td>
<td>208</td>
</tr>
<tr>
<td>Program to Count Poker Hands</td>
<td>211</td>
</tr>
<tr>
<td>Summary</td>
<td>218</td>
</tr>
<tr>
<td>Exercises</td>
<td>218</td>
</tr>
<tr>
<td>Programming Projects</td>
<td>222</td>
</tr>
</tbody>
</table>

**Chapter 5** Functions—QuickStart 227

| 5.1 What is a Function? | 227 |
| Why Have Functions? | 228 |
| 5.2 Python Functions | 229 |
| 5.3 Flow of Control with Functions | 231 |
| Function Flow in Detail | 232 |
| Another Function Example | 235 |
| Function Example: Word Puzzle | 236 |
| Functions Calling Functions | 242 |
| When to Use a Function | 242 |
| What if There is No Return Statement? | 243 |
| What if There Are Multiple Return Statements? | 243 |
| 5.4 Visual Vignette: Turtle Flag | 244 |
| Summary | 245 |
| Exercises | 245 |
| Programming Projects | 249 |
Chapter 6  Lists and Tuples  251
6.1  What is a List?  251
6.2  What You Already Know How to Do with Lists  253
   Indexing and Slicing  253
   Operators  254
   Functions  256
   List Iteration  256
6.3  New Things in Lists  256
   Lists are Mutable  256
   List Methods  258
6.4  Old and New Friends: Range, Split, and Other Functions and Methods  261
   Range, Split, and Multiple Assignment  261
   List to String and Back Again, Using join  262
   The Sorted Function  263
6.5  Working with Some Examples  264
   Anagrams  264
   Example: File Analysis  268
6.6  Mutable Objects and References  273
   Shallow vs. Deep Copy  277
   Mutable versus Immutable  280
6.7  Tuples  281
   Tuples from Lists  282
   Why Tuples?  283
6.8  Lists: The Data Structure  283
   Example Data Structure  284
   Other Example Data Structures  285
6.9  Algorithm Example U.S. EPA Automobile Mileage Data  286
6.10 Python Diversion: List Comprehension  296
6.11 Visual Vignette: More Plotting  297
   Numpy Arrays  297
   Plotting Trigonometric Functions  299
   Summary  300
   Exercises  300
   Programming Projects  307

Chapter 7  More on Functions  313
7.1 Functions Calling Functions  313
7.2 Scope: A First Cut 316
Arguments, Parameters, and Namespaces 317
Passing Mutable Objects 319
Returning a Complex Object 321
Refactoring events 323

7.3 Default Values and Parameters as Keywords 324
Example: Default Values and Parameter Keywords 325
Issues with Default Values 326

7.4 Functions as Objects 328
Docstrings 328

7.5 Example: Determining a Final Grade 329
The Data 329
The Design 329
Function: weightedGrade 330
Function: grade 330
Function: main 331
Example Use 332

7.6 Esoterica: “by value” or “by reference” 332
Summary 333
Exercises 333
Programming Projects 336

Chapter 8 Dictionaries and Sets 341

8.1 Dictionaries 341
Dictionary Example 342
Python Dictionaries 343
Dictionary Indexing and Assignment 343
Operators 344

8.2 Word Count Example 348
Count Words in a String 348
Word Frequency for Gettysburg Address 349
Output and Comments 353

8.3 Periodic Table Example 354
Working with CSV Files 354
Algorithm Overview 356
Functions for Divide and Conquer 356

8.4 Sets 361
History 361
What's in a Set? 361
Python Sets 361
9.9 Module: os 407
  Directory/Folder Structure 408
  os Module Functions 409
  os Module Example 411
  Summary 413
  Exercises 414
  Programming Projects 415

Chapter 10  More Program Development 417

10.1 Introduction 417

10.2 Divide and Conquer 417
  Top-Down Refinement 418

10.3 The Breast Cancer Classifier 418
  The Problem 418
  The Approach: Classification 419
  Training and Testing the Classifier 419
  Building the Classifier 419

10.4 Designing the Classifier Algorithm 420
  Divided, now Conquer 424
  Data Structures 425
  File Format 425
  Function: makeTrainingSet 425
  The makeTestSet Function 430
  The trainClassifier Function 431
  trainClassifier, Round 2 433
  Testing the Classifier on New Data 436
  The reportResults Function 441

10.5 Running the Classifier on Full Data 442
  Training versus Testing 442

10.6 Other Interesting Problems 446
  Tag Clouds 446
  S&P 500 Predictions 448
  Predicting Religion with Flags 450
  Summary 452
  Exercises 452
  Programming Projects 453

PART 4  CLASSES: MAKING YOUR OWN DATA STRUCTURES & ALGORITHMS 455

Chapter 11  Introduction to Classes 457
  Simple Student Class 457
11.1 Object-Oriented Programming 458
   Python is Object-Oriented! 458
   Characteristics of OOP 459

11.2 Working with Object-Oriented Programming 459
   Class and Instance 459

11.3 Working with Classes and Instances 460
   Built-In Class and Instance 460
   Our First Class 461
   Changing Attributes 463
   The Special Relationship Between an Instance and Class:
     instance-of 465

11.4 Object Methods 467
   Using Object Methods 467
   Writing Methods 469
   The Special Argument self 470
   Methods are the Interface to a Class Instance 472

11.5 Fitting into the Python Class Model 472
   Making Programmer-Defined Classes 472
   A Student Class 473
   Python Standard Methods 473
   Now There Are Three: Class Designer, Programmer,
   and User 478

11.6 Example: Point Class 478
   Construction 480
   Distance 480
   Summing Two Points 480
   Improving the Point Class 481

11.7 Python and OOP 483
   Encapsulation 483
   Inheritance 484
   Polymorphism 484

11.8 An Aside: Python and Other OOP Languages 484
   Public versus Private 484
   Indicating Privacy Using Double Underscores (____) 485
   Python's Philosophy 486
   Modifying an Instance 486

11.9 Conclusion 487
   Exercises 487
   Programming Projects 488
Chapter 12  More on Classes  493

12.1  More About Class Properties  493
   Rational Number (Fraction) Class Example  494

12.2  How Does Python Know?  495
   Classes, Types, and Introspection  495
   Remember Operator Overloading  497

12.3  Creating Your Own Operator Overloading  497
   Mapping Operators to Special Methods  499

12.4  Building the Rational Number Class  501
   Making the Class  501
   Review Fraction Addition  503
   Back to Adding Fractions  506
   Equality and Reducing Fractions  510
   Divide and Conquer at Work  513

12.5  What Doesn't Work (Yet)  513
   Introspection  514
   Repairing int + Rational Errors  516

12.6  Inheritance  518
   The "Find the Attribute" Game  518
   Using Inheritance  521
   Example: The Standard Model  522
   Summary  528
   Exercises  528

Chapter 13  Program Development with Classes  533

13.1  Predator–Prey Problem  533
   The Rules  534
   Simulation Using Object-Oriented Programming  535

13.2  Classes  535
   Island Class  535
   Predator and Prey, Kinds of Animals  537
   Predator and Prey Classes  541
   Object Diagram  541
   Filling the Island  541

13.3  Adding Behavior  544
   Refinement: Add Movement  544
   Refinement: Time Simulation Loop  547

13.4  Refinement: Eating, Breeding, and Keeping Time  548
   Improved Time Loop  549
   Breeding  552
Eating 554
The Tick of the Clock 555
13.5 Refinements 556
Refinement: How Many Times to Move? 556
Refinement: Graphing Population Size 557
13.6 Conclusion 559
Exercises 559

PART 5 BECOMING A BETTER PROGRAMMER 561

Chapter 14 Exceptions and Exception Handling 563
14.1 Introduction 563
14.2 Basic Exception Handling 565
A Simple Example 566
14.3 A Philosophy Concerning Exceptions 569
14.4 Exception: else and finally 570
Example: Refactoring the Reprompting of a File Name 570
14.5 Exception Usage 572
Check Input 572
Check File Opening 573
14.6 More on Exceptions 574
Raise 574
Create Your Own 575
14.7 Example: Password Manager 576
Summary 580
Exercises 580

Chapter 15 Testing 583
15.1 Why Testing? 583
Kinds of Errors 583
"Bugs" and Debugging 584
15.2 Kinds of Testing 585
Testing is Hard! 586
Importance of Testing 587
15.3 Example Problem 587
NBA Efficiency 588
Basic Algorithm 588
15.4 Incorporating Testing 591
Catching User Errors 591
Catching Developer Errors 593