# Contents

Introduction .................................................. xvii

Chapter 1 Basic Python Programming ......................... 1
  Basic Program Elements ................................................ 1
  Programs and Modules ................................................ 1
  An Example Python Program: Guessing a Number ............... 2
  Editing, Compiling, and Running Python Programs ............. 3
  Program Comments .................................................. 4
  Lexical Elements .................................................. 4
  Spelling and Naming Conventions .............................. 4
  Syntactic Elements ................................................ 5
  Literals ............................................................... 5
  String Literals .................................................... 5
  Operators and Expressions ........................................ 6
  Function Calls .................................................... 7
  The print Function ................................................ 7
  The input Function ............................................... 7
  Type Conversion Functions and Mixed-Mode Operations ....... 7
  Optional and Keyword Function Arguments .................... 7
  Variables and Assignment Statements .......................... 8
  Python Data Typing ............................................... 9
  Import Statements ................................................ 9
  Getting Help on Program Components ........................... 9
  Control Statements ............................................... 10
  Conditional Statements .......................................... 10
Using if __name__ == "__main__" ........................................... 11
Loop Statements ......................................................... 12
Strings and Their Operations ........................................ 13
  Operators ..................................................................... 13
  Formatting Strings for Output ................................. 14
  Objects and Method Calls ........................................ 16
Built-In Python Collections and Their Operations ................. 17
  Lists ........................................................................ 17
  Tuples ................................................................. 18
  Loops Over Sequences ............................................ 18
  Dictionaries .......................................................... 19
  Searching for a Value .............................................. 19
  Pattern Matching with Collections ............................ 19
Creating New Functions .................................................. 20
  Function Definitions ................................................ 20
  Recursive Functions .............................................. 21
  Nested Function Definitions .................................. 24
  Higher-Order Functions ....................................... 24
  Creating Anonymous Functions with lambda ................. 25
Catching Exceptions ......................................................... 26
Files and Their Operations ............................................. 27
  Text File Output ........................................................ 27
  Writing Numbers to a Text File .......................... 28
  Reading Text from a Text File .............................. 29
  Reading Numbers from a File ................................ 30
  Reading and Writing Objects with pickle .................. 31
Creating New Classes ..................................................... 32
Projects ................................................................. 36

Chapter 2 An Overview of Collections .............................. 39
Collection Types ............................................................. 39
  Linear Collections ................................................... 40
  Hierarchical Collections ........................................ 40
  Graph Collections .................................................. 41
  Unordered Collections .......................................... 41
  Sorted Collections .................................................. 41
  A Taxonomy of Collection Types ............................ 42
Operations on Collections ............................................... 43
Implementations of Collections ..................................... 45
Summary .................................................................... 47
Chapter 3 Searching, Sorting, and Complexity Analysis ............................ 49

Measuring the Efficiency of Algorithms ............................................. 49
  Measuring the Run Time of an Algorithm ........................................ 50
  Counting Instructions ...................................................................... 52
  Measuring the Memory Used by an Algorithm .................................. 55
  Exercises 3.1 ................................................................................. 55
Complexity Analysis ........................................................................... 56
  Orders of Complexity ...................................................................... 56
  Big-O Notation .............................................................................. 58
  The Role of the Constant of Proportionality .................................... 58
  Exercises 3.2 ................................................................................. 59
Search Algorithms ................................................................................ 60
  Search for the Minimum ................................................................... 60
  Sequential Search of a List .............................................................. 60
  Best-Case, Worst-Case, and Average-Case Performance ................... 61
  Binary Search of a Sorted List .......................................................... 62
  Comparing Data Items ................................................................. 63
  Exercises 3.3 ................................................................................. 65
Basic Sort Algorithms .......................................................................... 65
  Selection Sort ................................................................................. 66
  Bubble Sort .................................................................................... 67
  Insertion Sort ................................................................................ 68
  Best-Case, Worst-Case, and Average-Case Performance Revisited ...... 70
  Exercises 3.4 ................................................................................. 71
Faster Sorting ....................................................................................... 71
  Overview of Quicksort .................................................................... 72
  Merge Sort ...................................................................................... 76
  Exercises 3.5 ................................................................................. 79
An Exponential Algorithm: Recursive Fibonacci ................................... 80
  Converting Fibonacci to a Linear Algorithm ...................................... 81
Case Study: An Algorithm Profiler ....................................................... 82
  Request ......................................................................................... 82
  Analysis ......................................................................................... 82
  Design ........................................................................................... 84
  Implementation (Coding) ................................................................. 85
Summary .............................................................................................. 87
Review Questions ................................................................................. 88
Projects .............................................................................................. 90
**Chapter 4** Arrays and Linked Structures .......................... 93

The Array Data Structure ........................................... 93
  Random Access and Contiguous Memory ........................ 96
  Static Memory and Dynamic Memory ............................ 97
  Physical Size and Logical Size ................................. 97
  Exercises 4.1 .................................................... 98

Operations on Arrays ................................................ 98
  Increasing the Size of an Array ............................... 99
  Decreasing the Size of an Array ............................. 99
  Inserting an Item into an Array That Grows .................. 100
  Removing an Item from an Array ............................... 101
  Complexity Trade-Off: Time, Space, and Arrays .......... 102
  Exercises 4.2 .................................................... 103

Two-Dimensional Arrays (Grids) .................................. 104
  Processing a Grid ............................................... 104
  Creating and Initializing a Grid ............................. 105
  Defining a Grid Class .......................................... 105
  Ragged Grids and Multidimensional Arrays .................. 106
  Exercises 4.3 .................................................... 106

Linked Structures .................................................. 107
  Singly Linked Structures and Doubly Linked Structures ... 107
  Noncontiguous Memory and Nodes ............................ 109
  Defining a Singly Linked Node Class ......................... 111
  Using the Singly Linked Node Class ......................... 111
  Exercises 4.4 .................................................... 113

Operations on Singly Linked Structures ......................... 113
  Traversal ......................................................... 113
  Searching ......................................................... 114
  Replacement ...................................................... 115
  Inserting at the Beginning ................................. 116
  Inserting at the End ............................................ 117
  Removing at the Beginning .................................. 118
  Removing at the End ........................................... 118
  Inserting at Any Position .................................. 120
  Removing at Any Position .................................. 120
  Complexity Trade-Off: Time, Space, and Singly Linked Structures ... 123
  Exercises 4.5 .................................................... 124

Variations on a Link ................................................ 124
  A Circular Linked Structure with a Dummy Header Node .... 124
  Doubly Linked Structures ..................................... 125
  Exercises 4.6 .................................................... 128
Using Abstract Classes to Eliminate Redundant Code ........................................... 161
Designing an AbstractBag Class ........................................................................... 161
Redoing the __init__ Method in AbstractBag .................................................... 163
Modifying the Subclasses of AbstractBag .......................................................... 163
Generalizing the __add__ Method in AbstractBag ............................................. 164
An Abstract Class for All Collections ................................................................. 165
Integrating AbstractCollection into the Collection Hierarchy ........................... 165
Using Two Iterators in the __eq__ Method ............................................................ 167
Exercises 6.2 ......................................................................................................... 168
Summary ............................................................................................................. 168
Review Questions .................................................................................................. 169
Projects ................................................................................................................ 169
Chapter 7 Stacks .................................................................................................... 171
Overview of Stacks ............................................................................................... 171
Using a Stack ......................................................................................................... 172
The Stack Interface ............................................................................................... 173
Instantiating a Stack .............................................................................................. 175
Example Application: Matching Parentheses .................................................. 175
Exercises 7.1 ......................................................................................................... 177
Three Applications of Stacks ................................................................................ 178
Evaluating Arithmetic Expressions ....................................................................... 178
Evaluating Postfix Expressions ............................................................................ 179
Exercises 7.2 ......................................................................................................... 180
Converting Infix to Postfix .................................................................................... 181
Exercises 7.3 ......................................................................................................... 183
Backtracking ......................................................................................................... 183
Memory Management .......................................................................................... 186
Implementations of Stacks .................................................................................... 188
Test Driver ............................................................................................................. 188
Adding Stacks to the Collection Hierarchy .......................................................... 190
Array Implementation .......................................................................................... 190
Linked Implementation ......................................................................................... 192
The Role of the Abstract Stack Class ................................................................. 195
Time and Space Analysis of the Two Implementations ..................................... 196
Exercises 7.4 ......................................................................................................... 197
Case Study: Evaluating Postfix Expressions ...................................................... 197
Request ................................................................................................................ 197
Analysis ............................................................................................................... 197
Design ................................................................................................................ 201
Implementation .................................................................................................... 204
Summary ............................................................................................................. 207
## Chapter 8 Queues

**Overview of Queues** ........................................ 211
**The Queue Interface and Its Use** .......................... 213
  - Exercises 8.1 ........................................ 215
**Two Applications of Queues** ................................ 216
  - Simulations ............................................. 216
  - Round-Robin CPU Scheduling ......................... 218
  - Exercises 8.2 ........................................ 219
**Implementations of Queues** ................................ 219
  - A Linked Implementation of Queues .................. 220
  - An Array Implementation ............................... 221
  - Time and Space Analysis for the Two Implementations. ......................................................... 224
  - Exercises 8.3 ........................................ 224
**Case Study: Simulating a Supermarket Checkout Line** .............................. 224
  - Request ................................................... 225
  - Analysis ................................................... 225
  - The User Interface ...................................... 226
  - Classes and Responsibilities ............................ 226
**Priority Queues** ............................................. 233
  - Exercise 8.4 ........................................... 238
**Case Study: An Emergency Room Scheduler** .............................. 238
  - Request ................................................... 239
  - Analysis ................................................... 239
  - Classes .................................................... 240
  - Design and Implementation .............................. 241
**Summary** .................................................... 243
**Review Questions** ........................................... 244
**Projects** .................................................... 245

## Chapter 9 Lists

**Overview of Lists** ........................................... 247
**Using Lists** ................................................... 249
  - Index-Based Operations ................................. 249
  - Content-Based Operations .............................. 250
  - Position-Based Operations .............................. 251
  - Interfaces for Lists .................................... 255
  - Exercises 9.1 ......................................... 258
**Applications of Lists** ........................................ 258
  - Heap-Storage Management ............................. 258
Chapter 10  Trees ...................................................... 285

An Overview of Trees ............................................... 285
  Tree Terminology ............................................. 286
  General Trees and Binary Trees ......................... 287
  Recursive Definitions of Trees ......................... 288
  Exercises 10.1 ................................................ 288

Why Use a Tree? .................................................. 288

The Shape of Binary Trees ....................................... 290
  Exercises 10.2 ................................................ 293

Three Common Applications of Binary Trees ............. 293
  Heaps .......................................................... 293
  Binary Search Trees ........................................ 294
  Expression Trees ........................................... 295
  Exercises 10.3 ................................................ 297

Binary Tree Traversals .......................................... 297
  Preorder Traversal .......................................... 297
  Inorder Traversal ........................................... 298
Complexity Analysis of the Array-Based and Linked Implementations of Sets and Dictionaries ........................................ 340
Exercises 11.2 ........................................................................ 340
Hashing Strategies ................................................................ 340
The Relationship of Collisions to Density ................................. 341
Hashing with Nonnumeric Keys .............................................. 343
Linear Probing ..................................................................... 345
Quadratic Probing ............................................................... 347
Chaining ............................................................................. 347
Complexity Analysis ............................................................. 348
Exercises 11.3 ........................................................................ 349
Case Study: Profiling Hashing Strategies .................................. 350
Request .............................................................................. 350
Analysis .............................................................................. 350
Design ................................................................................ 352
Implementation .................................................................... 353
Hashing Implementation of Sets ............................................. 355
Hashing Implementation of Dictionaries ................................. 358
Exercises 11.4 ........................................................................ 360
Sorted Sets and Dictionaries .................................................. 361
Summary ............................................................................ 362
Review Questions .................................................................. 362
Projects ................................................................................ 364

Chapter 12 Graphs ................................................................. 365
Graph Terminology ............................................................... 365
Exercises 12.1 ....................................................................... 366
Why Use Graphs? ................................................................ 370
Representations of Graphs ....................................................... 370
Adjacency Matrix .................................................................. 370
Adjacency List ...................................................................... 372
Analysis of the Two Representations ...................................... 373
Further Run-Time Considerations .......................................... 374
Exercises 12.2 ....................................................................... 374
Graph Traversals ................................................................. 375
A Generic Traversal Algorithm .............................................. 375
Breadth-First and Depth-First Traversals ............................... 376
Graph Components ................................................................ 378
Exercises 12.3 ....................................................................... 379
Trees Within Graphs ............................................................ 379
Spanning Trees and Forests .................................................... 379
Minimum Spanning Tree ............................................. 380
Algorithms for Minimum Spanning Trees ......................... 380
Topological Sort ....................................................... 382
The Shortest-Path Problem .......................................... 384
Dijkstra’s Algorithm .................................................. 384
The Initialization Step .............................................. 384
The Computation Step .............................................. 386
Representing and Working with Infinity ......................... 387
Analysis ............................................................... 387
Exercises 12.4 ......................................................... 387
Floyd’s Algorithm .................................................... 388
Analysis ............................................................... 389
Developing a Graph Collection .................................... 389
Example Use of the Graph Collection ............................ 390
The Class LinkedDirectedGraph .................................. 391
The Class LinkedVertex ............................................ 395
The Class LinkedEdge ............................................... 397
Case Study: Testing Graph Algorithms .......................... 399
Request ............................................................... 399
Analysis ............................................................... 399
The Classes GraphDemoView and GraphDemoModel ........ 400
Implementation (Coding) .......................................... 401
Summary ............................................................. 405
Review Questions .................................................... 406
Projects ............................................................. 408

Appendix A Collection Framework for Python Programmers .......... 411

Index .................................................................... 413
Welcome to *Fundamentals of Python: Data Structures*. This text is intended for a second semester course in programming and problem solving with data structures. It covers the material taught in a typical Computer Science 2 course (CS2) at the undergraduate level. Although this book uses the Python programming language, you need only have a basic knowledge of programming in a high-level programming language before beginning Chapter 1.

**What You’ll Learn**

The book covers four major aspects of computing:

- **Programming basics**—Data types, control structures, algorithm development, and program design with functions are basic ideas that you need to master to solve problems with computers. You’ll review these core topics in the Python programming language and employ your understanding of them to solve a wide range of problems.

- **Object-Oriented Programming (OOP)**—Object-Oriented Programming is the dominant programming paradigm used to develop large software systems. You’ll be introduced to the fundamental principles of OOP so that you can apply them successfully. Unlike other textbooks, this book helps you develop a professional-quality framework of collection classes to illustrate these principles.

- **Data structures**—Most useful programs rely on data structures to solve problems. At the most concrete level, data structures include arrays and various types of linked structures. You’ll use these data structures to implement various types of...
collection structures, such as stacks, queues, lists, trees, bags, sets, dictionaries, and graphs. You’ll also learn to use complexity analysis to evaluate the space/time trade-offs of different implementations of these collections.

- **Software development life cycle**—Rather than isolate software development techniques in one or two chapters, this book deals with them throughout in the context of numerous case studies. Among other things, you’ll learn that coding a program is often not the most difficult or challenging aspect of problem solving and software development.

**Why Python?**

Computer technology and applications have become increasingly more sophisticated over the past two decades, and so has the computer science curriculum, especially at the introductory level. Today’s students learn a bit of programming and problem solving and are then expected to move quickly into topics like software development, complexity analysis, and data structures that, 20 years ago, were relegated to advanced courses. In addition, the ascent of object-oriented programming as the dominant paradigm has led instructors and textbook authors to bring powerful, industrial-strength programming languages such as C++ and Java into the introductory curriculum. As a result, instead of experiencing the rewards and excitement of solving problems with computers, beginning computer science students often become overwhelmed by the combined tasks of mastering advanced concepts as well as the syntax of a programming language.

This book uses the Python programming language as a way of making the second course in computer science more manageable and attractive for students and instructors alike. Python has the following pedagogical benefits:

- Python has simple, conventional syntax. Python statements are very close to those of pseudocode algorithms, and Python expressions use the conventional notation found in algebra. Thus, you can spend less time dealing with the syntax of a programming language and more time learning to solve interesting problems.

- Python has safe semantics. Any expression or statement whose meaning violates the definition of the language produces an error message.

- Python scales well. It is easy for beginners to write simple programs in Python. Python also includes all the advanced features of a modern programming language, such as support for data structures and object-oriented software development, for use when they become necessary.
Python is highly interactive. You can enter expressions and statements at an interpreter’s prompts to try out experimental code and receive immediate feedback. You can also compose longer code segments and save them in script files to be loaded and run as modules or standalone applications.

- Python is general purpose. In today’s context, this means that the language includes resources for contemporary applications, including media computing and web services.

- Python is free and is in widespread use in the industry. You can download Python to run on a variety of devices. There is a large Python user community, and expertise in Python programming has great resume value.

To summarize these benefits, Python is a comfortable and flexible vehicle for expressing ideas about computation, both for beginners and for experts. If you learn these ideas well in the first year, you should have no problems making a quick transition to other languages needed for courses later in the curriculum. Most importantly, you will spend less time staring at a computer screen and more time thinking about interesting problems to solve.

**Organization of This Book**

The approach in this book is easygoing, with each new concept introduced only when it is needed.

Chapter 1 provides a review of the features of Python programming that are needed to begin a second course in programming and problem solving in Python. The content of this chapter is organized so that you can skim it quickly if you have experience in Python programming, or you can dig a bit deeper to get up to speed in the language if you are new to Python.

The remainder of this book, in Chapters 2 through 12, covers the major topics in a typical CS2 course, especially the specification, implementation, and application of abstract data types, with the collection types as the primary vehicle and focus. Along the way, you will be thoroughly exposed to object-oriented programming techniques and the elements of good software design. Other important CS2 topics include recursive processing of data, search and sort algorithms, and the tools used in software development, such as complexity analysis and graphical notations (UML) to document designs.

Chapter 2 introduces the concept of an abstract data type (ADT) and provides an overview of various categories of collection ADTs.
Chapters 3 and 4 explore the data structures used to implement most collections and the tools for analyzing their performance trade-offs. Chapter 3 introduces complexity analysis with big-O notation. Enough material is presented to enable you to perform simple analyses of the running time and memory usage of algorithms and data structures, using search and sort algorithms as examples. Chapter 4 covers the details of processing arrays and linear linked structures, the concrete data structures used to implement most collections. You’ll learn the underlying models of computer memory that support arrays and linked structures and the time/space trade-offs that they entail.

Chapters 5 and 6 shift the focus to the principles of object-oriented design. These principles are used to organize a professional-quality framework of collection classes that will be covered in detail in later chapters.

Chapter 5 is concerned with the critical difference between interface and implementation. A single interface and several implementations of a bag collection are developed as a first example. Emphasis is placed on the inclusion of conventional methods in an interface, to allow different types of collections to collaborate in applications. For example, one such method creates an iterator, which allows you to traverse any collection with a simple loop. Other topics covered in this chapter include polymorphism and information hiding, which directly stem from the difference between interface and implementation.

Chapter 6 shows how class hierarchies can reduce the amount of redundant code in an object-oriented software system. The related concepts of inheritance, dynamic binding of method calls, and abstract classes are introduced here and used throughout the remaining chapters.

Armed with these concepts and principles, you’ll then be ready to consider the other major collection ADTs, which form the subject of Chapters 7 through 12.

Chapters 7 through 9 present the linear collections, stacks, queues, and lists. Each collection is viewed first from the perspective of its users, who are aware only of an interface and a set of performance characteristics possessed by a chosen implementation. The use of each collection is illustrated with one or more applications, and then several implementations are developed and their performance trade-offs are analyzed.

Chapters 10 through 12 present advanced data structures and algorithms as a transition to later courses in computer science. Chapter 10 discusses various tree structures, including binary search trees, heaps, and expression trees. Chapter 11 examines the
implementation of the unordered collections, bags, sets, and dictionaries, using hashing strategies. Chapter 12 introduces graphs and graph-processing algorithms.

As mentioned earlier, this book is unique in presenting a professional-quality framework of collection types. Instead of encountering a series of apparently unrelated collections, you will explore the place of each collection in an integrated whole. This approach allows you to see what the collection types have in common as well as what makes each one unique. At the same time, you will be exposed to a realistic use of inheritance and class hierarchies, topics in object-oriented software design that are difficult to motivate and exemplify at this level of the curriculum.

**Special Features**

This book explains and develops concepts carefully, using frequent examples and diagrams. New concepts are then applied in complete programs to show how they aid in solving problems. The chapters place an early and consistent emphasis on good writing habits and neat, readable documentation.

The book includes several other important features:

- **Case studies**—These present complete Python programs ranging from the simple to the substantial. To emphasize the importance and usefulness of the software development life cycle, case studies are discussed in the framework of a user request, followed by analysis, design, implementation, and suggestions for testing, with well-defined tasks performed at each stage. Some case studies are extended in end-of-chapter programming projects.

- **Chapter summaries**—Each chapter after the first one ends with a summary of the major concepts covered in the chapter.

- **Key terms**—When a new term is introduced in the text, it appears in italic.

- **Exercises**—Most major sections of each chapter after the first one end with exercise questions that reinforce the reading by asking basic questions about the material in the section. Each chapter after the second one ends with a set of review exercises.

- **Programming projects**—Each chapter ends with a set of programming projects of varying difficulty.

- **Appendix**—The appendix includes information on the collection framework used in the book.