SECOND EDITION

# FUNDAMENTALS OF PYTHON: DATA STRUCTURES

KENNETH A. LAMBERT

CENGAGE

Australia • Brazil • Mexico • Singapore • United Kingdom • United States

***Fundamentals of Python: Data Structures*, Second Edition**

**Kenneth A. Lambert**

SVP, GM Skills & Global Product Management: Jonathan Lau

Product Team Manager: Kristin McNary

Product Manager: Chris Shortt

Product Assistant: Thomas Benedetto

Executive Director, Content Design: Marah Bellegarde

Director, Learning Design: Leigh Hefferon

Learning Designer: Kate Mason

Vice President, Strategic Marketing Services: Jennifer Baker

Marketing Director: Michele McTighe

Associate Marketing Manager: Cassie Cloutier

Director, Content Delivery: Patty Stephan

Senior Content Manager: Michelle Ruelos Cannistraci

Designer: Erin K. Griffin

Cover image: Digital_Art/Shutterstock.com

Service Provider/Compositor: SPi Global



Library of Congress Control Number: 2018956860

ISBN: 978-0-357-12275-4

**Cengage**

20 Channel Center Street

Boston, MA 02210

USA

Cengage is a leading provider of customized learning solutions with employees residing in nearly 40 different countries and sales in more than 125 countries around the world. Find your local representative at **www.cengage.com.**

Cengage products are represented in Canada by Nelson Education, Ltd.

To learn more about Cengage platforms and services, register or access your online learning solution, or purchase materials for your course, visit **www.cengage.com.**



Printed in the United States of America
Print Number: 01 Print Year: 2018

# Table of Contents

# Preface



Welcome to *Fundamentals of Python: Data Structures, 2nd Edition*. 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:

1. **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.
2. **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.
3. **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.
4. **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 three 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, 30 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, especially in the second course in computer science
- 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 stand-alone 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.

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 bold face. Definitions of the key terms are also collected in a glossary.
- **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. After Chapter 2, each chapter ends with review questions.
- **Programming projects**—Each chapter ends with a set of programming projects of varying difficulty.

### New in This Edition

The most obvious change in this edition is the addition of full color. All program examples include the color coding used in Python's IDLE, so students can easily identify program elements such as keywords, comments, and function, method, and class names. Learning

objectives have been added to the beginning of each chapter. Several new figures have been added to illustrate concepts, and many programming projects have been added or reworked. A new section on iterators and higher-order functions has been added to Chapter 2. Finally, a new section on Lisp-like lists, recursive list processing, and functional programming has been added to Chapter 9.



# Instructor Resources

## MindTap

MindTap activities for *Fundamentals of Python: Data Structures* are designed to help students master the skills they need in today's workforce. Research shows employers need critical thinkers, troubleshooters, and creative problem-solvers to stay relevant in our fast-paced, technology-driven world. MindTap helps you achieve this with assignments and activities that provide hands-on practice and real-life relevance. Students are guided through assignments that help them master basic knowledge and understanding before moving on to more challenging problems.

All MindTap activities and assignments are tied to defined unit learning objectives. Hands-on coding labs provide real-life application and practice. Readings and dynamic visualizations support the lecture, while a post-course assessment measures exactly how much a class stands in terms of progress, engagement, and completion rates. Use the content and learning path as-is, or pick and choose how our materials will wrap around yours. You control what the students see and when they see it. Learn more at http://www.cengage.com/mindtap/.

## Instructor Companion Site

The following teaching tools are available for download at the Companion Site for this text. Go to instructor.cengage.com and sign in to the instructor account. Search for the textbook and add the text to the instructor dashboard.

- **Instructor's Manual:** The Instructor's Manual that accompanies this textbook includes additional instructional material to assist in class preparation, including items such as Overviews, Chapter Objectives, Teaching Tips, Quick Quizzes, Class Discussion Topics, Additional Projects, Additional Resources, and Key Terms. A sample syllabus is also available.
- **Test Bank:** Cengage Testing Powered by Cognero is a flexible, online system that allows you to:
  - author, edit, and manage test bank content from multiple Cengage solutions
  - create multiple test versions in an instant
  - deliver tests from your LMS, your classroom, or wherever you want
- **PowerPoint Presentations:** This text provides PowerPoint slides to accompany each chapter. Slides may be used to guide classroom presentations, to make available to students for chapter review, or to print as classroom handouts. Files are provided for every figure in the text. Instructors may use the files to customize PowerPoint slides, illustrate quizzes, or create handouts.

- **Solutions:** Solutions to all programming exercises are available. If an input file is needed to run a programming exercise, it is included with the solution file.
- **Source Code:** The source code is available at www.cengage.com. If an input file is needed to run a program, it is included with the source code.




The first-of-its-kind digital subscription designed specially to lower costs. Students get total access to everything Cengage has to offer on demand—in one place. That's 20,000 eBooks, 2,300 digital learning products, and dozens of study tools across 70 disciplines and over 675 courses. Currently available in select markets. Details at **www.cengage.com/unlimited**

## We Appreciate Your Feedback

We have tried to produce a high-quality text, but should you encounter any errors, please report them to lambertk@wlu.edu. A listing of errata, should they be found, as well as other information about the book, will be posted on the website http://home.wlu.edu/~lambertk/python/.

## Acknowledgments

I would like to thank my friend, Martin Osborne, for many years of advice, friendly criticism, and encouragement on several of my book projects.

I would also like to thank my students in Computer Science 112 at Washington and Lee University for classroom testing this book over several semesters.

Finally, I would like to thank Kristin McNary, Product Team Manager; Chris Shortt, Product Manager; Maria Garguilo and Kate Mason, Learning Designers; Magesh Rajagopalan, Senior Project Manager; Danielle Shaw, Tech Editor; and especially Michelle Ruelos Cannistraci, Senior Content Manager, for handling all the details of producing this edition of the book.

## About the Author

Kenneth A. Lambert is a professor of computer science and the chair of that department at Washington and Lee University. He has taught introductory programming courses for over 30 years and has been an active researcher in computer science education. Lambert has authored or coauthored a total of 28 textbooks, including a series of introductory C++ textbooks with Douglas Nance and Thomas Naps, a series of introductory Java textbooks with Martin Osborne, and a series of introductory Python textbooks.

## Dedication

To Brenda Wilson, with love and admiration.
Kenneth A. Lambert
Lexington, VA