

starting out with >>>

# PYTHON<sup>®</sup>

FOURTH EDITION



TONY GADDIS

# Starting Out with Python<sup>®</sup>

Fourth Edition

Tony Gaddis

Haywood Community College



330 Hudson Street, New York, NY 10013

<b>Senior Vice President Courseware Portfolio Management:</b>	Marcia J. Horton
<b>Director, Portfolio Management: Engineering, Computer Science &amp; Global Editions:</b>	Julian Partridge
<b>Portfolio Manager:</b>	Matt Goldstein
<b>Portfolio Management Assistant:</b>	Kristy Alaura
<b>Field Marketing Manager:</b>	Demetrius Hall
<b>Product Marketing Manager:</b>	Yvonne Vannatta
<b>Managing Producer, ECS and Math:</b>	Scott Disanno
<b>Content Producer:</b>	Sandra L. Rodriguez
<b>Composition:</b>	iEnergizer Aptara <sup>®</sup> , Ltd.
<b>Cover Designer:</b>	Joyce Wells
<b>Cover Photo:</b>	Westend61 GmbH/Alamy Stock Photo

Credits and acknowledgments borrowed from other sources and reproduced, with permission, appear on the Credits page in the endmatter of this textbook.

Copyright © 2018, 2015, 2012, 2009 Pearson Education, Inc. Hoboken, NJ 07030. All rights reserved. Manufactured in the United States of America. This publication is protected by copyright and permissions should be obtained from the publisher prior to any prohibited reproduction, storage in a retrieval system, or transmission in any form or by any means, electronic, mechanical, photocopying, recording, or otherwise. For information regarding permissions, request forms and the appropriate contacts within the Pearson Education Global Rights & Permissions department, please visit [www.pearsoned.com/permissions/](http://www.pearsoned.com/permissions/).

Many of the designations by manufacturers and seller to distinguish their products are claimed as trademarks. Where those designations appear in this book, and the publisher was aware of a trademark claim, the designations have been printed in initial caps or all caps.

The author and publisher of this book have used their best efforts in preparing this book. These efforts include the development, research, and testing of theories and programs to determine their effectiveness. The author and publisher make no warranty of any kind, expressed or implied, with regard to these programs or the documentation contained in this book. The author and publisher shall not be liable in any event for incidental or consequential damages with, or arising out of, the furnishing, performance, or use of these programs.

- Pearson Education Ltd., London
- Pearson Education Singapore, Pte. Ltd
- Pearson Education Canada, Inc.
- Pearson Education Japan
- Pearson Education Australia PTY, Ltd
- Pearson Education North Asia, Ltd., Hong Kong
- Pearson Education de Mexico, S.A. de C.V.
- Pearson Education Malaysia, Pte. Ltd.

- Pearson Education, Inc., Hoboken

### **Library of Congress Cataloging-in-Publication Data**

Names: Gaddis, Tony, author.

Title: Starting out with Python/Tony Gaddis, Haywood Community College.

Description: Fourth edition. | Boston : Pearson, [2018] | Includes index.

Identifiers: LCCN 2016058388 | ISBN 9780134444321 (alk. paper) | ISBN 0134444329 (alk. paper)

Subjects: LCSH: Python (Computer program language)

Classification: LCC QA76.73.P98 G34 2018 | DDC 005.13/3—dc23 LC record available at <https://lcn.loc.gov/2016058388>

1 17



ISBN 10: 0-13-444432-9

ISBN 13: 978-0-13-444432-1

# Contents in a Glance

1. [Preface xiii](#)
1. [Chapter 1 Introduction to Computers and Programming 1](#)
2. [Chapter 2 Input, Processing, and Output 31](#)
3. [Chapter 3 Decision Structures and Boolean Logic 109](#)
4. [Chapter 4 Repetition Structures 159](#)
5. [Chapter 5 Functions 209](#)
6. [Chapter 6 Files and Exceptions 287](#)
7. [Chapter 7 Lists and Tuples 343](#)
8. [Chapter 8 More About Strings 407](#)
9. [Chapter 9 Dictionaries and Sets 439](#)
10. [Chapter 10 Classes and Object-Oriented Programming 489](#)
11. [Chapter 11 Inheritance 551](#)
12. [Chapter 12 Recursion 577](#)
13. [Chapter 13 GUI Programming 597](#)
1. [Appendix A Installing Python 659](#)
2. [Appendix B Introduction to IDLE 663](#)
3. [Appendix C The ASCII Character Set 671](#)
4. [Appendix D Predefined Named Colors 673](#)
5. [Appendix E More About the `import` Statement 679](#)
6. [Appendix F Installing Modules with the `pip` Utility 683](#)
7. [Appendix G Answers to Checkpoints 685](#)
8. [Index 703](#)
9. [Credits 721](#)

# Contents

1. [Preface xiii](#)
1. [Chapter 1 Introduction to Computers and Programming 1](#)
  1. [1.1 Introduction 1](#)
  2. [1.2 Hardware and Software 2](#)
  3. [1.3 How Computers Store Data 7](#)
  4. [1.4 How a Program Works 12](#)
  5. [1.5 Using Python 20](#)
  1. [Review Questions 24](#)
2. [Chapter 2 Input, Processing, and Output 31](#)
  1. [2.1 Designing a Program 31](#)
  2. [2.2 Input, Processing, and Output 35](#)
  3. [2.3 Displaying Output with the `print` Function 36](#)
  4. [2.4 Comments 39](#)
  5. [2.5 Variables 40](#)
  6. [2.6 Reading Input from the Keyboard 49](#)
  7. [2.7 Performing Calculations 53](#)
  8. [2.8 More About Data Output 65](#)
  9. [2.9 Named Constants 73](#)
  10. [2.10 Introduction to Turtle Graphics 74](#)
  1. [Review Questions 100](#)
  2. [Programming Exercises 104](#)
3. [Chapter 3 Decision Structures and Boolean Logic 109](#)
  1. [3.1 The `if` Statement 109](#)
  2. [3.2 The `if-else` Statement 118](#)
  3. [3.3 Comparing Strings 121](#)
  4. [3.4 Nested Decision Structures and the `if-elif-else` Statement 125](#)

5. [3.5 Logical Operators 133](#)
6. [3.6 Boolean Variables 139](#)
7. [3.7 Turtle Graphics: Determining the State of the Turtle 140](#)
1. [Review Questions 148](#)
2. [Programming Exercises 151](#)
4. [Chapter 4 Repetition Structures 159](#)
  1. [4.1 Introduction to Repetition Structures 159](#)
  2. [4.2 The `while` Loop: A Condition-Controlled Loop 160](#)
  3. [4.3 The `for` Loop: A Count-Controlled Loop 168](#)
  4. [4.4 Calculating a Running Total 179](#)
  5. [4.5 Sentinels 182](#)
  6. [4.6 Input Validation Loops 185](#)
  7. [4.7 Nested Loops 190](#)
  8. [4.8 Turtle Graphics: Using Loops to Draw Designs 197](#)
  1. [Review Questions 201](#)
  2. [Programming Exercises 203](#)
5. [Chapter 5 Functions 209](#)
  1. [5.1 Introduction to Functions 209](#)
  2. [5.2 Defining and Calling a Void Function 212](#)
  3. [5.3 Designing a Program to Use Functions 217](#)
  4. [5.4 Local Variables 223](#)
  5. [5.5 Passing Arguments to Functions 225](#)
  6. [5.6 Global Variables and Global Constants 235](#)
  7. [5.7 Introduction to Value-Returning Functions: Generating Random Numbers 239](#)
  8. [5.8 Writing Your Own Value-Returning Functions 250](#)
  9. [5.9 The `math` Module 261](#)
  10. [5.10 Storing Functions in Modules 264](#)
  11. [5.11 Turtle Graphics: Modularizing Code with Functions 268](#)

1. [Review Questions 275](#)
2. [Programming Exercises 280](#)
6. [Chapter 6 Files and Exceptions 287](#)
  1. [6.1 Introduction to File Input and Output 287](#)
  2. [6.2 Using Loops to Process Files 304](#)
  3. [6.3 Processing Records 311](#)
  4. [6.4 Exceptions 324](#)
  1. [Review Questions 337](#)
  2. [Programming Exercises 340](#)
7. [Chapter 7 Lists and Tuples 343](#)
  1. [7.1 Sequences 343](#)
  2. [7.2 Introduction to Lists 343](#)
  3. [7.3 List Slicing 351](#)
  4. [7.4 Finding Items in Lists with the in Operator 354](#)
  5. [7.5 List Methods and Useful Built-in Functions 355](#)
  6. [7.6 Copying Lists 362](#)
  7. [7.7 Processing Lists 364](#)
  8. [7.8 Two-Dimensional Lists 376](#)
  9. [7.9 Tuples 380](#)
  10. [7.10 Plotting List Data with the matplotlib Package 383](#)
  1. [Review Questions 399](#)
  2. [Programming Exercises 402](#)
8. [Chapter 8 More About Strings 407](#)
  1. [8.1 Basic String Operations 407](#)
  2. [8.2 String Slicing 415](#)
  3. [8.3 Testing, Searching, and Manipulating Strings 419](#)
  1. [Review Questions 431](#)
  2. [Programming Exercises 434](#)



9. [Chapter 9 Dictionaries and Sets 439](#)
  1. [9.1 Dictionaries 439](#)
  2. [9.2 Sets 462](#)
  3. [9.3 Serializing Objects 474](#)
  1. [Review Questions 480](#)
  2. [Programming Exercises 485](#)
10. [Chapter 10 Classes and Object-Oriented Programming 489](#)
  1. [10.1 Procedural and Object-Oriented Programming 489](#)
  2. [10.2 Classes 493](#)
  3. [10.3 Working with Instances 510](#)
  4. [10.4 Techniques for Designing Classes 532](#)
  1. [Review Questions 543](#)
  2. [Programming Exercises 546](#)
11. [Chapter 11 Inheritance 551](#)
  1. [11.1 Introduction to Inheritance 551](#)
  2. [11.2 Polymorphism 566](#)
  1. [Review Questions 572](#)
  2. [Programming Exercises 574](#)
12. [Chapter 12 Recursion 577](#)
  1. [12.1 Introduction to Recursion 577](#)
  2. [12.2 Problem Solving with Recursion 580](#)
  3. [12.3 Examples of Recursive Algorithms 584](#)
  1. [Review Questions 592](#)
  2. [Programming Exercises 594](#)
13. [Chapter 13 GUI Programming 597](#)
  1. [13.1 Graphical User Interfaces 597](#)
  2. [13.2 Using the `tkinter` Module 599](#)
  3. [13.3 Display Text with `Label` Widgets 602](#)

4. [13.4 Organizing Widgets with `Frames` 605](#)
5. [13.5 `Button` Widgets and Info Dialog Boxes 608](#)
6. [13.6 Getting Input with the `Entry` Widget 611](#)
7. [13.7 Using Labels as Output Fields 614](#)
8. [13.8 Radio Buttons and Check Buttons 622](#)
9. [13.9 Drawing Shapes with the `Canvas` Widget 629](#)
1. [Review Questions 651](#)
2. [Programming Exercises 654](#)
1. [Appendix A Installing Python 659](#)
2. [Appendix B Introduction to IDLE 663](#)
3. [Appendix C The ASCII Character Set 671](#)
4. [Appendix D Predefined Named Colors 673](#)
5. [Appendix E More About the `import` Statement 679](#)
6. [Appendix F Installing Modules with the `pip` Utility 683](#)
7. [Appendix G Answers to Checkpoints 685](#)
8. [Index 703](#)
9. [Credits 721](#)



# Location of Videonotes in the Text

<a href="#">Chapter 1</a>	<a href="#">Using Interactive Mode in IDLE</a> , p. <a href="#">23</a>
	<a href="#">Performing Exercise 2</a> , p. <a href="#">28</a>
	<a href="#">The <code>print</code> Function</a> , p. <a href="#">36</a>
<a href="#">Chapter 2</a>	<a href="#">Reading Input from the Keyboard</a> , p. <a href="#">49</a>
	<a href="#">Introduction to Turtle Graphics</a> , p. <a href="#">5</a>
	<a href="#">The Sales Prediction Problem</a> , p. <a href="#">104</a>
	<a href="#">The <code>if</code> Statement</a> , p. <a href="#">109</a>
<a href="#">Chapter 3</a>	<a href="#">The <code>if-else</code> Statement</a> , p. <a href="#">118</a>
	<a href="#">The Areas of Rectangles Problem</a> , p. <a href="#">151</a>
	<a href="#">The <code>while</code> Loop</a> , p. <a href="#">160</a>
<a href="#">Chapter 4</a>	<a href="#">The <code>for</code> Loop</a> , p. <a href="#">168</a>
	<a href="#">The Bug Collector Problem</a> , p. <a href="#">203</a>
	<a href="#">Defining and Calling a Function</a> , p. <a href="#">212</a>
	<a href="#">Passing Arguments to a Function</a> , p. <a href="#">225</a>
<a href="#">Chapter 5</a>	<a href="#">Writing a Value-Returning Function</a> , p. <a href="#">250</a>
	<a href="#">The Kilometer Converter Problem</a> , p. <a href="#">280</a>
	<a href="#">The Feet to Inches Problem</a> , p. <a href="#">281</a>
	<a href="#">Using Loops to Process Files</a> , p. <a href="#">304</a>
<a href="#">Chapter 6</a>	<a href="#">File Display</a> , p. <a href="#">340</a>
	<a href="#">List Slicing</a> , p. <a href="#">351</a>
<a href="#">Chapter 7</a>	<a href="#">The Lottery Number Generator Problem</a> , p. <a href="#">402</a>

[Chapter 8](#) [The Vowels and Consonants problem](#), p. [435](#)

[Introduction to Dictionaries](#), p. [439](#)

[Chapter 9](#) [Introduction to Sets](#), p. [462](#)

[The Capital Quiz Problem](#), p. [486](#)

[Classes and Objects](#), p. [493](#)

[Chapter 10](#)

[The `Pet` class](#), p. [546](#)

[Chapter 11](#) [The `Person` and `Customer` Classes](#), p. [575](#)

[Chapter 12](#) [The Recursive Multiplication Problem](#), p. [594](#)

[Creating a Simple GUI application](#), p. [602](#)

[Chapter 13](#) [Responding to Button Clicks](#), p. [608](#)

[The Name and Address Problem](#), p. [654](#)

[Appendix B](#) [Introduction to IDLE](#), p. [663](#)

# Preface

Welcome to *Starting Out with Python*, Fourth Edition. This book uses the Python language to teach programming concepts and problem-solving skills, without assuming any previous programming experience. With easy-to-understand examples, pseudocode, flowcharts, and other tools, the student learns how to design the logic of programs then implement those programs using Python. This book is ideal for an introductory programming course or a programming logic and design course using Python as the language.

As with all the books in the *Starting Out With* series, the hallmark of this text is its clear, friendly, and easy-to-understand writing. In addition, it is rich in example programs that are concise and practical. The programs in this book include short examples that highlight specific programming topics, as well as more involved examples that focus on problem solving. Each chapter provides one or more case studies that provide step-by-step analysis of a specific problem and shows the student how to solve it.

## Control Structures First, Then Classes

Python is a fully object-oriented programming language, but students do not have to understand object-oriented concepts to start programming in Python. This text first introduces the student to the fundamentals of data storage, input and output, control structures, functions, sequences and lists, file I/O, and objects that are created from standard library classes. Then the student learns to write classes, explores the topics of inheritance and polymorphism, and learns to write recursive functions. Finally, the student learns to develop simple event-driven GUI applications.

## Changes in the Fourth Edition

This book's clear writing style remains the same as in the previous edition. However, many additions and improvements have been made, which are summarized here:

- New sections on the Python Turtle Graphics library have been added to [Chapters 2](#) through [5](#). The Turtle Graphics library, which is a standard part of Python, is a fun and motivating way to introduce programming concepts to students who have never written code before. The library allows the student to write Python statements that draw graphics by moving a cursor on a canvas. The new sections that have been added to this edition are:
  - [Chapter 2](#): Introduction to Turtle Graphics
  - [Chapter 3](#): Determining the State of the Turtle
  - [Chapter 4](#): Using loops to draw designs
  - [Chapter 5](#): Modularizing Turtle Graphics Code with Functions

The new Turtle Graphics sections are designed with flexibility in mind. They can be assigned as optional material, incorporated into your existing syllabus, or skipped altogether.

- [Chapter 2](#) has a new section on named constants. Although Python does not support true

constants, you can create variable names that symbolize values that should not change as the program executes. This section teaches the student to avoid the use of “magic numbers,” and to create symbolic names that his or her code more self-documenting and easier to maintain.

- [Chapter 7](#) has a new section on using the matplotlib package to plot charts and graphs from lists. The new section describes how to install the matplotlib package, and use it to plot line graphs, bar charts, and pie charts.
- [Chapter 13](#) has a new section on creating graphics in a GUI application with the Canvas widget. The new section describes how to use the Canvas widget to draw lines, rectangles, ovals, arcs, polygons, and text.
- Several new, more challenging, programming problems have been added throughout the book.
- [Appendix E](#) is a new appendix that discusses the various forms of the import statement.
- [Appendix F](#) is a new appendix that discusses installing third-party modules with the pip utility.

## Brief Overview of Each Chapter

# Chapter 1: Introduction to Computers and Programming

This chapter begins by giving a very concrete and easy-to-understand explanation of how computers work, how data is stored and manipulated, and why we write programs in high-level languages. An introduction to Python, interactive mode, script mode, and the IDLE environment are also given.

# Chapter 2: Input, Processing, and Output

This chapter introduces the program development cycle, variables, data types, and simple programs that are written as sequence structures. The student learns to write simple programs that read input from the keyboard, perform mathematical operations, and produce screen output. Pseudocode and flowcharts are also introduced as tools for designing programs. The chapter also includes an optional introduction to the turtle graphics library.



# Chapter 3: Decision Structures and Boolean Logic

In this chapter, the student learns about relational operators and Boolean expressions and is shown how to control the flow of a program with decision structures. The `if`, `if-else`, and `if-elif-else` statements are covered. Nested decision structures and logical operators are discussed as well. The chapter also includes an optional turtle graphics section, with a discussion of how to use decision structures to test the state of the turtle.

# Chapter 4: Repetition Structures

This chapter shows the student how to create repetition structures using the `while` loop and `for` loop. Counters, accumulators, running totals, and sentinels are discussed, as well as techniques for writing input validation loops. The chapter also includes an optional section on using loops to draw designs with the turtle graphics library.

# Chapter 5: Functions

In this chapter, the student first learns how to write and call void functions. The chapter shows the benefits of using functions to modularize programs and discusses the top-down design approach. Then, the student learns to pass arguments to functions. Common library functions, such as those for generating random numbers, are discussed. After learning how to call library functions and use their return value, the student learns to define and call his or her own functions. Then the student learns how to use modules to organize functions. An optional section includes a discussion of modularizing turtle graphics code with functions.

# Chapter 6: Files and Exceptions

This chapter introduces sequential file input and output. The student learns to read and write large sets of data and store data as fields and records. The chapter concludes by discussing exceptions and shows the student how to write exception-handling code.

# Chapter 7: Lists and Tuples

This chapter introduces the student to the concept of a sequence in Python and explores the use of two common Python sequences: lists and tuples. The student learns to use lists for arraylike operations, such as storing objects in a list, iterating over a list, searching for items in a list, and calculating the sum and average of items in a list. The chapter discusses slicing and many of the list methods. One- and two-dimensional lists are covered. The chapter also includes a discussion of the `matplotlib` package, and how to use it to plot charts and graphs from lists.

# Chapter 8: More About Strings

In this chapter, the student learns to process strings at a detailed level. String slicing and algorithms that step through the individual characters in a string are discussed, and several built-in functions and string methods for character and text processing are introduced.

# Chapter 9: Dictionaries and Sets

This chapter introduces the dictionary and set data structures. The student learns to store data as key-value pairs in dictionaries, search for values, change existing values, add new key-value pairs, and delete key-value pairs. The student learns to store values as unique elements in sets and perform common set operations such as union, intersection, difference, and symmetric difference. The chapter concludes with a discussion of object serialization and introduces the student to the Python `pickle` module.

# Chapter 10: Classes and Object-Oriented Programming

This chapter compares procedural and object-oriented programming practices. It covers the fundamental concepts of classes and objects. Attributes, methods, encapsulation and data hiding, `__init__` functions (which are similar to constructors), accessors, and mutators are discussed. The student learns how to model classes with UML and how to find the classes in a particular problem.



# Chapter 11: Inheritance

The study of classes continues in this chapter with the subjects of inheritance and polymorphism. The topics covered include superclasses, subclasses, how `__init__` functions work in inheritance, method overriding, and polymorphism.

# Chapter 12: Recursion

This chapter discusses recursion and its use in problem solving. A visual trace of recursive calls is provided, and recursive applications are discussed. Recursive algorithms for many tasks are presented, such as finding factorials, finding a greatest common denominator (GCD), and summing a range of values in a list, and the classic Towers of Hanoi example are presented.

# Chapter 13: GUI Programming

This chapter discusses the basic aspects of designing a GUI application using the `tkinter` module in Python. Fundamental widgets, such as labels, buttons, entry fields, radio buttons, check buttons, and dialog boxes, are covered. The student also learns how events work in a GUI application and how to write callback functions to handle events. The Chapter includes a discussion of the `Canvas` widget, and how to use it to draw lines, rectangles, ovals, arcs, polygons, and text.

## Appendix A: Installing Python

This appendix explains how to download and install the Python 3 interpreter.

## Appendix B: Introduction to IDLE

This appendix gives an overview of the IDLE integrated development environment that comes with Python.

## Appendix C: The ASCII Character Set

As a reference, this appendix lists the ASCII character set.

## Appendix D: Predefined Named Colors

This appendix lists the predefined color names that can be used with the `turtle` graphics library, `matplotlib` and `tkinter`.

## Appendix E: More About the `import` Statement

This appendix discusses various ways to use the `import` statement. For example, you can use the `import` statement to import a module, a class, a function, or to assign an alias to a module.

## Appendix F: Installing Modules with the `pip` Utility

This appendix discusses how to use the `pip` utility to install third-party modules from the Python Package Index, or PyPI.

## Appendix G: Answers to Checkpoints

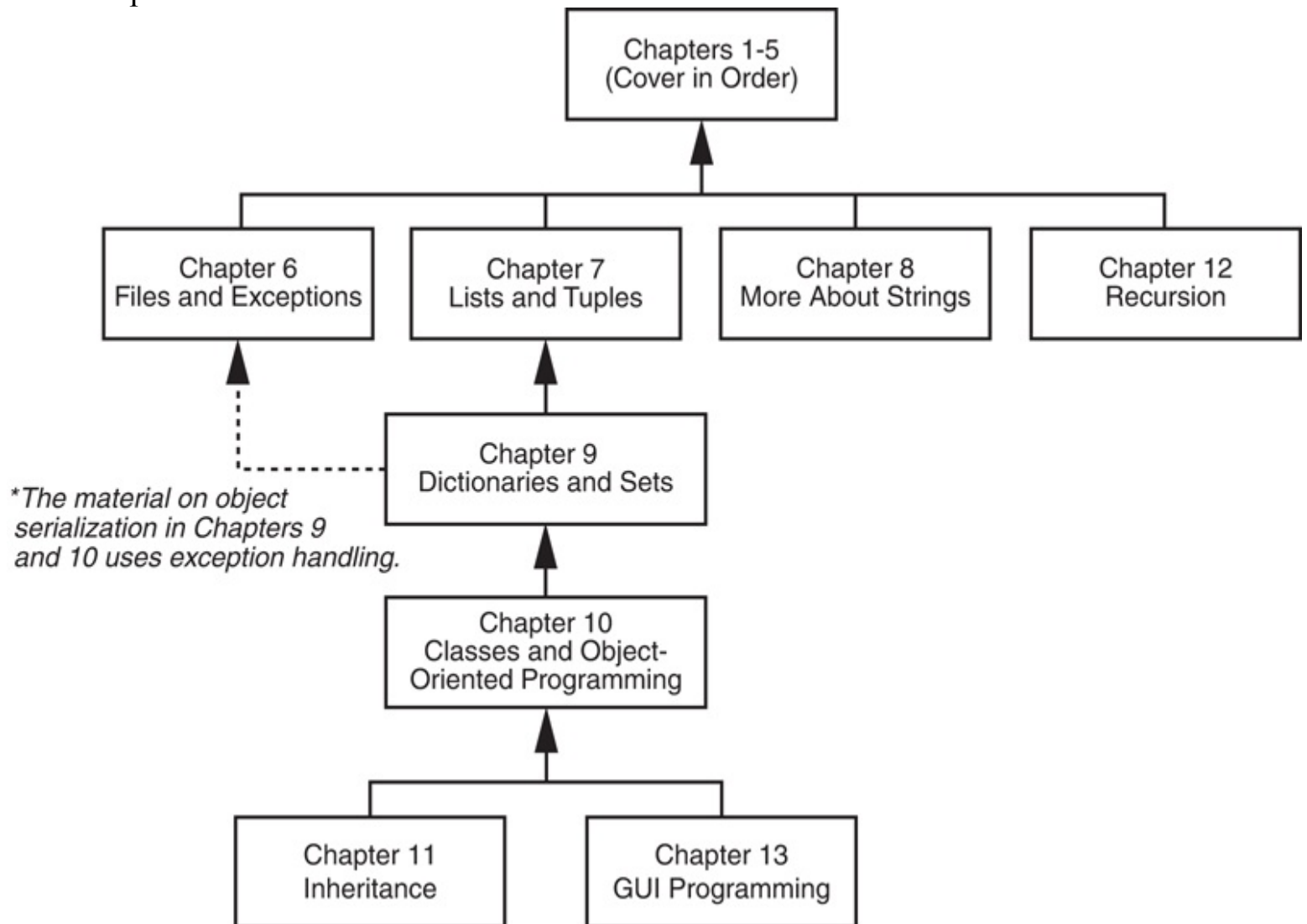
This appendix gives the answers to the Checkpoint questions that appear throughout the text.

## Organization of the Text

The text teaches programming in a step-by-step manner. Each chapter covers a major set of topics and builds knowledge as students progress through the book. Although the chapters can be easily taught in their existing sequence, you do have some flexibility in the order that you wish to cover them. [Figure P-1](#) shows chapter dependencies. Each box represents a chapter or a group of chapters. An arrow points from a chapter to the chapter that must be covered before it.

# Figure P-1 Chapter dependencies

► Description



## Features of the Text

### Concept

Each major section of the text starts with a concept statement.

### Statements

This statement concisely summarizes the main point of the section.

### Example Programs

Each chapter has an abundant number of complete and partial example programs, each designed to highlight the current topic.



### In the Spotlight Case Studies

Each chapter has one or more *In the Spotlight* case studies that provide detailed, step-by-step analysis of problems and show the student how to solve them.

### VideoNotes

Online videos developed specifically for this book are available for viewing at [www.pearsonhighered.com/cs-resources](http://www.pearsonhighered.com/cs-resources). Icons appear throughout the text alerting the student to videos about specific topics.

Notes appear at several places throughout the text. They are short explanations of interesting or often



**Notes**



**Tips**



**Warnings**



**Checkpoints**

## **Review Questions**

## **Programming Exercises**

misunderstood points relevant to the topic at hand.

Tips advise the student on the best techniques for approaching different programming problems.

Warnings caution students about programming techniques or practices that can lead to malfunctioning programs or lost data.

Checkpoints are questions placed at intervals throughout each chapter. They are designed to query the student's knowledge quickly after learning a new topic.

Each chapter presents a thorough and diverse set of review questions and exercises. They include Multiple Choice, True/False, Algorithm Workbench, and Short Answer.

Each chapter offers a pool of programming exercises designed to solidify the student's knowledge of the topics currently being studied.

# **Supplements**

## **Student Online Resources**

Many student resources are available for this book from the publisher. The following items are available at [www.pearsonhighered.com/cs-resources](http://www.pearsonhighered.com/cs-resources)

- The source code for each example program in the book
- Access to the book's companion VideoNotes

## **Instructor Resources**

The following supplements are available to qualified instructors only:

- Answers to all of the Review Questions
- Solutions for the exercises
- PowerPoint presentation slides for each chapter
- Test bank

Visit the Pearson Education Instructor Resource Center ([www.pearsonhighered.com/irc](http://www.pearsonhighered.com/irc)) or contact your local Pearson Education campus representative for information on how to access them.

# **Acknowledgments**