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 and 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 make 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.

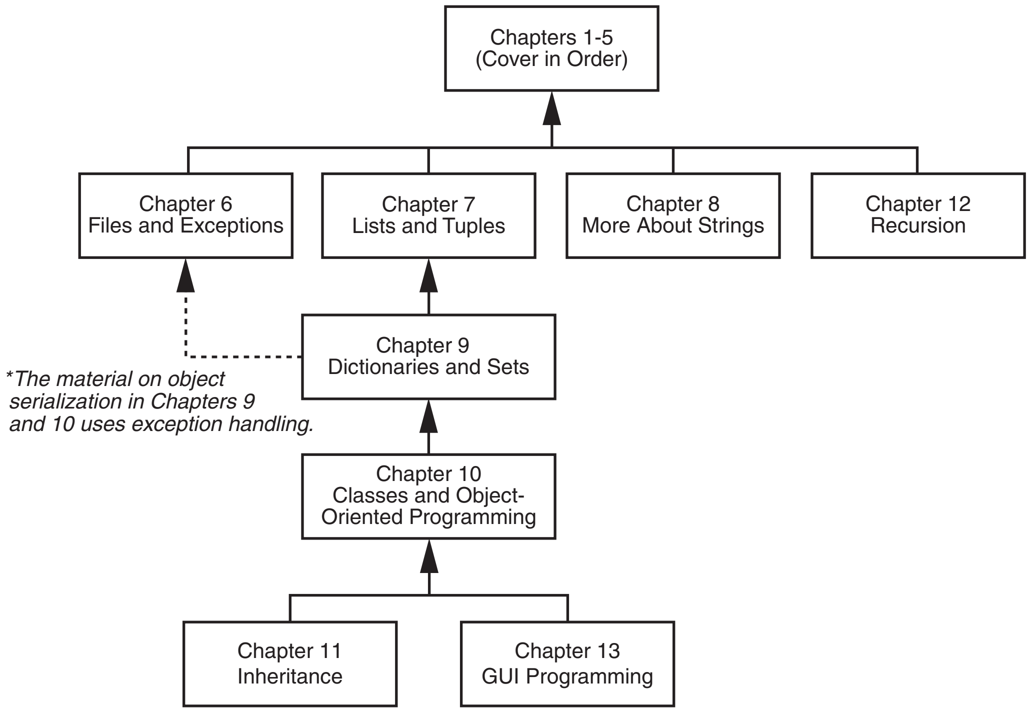
Appendix G: Answers to Checkpoints

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



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.pearsonglobaleditions.com/gaddis. Icons appear throughout the text alerting the student to videos about specific topics.
Notes	Notes appear at several places throughout the text. They are short explanations of interesting or often misunderstood points relevant to the topic at hand.
Tips	Tips advise the student on the best techniques for approaching different programming problems.
Warnings	Warnings caution students about programming techniques or practices that can lead to malfunctioning programs or lost data.
Checkpoints	Checkpoints are questions placed at intervals throughout each chapter. They are designed to query the student's knowledge quickly after learning a new topic.
Review Questions	Each chapter presents a thorough and diverse set of review questions and exercises. They include Multiple Choice, True/False, Algorithm Workbench, and Short Answer.
Programming Exercises	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.pearsonglobaleditions.com/gaddis

- 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 Instructor Resource Center (www.pearsonglobaleditions.com/gaddis) or contact your local Pearson campus representative for information on how to access them.