TABLE OF CONTENTS

Writing programs (or programming) is a very creative and rewarding activity. You can write programs for many reasons, ranging from making your living to solving a difficult data analysis problem to having fun to helping someone else solve a problem. This book assumes thatÂ everyoneÂ needs to know how to program, and that once you know how to program you will figure out what you want to do with your newfound skills.

1: INTRODUCTION
1.1: Why should you learn to write programs?
1.2: Creativity and Motivation
1.3: Computer Hardware Architecture
1.4: Understanding Programming
1.5: Words and Sentences
1.6: Conversing with Python
1.7: Terminology - Interpreter and Compiler
1.8: Writing a Program
1.9: What is a program?
1.10: The Building Blocks of Programs
1.11: What could possibly go wrong?
1.12: The Learning Journey
1.E: Introduction (Exercises)
1.G: Introduction (Glossary)

2: VARIABLES, EXPRESSIONS, AND STATEMENTS
2.1: Values and Types
2.2: Variables
2.3: Variable names and Keywords
2.4: Statements
2.5: Operators and Operands
2.6: Expressions
2.7: Order of Operations
2.8: Modulus Operator
2.9: String Operations
2.10: Asking the user for input
2.11: Comments
2.12: Choosing Mnemonic Variable Names
2.13: Debugging
2.E: Variables, Expressions, and Statements (Exercises)
2.G: Glossary

3: CONDITIONAL EXECUTION
3.1: Boolean Expressions
3.2: Logical Operators
3.3: Conditional Execution
3.4: Alternative Execution
3.5: Chained Conditionals
3.6: Nested Conditionals
3.7: Catching exceptions Using Try and Except
3.8: Short-Circuit Evaluation of Logical Expressions
3.9: Debugging
3.E: Conditional Execution (Exercises)
3.G: Conditional Execution (Glossary)

4: FUNCTIONS
4.1: Function Calls
4.2: Built-in Functions
4.3: Type Conversion Functions

4.4: Random Numbers
4.5: Math Functions
4.6: Adding New Functions
4.7: Definitions and Uses
4.8: Flow of Execution
4.9: Parameters and Arguments
4.10: Fruitful functions and void functions
4.11: Why functions?
4.12: Debugging
4.E: Functions (Exercises)
4.G: Functions (Glossary)

5: ITERATIONS

5.1: Updating Variables
5.2: The while Statement
5.3: Infinite Loops
5.4: "Infinite loops" and break
5.5: Finishing iterations with continue
5.6: Definite loops using for
5.7: Loop patterns
5.8: Counting and Summing Loops
5.9: Maximum and Minimum Loops
5.10: Debugging
5.E: Iterations (Exercises)
5.G: Iterations (Glossary)

6: STRINGS

6.1: A string is a sequence
6.2: Getting the length of a string using len
6.3: Traversal through a string with a loop
6.4: String Slices
6.5: Strings are immutable
6.6: Looping and Counting
6.7: The in operator
6.8: String Comparison
6.9: String Methods
6.10: Parsing strings
6.11: Format operator
6.12: Debugging
6.E: Strings (Exercises)
6.G: Strings (Glossary)

7: FILES

In this chapter, we start to work with Secondary Memory (or files). Secondary memory is not erased when the power is turned off. Or in the case of a USB flash drive, the data we write from our programs can be removed from the system and transported to another system.

7.1: Persistence
7.2: Opening Files
7.3: Text files and Lines
7.4: Reading Files
7.5: Searching through a File
7.6: Letting the user choose the file name
7.7: Using try, except, and open
7.8: Writing Files
7.9: Debugging
7.E: Files (Exercises)
7.G: Files (Glossary)

8: LISTS

8.1: A list is a sequence

8.2: Lists are mutable
8.3: Traversing a List
8.4: List operations
8.5: List Slices
8.6: List Methods
8.7: Deleting Elements
8.8: Lists and Functions
8.9: Lists and Strings
8.10: Parsing lines
8.11: Objects and Values
8.12: Aliasing
8.13: List arguments
8.14: Debugging
8.E: Lists (Exercises)
8.G: Lists (Glossary)

9: DICTIONARIES

9.1: Dictionaries
9.2: Dictionary as a Set of Counters
9.3: Dictionaries and Files
9.4: Looping and Dictionaries
9.5: Advanced Text Parsing
9.6: Debugging
9.E: Dictionaries (Exercises)
9.G: Dictionaries (Glossary)

10: TUPLES

10.1: Tuples are Immutable
10.2: Comparing Tuples
10.3: Tuple Assignment
10.4: Dictionaries and Tuples
10.5: Multiple assignment with dictionaries
10.6: The most common words
10.7: Using Tuples as Keys in Dictionaries
10.8: Sequences: strings, lists, and tuples - Oh My!
10.9: Debugging
10.E: Tuples (Exercises)
10.G: Tuples (Glossary)

11: REGULAR EXPRESSIONS

11.1: Regular Expressions
11.2: Character matching in regular expressions
11.3: Extracting data using regular expressions
11.4: Combining searching and extracting
11.5: Escape Character
11.6: Bonus section for Unix / Linux users
11.7: Debugging
11.E: Regular Expressions (Exercises)
11.G: Regular Expressions (Glossary)
11.S: Regular Expressions (Summary)

12: NETWORKED PROGRAMS

While many of the examples in this book have focused on reading files and looking for data in those files, there are many different sources of information when one considers the Internet. In this chapter we will pretend to be a web browser and retrieve web pages using the HyperText Transfer Protocol (HTTP). Then we will read through the web page data and parse it.

12.1: HyperText Transfer Protocol - HTTP
12.2: The World's Simplest Web Browser
12.3: Retrieving an image over HTTP
12.4: Retrieving web pages with urllib

12.5: Parsing HTML and scraping the web
12.6: Parsing HTML using regular expressions
12.7: Parsing HTML using BeautifulSoup
12.8: Reading binary files using urllib
12.E: Networked Programs (Exercises)
12.G: Networked Programs (Glossary)

13: PYTHON AND WEB SERVICES

There are two common formats that we use when exchanging data across the web. The "eXtensible Markup Language" or XML has been in use for a very long time and is best suited for exchanging document-style data. When programs just want to exchange dictionaries, lists, or other internal information with each other, they use JavaScript Object Notation or JSON (see www.json.org). We will look at both formats.

13.1: eXtensible Markup Language - XML
13.2: Looping through Nodes
13.3: JavaScript Object Notation - JSON
13.4: Parsing JSON
13.5: Application Programming Interfaces
13.6: Google geocoding web service
13.7: Security and API usage
13.E: Python and Web Services (Exercises)
13.G: Python and Web Services (Glossary)

14: OBJECT-ORIENTED PROGRAMMING

As programs get to be millions of lines long, it becomes increasingly important to write code that is easy to understand. If you are working on a million line program, you can never keep the entire program in your mind at the same time. So we need ways to break the program into multiple smaller pieces so to solve a problem, fix a bug, or add a new feature we have less to look at.

14.1: Managing Larger Programs
14.2: Getting Started
14.3: Using Objects
14.4: Starting with Programs
14.5: Subdividing a Problem - Encapsulation
14.6: Our First Python Object
14.7: Classes as Types
14.8: Many Instances
14.8: Object Lifecycle
14.9: Inheritance
14.G: Object-Oriented Programming (Glossary)
14.S: Object-Oriented Programming (Summary)

15: USING DATABASES AND SQL

A database is a file that is organized for storing data. Most databases are organized like a dictionary in the sense that they map from keys to values. The biggest difference is that the database is on disk (or other permanent storage), so it persists after the program ends. Because a database is stored on permanent storage, it can store far more data than a dictionary, which is limited to the size of the memory in the computer.

15.1: What is a database?
15.2: Database Concepts
15.3: Database Browser for SQLite
15.4: Creating a database table
15.5: Structured Query Language summary
15.6: Spidering Twitter using a database
15.7: Basic data modeling
15.8: Programming with Multiple Tables
15.9: Constraints in Database Tables
15.10: Retrieve and/or insert a record
15.11: Storing the friend relationship
15.12: Three Kinds of Keys
15.13: Using JOIN to retrieve data
15.14: Degugging

15.G: Using Databases and SQL (Glossary)
15.S: Using Databases and SQL (Summary)

16: VISUALIZING DATA

16.1: Building a Google map from geocoded data
16.2: Visualizing Networks and Interconnections
16.3: Visualizing mail data