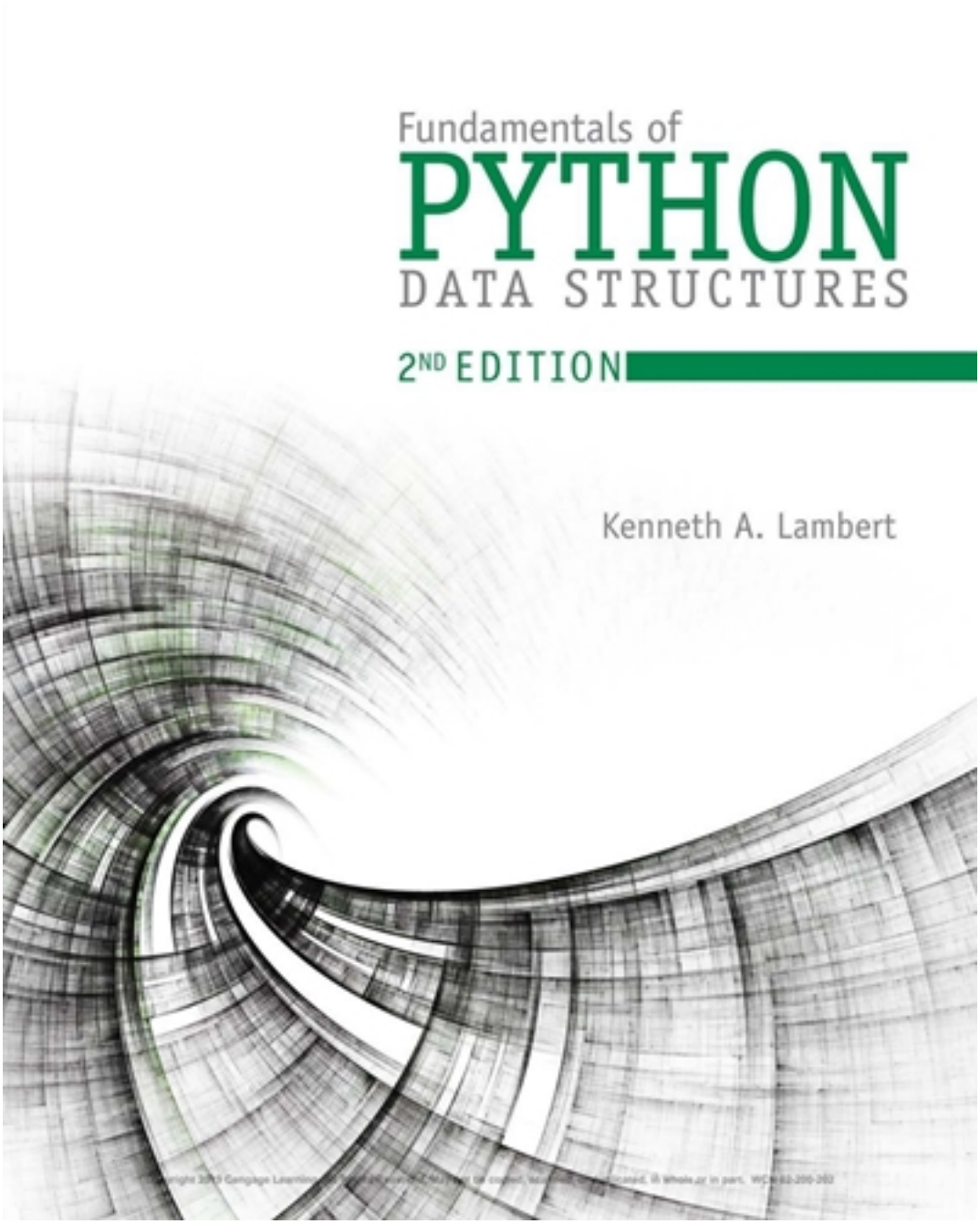
Solutions for Fundamentals of Python Data Structures 2nd Edition by Lambert

CLICK HERE TO ACCESS COMPLETE Solutions



Solutions

[ct]Answers to Review Questions for Chapter 2

1. b
2. b
3. b
4. b
5. b
6. a
7. b
8. a
9. b
10. a

Chapter 1

Basic Python Programming

At a Glance

Instructor's Manual Table of Contents

Lecture Notes

Overview

This chapter gives a quick overview of Python programming. It is intended to bring those new to or rusty in Python up to speed, but it does not pretend to be a thorough introduction to computer science or the Python programming language. For a more detailed treatment of programming in Python, see the book Fundamentals of Python: First Programs, Second Edition (Cengage Learning, 2019). For documentation on the Python programming language, visit www.python.org.

Chapter Objectives

After reading this chapter and completing the exercises, the student will be able to:

- Write a simple Python program using its basic structure
- Perform simple input and output operations
- Perform operations with numbers such as arithmetic and comparisons
- Perform operations with Boolean values
- Implement an algorithm using the basic constructs of sequences of statements, selection statements, and loops
- Define functions to structure code
- Use built-in data structures such as strings, files, lists, tuples, and dictionaries
- Define classes to represent new types of objects
- Structure programs in terms of cooperating functions, data structures, classes, and modules

Teaching Tips

Basic Program Elements

1. Point out that Python has a vast array of features and constructs but that the basic program elements are quite simple.

Programs and Modules

1. Explain to students that a module is a file of Python code and can include statements, function definitions, and class definitions.

2. Mention that a short Python program is also called a script and can be contained in one module.

3. Discuss the role of the main module and supporting modules.

An Example Python Program: Guessing a Number

1. Use the example program in the text to discuss a complete Python program that plays a game of guess-the-number with the user.

2. Point out that the sample program includes several types of Python statements that will be discussed later in the chapter.

Teaching Tip	Students can see more examples of simple Python programs by visiting: http://www.pythonforbeginners.com/code-snippets-source-code/python-code-examples

Editing, Compiling, and Running Python Programs

1. Discuss the command syntax required to run Python programs. Use the `numberguess` program as an example.

2. Introduce students to Python's IDLE (Integrated DeveLopement Environment). Show students the ways in which IDLE can be launched and then demonstrate using IDLE to edit a Python module.

Program Comments

1. Explain that a program comment is text ignored by the Python compiler but could contain valuable information to the reader of the program.

2. Point out to students what an end-of-line comment looks like. Compare this to a multiline comment, which is a string enclosed in triple single quotes (or triple double quotes). Be sure to mention that multiline comments are also called *docstrings*.

Lexical Elements

1. Define lexical elements as types of words or symbols used to construct sentences, such as:
    a. basic symbols
    b. identifiers
    c. literals
    d. operators
    e. delimiters

Spelling and Naming Conventions

1. Point out that Python keywords and names are case-sensitive and are spelled in lowercase letters and color-coded in orange in an IDLE window.

2. Explain that all Python names (other than those of built-in functions) are color-coded in black and can begin with a letter or an underscore (_).

3. Discuss the suggestion that Python programmers should use names that describe their role in a program. Point out that variable names should be nouns or adjectives and function and method names should be verbs if they denote actions.

4. Use Table 1.1 to discuss examples of Python Naming Conventions

Syntactic Elements

1. Explain that syntactic elements are the types of sentences composed from lexical elements. Point out that Python uses white spaces (spaces, tabs, or line breaks) to mark the syntax of many types of sentences.

Literals

1. Mention that numbers are written in Python as they are in other programming languages. Further mention that Boolean values True and False are keywords.

2. Explain that you can enclose strings in single quotes, double quotes, or sets of three double quotes or three single quotes. Discuss the example code in the text, which demonstrates all of the above.

3. Discuss the use of the \ character, which is used to escape nongraphic characters such as the newline (\n) and the tab (\t), or the \ character itself.

Operators and Expressions

1. Point out that arithmetic expressions use the standard operators (+, -, *, /, %).

2. Explain that the / operator produces a floating-point result, whereas the / / operator produces an integer quotient. Also mention that the + operator means concatenation when used with collections and that the * * operator is used for exponentiation.

3. Discuss the comparison operators that work with numbers and strings. Point out that the == operator compares the internal contents of data structures, such as two lists.

4. Describe the standard precedence that operators have as the following: selector, function call, subscript, arithmetic, comparison, logical, assignment.

Teaching Tip	For more information on Python operators, visit: https://www.tutorialspoint.com/python/python_basic_operators.htm

Function Calls

1. Explain that functions are called with the function's name followed by a parenthesized list of arguments.

2. List some examples of function calls. Point out that Python includes a few standard functions, such as abs and round.

The `print` Function

1. Explain that the `print` function is used to display its argument on the console and that Python automatically runs the `str` function on each argument to obtain its string representation.

2. Mention that by default, `print` terminates its output with a newline.

The input Function

1. Explain that the `input` function waits for the user to enter text at the keyboard. When the user presses the Enter key, the function returns a string containing the characters entered.

Type Conversion Functions and Mixed-Mode Operations

1. Point out to students that when a user enters a number at the keyboard, the `input` function returns a string of digits, not a numeric value. Further explain that the program must convert this string to an `int` or a `float` before numeric processing.

2. Review the code segment in the text that inputs the radius of a circle, converts the string to a float, and computes and outputs the circle's area.

Optional and Keyword Function Arguments

1. Discuss required arguments versus optional arguments. Point out that required arguments have no default values and optional arguments have default values and can appear in any order when their keywords are used.

2. Provide examples of functions that offer optional arguments and discuss.

Variables and Assignment Statements

1. Review the syntax of a simple assignment statement. Point out that assignment statements must appear on a single line of code, unless the line is broken after a comma, parenthesis, curly brace, or square bracket.

Python Data Typing

1. Remind students that in Python, any variable can name a value of any time. However, variables are not declared to have a type, they are simply assigned a value.

`import` Statements

1. Explain that import statements make visible to a program the identifiers from another module.

2. Discuss the ways to express import statements.

Getting Help on Program Components

1. Demonstrate how to use Python help by entering the function call `help(<component>)` at the shell prompt.

Control Statements

1. Introduce this section by explaining that Python includes several control statements for sequencing, conditional execution, and iteration.

Conditional Statements

1. Review conditional statements with students. Point out that the structure of Python's conditional statements is similar to that of other languages.

2. Discuss the following types of conditional statements:
   a. one-way if statement
   b. two-way if statement
   c. multiway if statement

Using `if _name_ == "_main_"`

1. Explain to students that the purpose of this statement is to allow the programmer either to run the module as a standalone program or to import it from the shell or another module.

2. Mention that the statement above is useful when developing standalone program modules because it allows the programmer to view help on the module just by importing it into the shell.

Loop Statements

1. Review the syntax of a Python `while` loop statement with your students.

2. Explain that Python includes a `for` loop statement for more concise iteration over a sequence of values. Review the syntax of the `for` statement.

3. Discuss the use of Python's `range` function, which returns an iterable sequence of integers.

Strings and Their Operations

1. Explain that a Python string is a compound object that includes other objects.

Operators

1. Introduce students to the + operator, which builds and returns a new string that contains the characters of the two operands.

2. Demonstrate the use of the subscript operator to return the character at the position specified in the string. For example:

```
a. "greater"[0]      #Returns 'g'
```

3. Introduce students to a variation of the subscript called the slice operator. Explain that programmer use this operator to obtain a substring of a string.

4. Review some of the examples of the slice operator, found on page 13 of the text.

Formatting Strings for Output

1. Define "field width" as the total number of data characters and additional spaces for a given datum in a formatted string.

2. Explain that Python includes a general formatting mechanism that allows the programmer to specify field widths for different types of data. Discuss some of the examples given in the text.

Object and Method Calls

1. Describe a method as being similar to a function. It expects arguments, performs a task, and returns a value. Mention that a method is always called on an associated object.
2. Review the syntax of a method call.

3. Discuss the examples of method calls on strings, found on page 16 of the text.

Quick Quiz 1

1. Which of the following is another name for a short Python program?
   a. method
   b. script
   c. comment
   d. import

   Answer: B

2. To create or edit a Python module, which of the following can be used?
   a. `Python3`
   b. `docstring`
   c. `idle`
   d. `edit`

   Answer: C

3. True or False: Python automatically runs the `str` function on each argument to obtain its string representation and separates each string with a space before output.
   Answer: True

4. Which of the following functions waits for the user to enter text at the keyboard?
   a. `input`
   b. `float`
   c. `print`
   d. `abs`

   Answer: A

5. What type of statement makes visible to a program the identifiers from another module?
   a. `float`
   b. `import`
   c. `print`
   d. `if`

   Answer: b

Built-In Python Collections and Their Operations

1. Introduce students to this section, which covers built-in Python collections, such as lists.

Lists

1. Describe a list as a sequence of zero or more Python objects, called items. Further explain that a list has a literal representation, which uses square brackets to enclose items separated by commas.

2. Point out that lists are mutable. Also mention that they list type includes several methods called mutators, whose purpose is to modify the structure of a list.

3. List the most commonly used list mutator methods:
   a. `append`
   b. `insert`
   c. `pop`
   d. `remove`
   e. `sort`

Tuples

1. Introduce students to tuples, which is an immutable sequence of items.

Teaching Tip	For more information about tuples, direct students to the following site: https://www.tutorialspoint.com/python/python_tuples.htm

Loops Over Sequences

1. Explain that the `for` loop is used to iterate over items in a sequence, such as a string, a list, or a tuple.

Dictionaries

1. Use the examples in the text to explain that a dictionary literal encloses the key-value entries in a set of braces.

2. Explain that you can use the subscript operator to access a value at a given key, add a value to a new key, and replace a value at a given key.

Searching for a Value

1. Introduce students to the `in` operator, which can be used to search strings, lists, tuples, or dictionaries for a given value.

2. Point out that for dictionaries, the methods `get` and `pop` can take two arguments: a key and a default value.

Pattern Matching with Collections

1. Review the code on page 19 of the text to demonstrate pattern matching.

Creating New Functions

1. Take a moment to discuss the built-in functions within Python that allow programmers to create new functions.

Function Definitions

1. Review the syntax of a Python function. Point out that the rules and conventions for spelling function names and parameter names are the same as for variable names.

2. Discuss the definition of a simple function to compute and return the square of a number, found on page 19 of the text.

3. Explain that functions can introduce new variables, also called temporary variables.

4. Mention to students that when a function does not include a `return` statement, it automatically returns the value `None` after its last statement executes.

5. Review the example on page 20 that shows an illegal function call at the beginning of a module.

Recursive Functions

1. Introduce students to a recursive function, which is a function that calls itself. Be sure to point out that the recursive function must contain at least one selection statement. Explain how this statement examines a condition called a base case to determine whether to stop or to continue with a recursive step.

2. Examine the two examples of the function `displayRange`. Point out that even though the syntax and design of the two functions are different, the same algorithmic process is executed.

3. Review the code for the recursive function that builds and returns a value, found on page 21 of the text. Also, review the code on page 22 followed by a session showing its use.

Nested Function Definitions

1. Explain that definitions of other functions may be nested within a function's sequence of statements. Review the code on pages 22-23 for an example.

Higher-Order Functions

1. Describe a higher-order function as a function that receives another function as an argument and applies it in some way.

2. Discuss Python's two built-in-higher-order functions:
   a. `map`
   b. `filter`

Creating Anonymous Functions with `lambda`

1. Review the syntax of `lambda`. Note that the expression cannot include a sequence of statements.

2. Explain that another high-order function, `functools.reduce`, boils an iterable object's items down to a single value by applying a function of two arguments to each next item and the result of the previous application.

Catching Exceptions

1. Explain that when the Python virtual machine encounters a semantic error during program execution, it raises an exception and halts the program with an error message.

2. Discuss the `try-except` statement, which allows a program to trap or catch exceptions and perform the appropriate recovery operations. Review the syntax found on page 24 of the text.

3. Review the demo program that defines a recursive function called `safeIntegerInput`. Mention that the function traps a `ValueError` exception that is raised if the user enters bad digits during input.

Files and Their Operations

1. Introduce students to this section, which examines some manipulations of text files and object files.

Text File Output

1. Explain that when data are treated as integers or floating-point numbers, they must be separated by whitespace characters (spaces, tabs, and newlines).

2. Mention that all data output to or input from a text file must be strings. Explain that numbers must be converted to strings before output and converted back to numbers after input.

3. Discuss the use of the `write` method, which is used to output data to a text file. Point out that when all outputs are finished, the file should be closed using the method `close`.

Writing Numbers to a Text File

1. Explain that the values of most data types can be converted to strings by using the str function. The resulting strings are then written to a file with a space or a newline as a separator character.

Reading Text from a Text File

1. Discuss the code for opening myfile.txt for input:
   a. `>>> f = open("myfile.txt" , 'r')`

2. Explain that the simplest way to read data from an input file is to use the file method `read` to input the entire contents of the file as a single string.

3. Point out that in cases where you might want to read a specified number of lines from a file, you can use the file method `readline`. Review the code on page 28 to see how this is accomplished.

Reading Numbers from a File

1. Discuss the use of the `int` and `float` functions to convert integers and floating-point numbers to numbers.

2. Review the code on page 28, which opens a file of random integers, reads them, and prints their sum.

3. Discuss the four steps that allow you to convert a sequence of words in the input file into a sequence of integers in order to use the `sum` function:
   a. Read the text of the file into a single string
   b. Split this string into a list of words
   c. Map the `int` function onto this list to convert the strings to integers

d. Sum the result

Reading and Writing Objects with pickle

1. Explain that Python includes a module that allows programmers to save and load objects using a process called pickling.

2. Discuss the steps a programmer would take in the "picking" process.

3. Review the code that will load objects from the file items.dat into a new list named `lyst`.

Creating New Classes

1. Remind students that a class describes the data and the methods pertaining to a set of objects by providing a blueprint for creating objects and the code to execute when methods are called on them.

2. Review the syntax of a Python class definition.

3. Point out that the parent class is optional and is assumed to be `object`. Discuss how all Python classes belong to a hierarchy.

4. Discuss the difference between instance methods and instance variables.

5. Review the code for the definition of a `Counter` class. Be sure to discuss the following:
   a. `_init_` method
   b. mutators
   c. accessors
   d. `reset` method
   e. `increment` and `decrement` methods
   f. `_str_` method
   g. `_eq_` method

6. Be sure to review the interaction with some counter objects found on page 32 of the text.

Quick Quiz 2

1. Which of the following is used in a list in order to enclose items separated by commas?
   a. < >
   b. “ “
   c. [ ]
   d. { }

   Answer: C

2. A function that calls itself is known as which of the following?
   a. default function
   b. return function
   c. display function
   d. recursive function

   Answer: D

3. Which type of function receives another function as an argument and applies it in the same way?
   a. higher-order
   b. default
   c. recursive
   d. factorial

   Answer: A

4. True or False: Definitions of other functions may not be nested within a function’s sequence of statement.
   Answer: False

5. When data are treated as integers or floating-point numbers, they must be separated by which of the following?
   a. commas
   b. periods
   c. slashes
   d. whitespace characters

   Answer: D

Class Discussion Topics

1. How does the use of functions to structure code make writing programs easier? Give an example of a function you could write that illustrates your point.

Additional Projects

1. Define a class that represents a sports team of your choosing. The class will be used for gathering statistics for the team.

Additional Resources

1. Python Input, Output, and Import Tutorial
   https://www.programiz.com/python-programming/input-output-import

2. Basic Operators in Python
   https://www.geeksforgeeks.org/basic-operators-python/

3. Decisions and Selection in Python
   http://interactivepython.org/runestone/static/pip/Selection/selection.html

4. Python Built-In Functions
   https://www.w3schools.com/python/python_ref_functions.asp

Key Terms

For definitions of key terms, see the Glossary near the end of the book.