APPENDIX D

Answers to Checkpoints

Chapter 1

1.1 A program is a set of instructions that a computer follows to perform a task.

1.2 Hardware is all the physical devices, or components, of which a computer is made.

1.3 The central processing unit (CPU), main memory, secondary storage devices, input devices, and output devices

1.4 The CPU

1.5 Main memory

1.6 Secondary storage

1.7 Input device

1.8 Output device

1.9 The operating system

1.10 A utility program

1.11 Application software

1.12 One byte

1.13 A bit

1.14 The binary numbering system

1.15 It is an encoding scheme that uses a set of 128 numeric codes to represent the English letters, various punctuation marks, and other characters. These numeric codes are used to store characters in a computer's memory. (ASCII stands for the American Standard Code for Information Interchange.)

1.16 Unicode

1.17 Digital data is data that is stored in binary, and a digital device is any device that works with binary data.

1.18 Machine language

1.19 Main memory, or RAM

1.20 The fetch-decode-execute cycle

1.21 It is an alternative to machine language. Instead of using binary numbers for instructions, assembly language uses short words that are known as mnemonics.

1.22 A high-level language

1.23 Syntax

1.24 A compiler

1.25 An interpreter

1.26 A syntax error

Chapter 2

2.1 Any person, group, or organization that is asking you to write a program

2.2 A single function that the program must perform in order to satisfy the customer

2.3 A set of well-defined logical steps that must be taken to perform a task

2.4 An informal language that has no syntax rules and is not meant to be compiled or executed. Instead, programmers use pseudocode to create models, or "mock-ups," of programs.

2.5 A diagram that graphically depicts the steps that take place in a program

2.6 Ovals are terminal symbols. Parallelograms are either output or input symbols. Rectangles are processing symbols.

2.7
```
print('Jimmy Smith')
```

2.8
```
print("Python's the best!")
```

2.9
```
print('The cat said "meow"')
```

2.10 A name that references a value in the computer's memory

2.11 `99bottles` is illegal because it begins with a number. `r&d` is illegal because the & character is not allowed.

2.12 No, it is not because variable names are case sensitive.

2.13 It is invalid because the variable that is receiving the assignment (in this case `amount`) must appear on the left side of the = operator.

2.14
```
The value is val.
```

2.15 `value1` will reference an `int`. `value2` will reference a `float`. `value3` will reference a `float`. `value4` will reference an `int`. `value5` will reference an `str` (string).

2.16 0

2.17
```
last_name = input("Enter the customer's last name: ")
```

2.18
```
sales = float(input('Enter the sales for the week: '))
```

2.19 Here is the completed table:

Expression	Value
`6 + 3 * 5`	21
`12 / 2 - 4`	2
`9 + 14 * 2 - 6`	31
`(6 + 2) * 3`	24
`14 / (11 - 4)`	2
`9 + 12 * (8 - 3)`	69

2.20 4

2.21 1

Chapter 3

3.1 A logical design that controls the order in which a set of statements execute

3.2 It is a program structure that can execute a set of statements only under certain circumstances.

3.3 A decision structure that provides a single alternative path of execution. If the condition that is being tested is true, the program takes the alternative path.

3.4 An expression that can be evaluated as either true or false

3.5 You can determine whether one value is greater than, less than, greater than or equal to, less than or equal to, equal to, or not equal to another value.

3.6
```
if y == 20:
x = 0
```

3.7
```
if sales >= 10000:
   commissionRate = 0.2
```

3.8 A dual alternative decision structure has two possible paths of execution; one path is taken if a condition is true, and the other path is taken if the condition is false.

3.9 `if-else`

3.10 When the condition is false

3.11 z is not less than a.

3.12 Boston
New York

3.13
```
if number == 1:
    print('One')
elif number == 2:
    print('Two')
elif number == 3:
    print('Three')
else:
    print('Unknown')
```

3.14 It is an expression that is created by using a logical operator to combine two Boolean subexpressions.

3.15 F
T
F
F
T
T
T
F
F
T

3.16 T
F
T
T
T

3.17 The `and` operator: If the expression on the left side of the `and` operator is false, the expression on the right side will not be checked.
The `or` operator: If the expression on the left side of the `or` operator is true, the expression on the right side will not be checked.

3.18
```
if speed >= 0 and speed <= 200:
    print('The number is valid')
```

3.19
```
if speed < 0 or speed > 200:
    print('The number is not valid')
```

3.20 `True` or `False`

3.21 A variable that signals when some condition exists in the program

Chapter 4

4.1 A structure that causes a section of code to repeat

4.2 A loop that uses a true/false condition to control the number of times that it repeats

4.3 A loop that repeats a specific number of times

4.4 An execution of the statements in the body of the loop

4.5 Before

4.6 None. The condition `count < 0` will be false to begin with.

4.7 A loop that has no way of stopping and repeats until the program is interrupted.

4.8
```
for x in range(6):
    print('I love to program!')
```

4.9
```
0
1
2
3
4
5
```

4.10
```
2
3
4
5
```

4.11
```
0
100
200
300
400
500
```

4.12
```
10
9
8
7
6
```

4.13 A variable that is used to accumulate the total of a series of numbers

4.14 Yes, it should be initialized with the value 0. This is because values are added to the accumulator by a loop. If the accumulator does not start at the value 0, it will not contain the correct total of the numbers that were added to it when the loop ends.

4.15
```
15
```

4.16
```
15
5
```

4.17
a) `quantity += 1`
b) `days_left -= 5`
c) `price *= 10`
d) `price /= 2`

4.18 A sentinel is a special value that marks the end of a list of items.

4.19 A sentinel value must be unique enough that it will not be mistaken as a regular value in the list.

4.20 It means that if bad data (garbage) is provided as input to a program, the program will produce bad data (garbage) as output.

4.21 When input is given to a program, it should be inspected before it is processed. If the input is invalid, then it should be discarded and the user should be prompted to enter the correct data.

4.22 The input is read, and then a pretest loop is executed. If the input data is invalid, the body of the loop executes. In the body of the loop, an error message is displayed so the user will know that the input was invalid, and then the input read again. The loop repeats as long as the input is invalid.

4.23 It is the input operation that takes place just before an input validation loop. The purpose of the priming read is to get the first input value.

4.24 None

Chapter 5

5.1 A function is a group of statements that exist within a program for the purpose of performing a specific task.

5.2 A large task is divided into several smaller tasks that are easily performed.

5.3 If a specific operation is performed in several places in a program, a function can be written once to perform that operation and then be executed any time it is needed.

5.4 Functions can be written for the common tasks that are needed by the different programs. Those functions can then be incorporated into each program that needs them.

5.5 When a program is developed as a set of functions in which each performs an individual task, then different programmers can be assigned the job of writing different functions.

5.6 A function definition has two parts: a header and a block. The header indicates the starting point of the function, and the block is a list of statements that belong to the function.

5.7 To call a function means to execute the function.

5.8 When the end of the function is reached, the computer returns back to the part of the program that called the function, and the program resumes execution at that point.

5.9 Because the Python interpreter uses the indentation to determine where a block begins and ends

5.10 A local variable is a variable that is declared inside a function. It belongs to the function in which it is declared, and only statements in the same function can access it.

5.11 The part of a program in which a variable may be accessed

5.12 Yes, it is permissible.

5.13 Arguments

5.14 Parameters

5.15 A parameter variable's scope is the entire function in which the parameter is declared.

5.16 No, it does not.

5.17 a. passes by keyword argument
b. passes by position

5.18 The entire program

5.19 Here are three:

- Global variables make debugging difficult. Any statement in a program can change the value of a global variable. If you find that the wrong value is being stored in a global variable, you have to track down every statement that accesses it to determine where the bad value is coming from. In a program with thousands of lines of code, this can be difficult.
- Functions that use global variables are usually dependent on those variables. If you want to use such a function in a different program, you will most likely have to redesign it so it does not rely on the global variable.
- Global variables make a program hard to understand. A global variable can be modified by any statement in the program. If you are to understand any part of the program that uses a global variable, you have to be aware of all the other parts of the program that access the global variable.

5.20 A global constant is a name that is available to every function in the program. It is permissible to use global constants. Because their value cannot be changed during the program's execution, you do not have to worry about its value being altered.

5.21 The difference is that a value returning function returns a value back to the statement that called it. A simple function does not return a value.

5.22 A prewritten function that performs some commonly needed task

5.23 The term "black box" is used to describe any mechanism that accepts input, performs some operation (that cannot be seen) using the input, and produces output.

5.24 It assigns a random integer in the range of 1 through 100 to the variable `x`.

5.25 It prints a random integer in the range of 1 through 20.

5.26 It prints a random integer in the range of 10 through 19.

5.27 It prints a random floating-point number in the range of 0.0 up to, but not including, 1.0.

5.28 It prints a random floating-point number in the range of 0.1 through 0.5.

5.29 It uses the system time, retrieved from the computer's internal clock.

5.30 If the same seed value were always used, the random number functions would always generate the same series of pseudorandom numbers.

5.31 It returns a value back to the part of the program that called it.

5.32 a) `do_something`
b) It returns a value that is twice the argument passed to it.
c) 20

5.33 A function that returns either `True` or `False`

5.34
```
import math
```

5.35
```
square_root = math.sqrt(100)
```

5.36
```
angle = math.radians(45)
```

Chapter 6

6.1 A file to which a program writes data. It is called an output file because the program sends output to it.

6.2 A file from which a program reads data. It is called an input file because the program receives input from it.

6.3 (1) Open the file. (2) Process the file. (3) Close the file.

6.4 Text and binary. A text file contains data that has been encoded as text using a scheme such as ASCII. Even if the file contains numbers, those numbers are stored in the file as a series of characters. As a result, the file may be opened and viewed in a text editor such as Notepad. A binary file contains data that has not been converted to text. As a consequence, you cannot view the contents of a binary file with a text editor.

6.5 Sequential and direct access. When you work with a sequential access file, you access data from the beginning of the file to the end of the file. When you work with a direct access file, you can jump directly to any piece of data in the file without reading the data that comes before it.

6.6 The file's name on the disk and the name of a variable that references a file object.

6.7 The file's contents are erased.

6.8 Opening a file creates a connection between the file and the program. It also creates an association between the file and a file object.

6.9 Closing a file disconnects the program from the file.

6.10 A file's read position marks the location of the next item that will be read from the file. When an input file is opened, its read position is initially set to the first item in the file.

6.11 You open the file in append mode. When you write data to a file in append mode, the data is written to the end of the file's existing contents.

6.12
```
outfile = open('numbers.txt', 'w')
for num in range(1, 11):
    outfile.write(str(num) + '\n')
outfile.close()
```

6.13 The `readline` method returns an empty string (`''`) when it has attempted to read beyond the end of a file.

6.14
```
infile = open('numbers.txt', 'r')
line = infile.readline()
while line != '':
    print(line)
    line = infile.readline()
infile.close()
```

6.15
```
infile = open('data.txt', 'r')
for line in infile:
    print(line)
infile.close()
```

6.16 A record is a complete set of data that describes one item, and a field is a single piece of data within a record.

6.17 You copy all the original file's records to the temporary file, but when you get to the record that is to be modified, you do not write its old contents to the temporary file. Instead, you write its new, modified values to the temporary file. Then, you finish copying any remaining records from the original file to the temporary file.

6.18 You copy all the original file's records to the temporary file, except for the record that is to be deleted. The temporary file then takes the place of the original file. You delete the original file and rename the temporary file, giving it the name that the original file had on the computer's disk.

6.19 An exception is an error that occurs while a program is running. In most cases, an exception causes a program to abruptly halt.

6.20 The program halts.

6.21 `IOError`

6.22 `ValueError`

Chapter 7

7.1 `[1, 2, 99, 4, 5]`

7.2 `[0, 1, 2]`

7.3 `[10, 10, 10, 10, 10]`

7.4
```
1
3
5
7
9
```

7.5
```
4
```

7.6 Use the built-in `len` function.

7.7
```
[1, 2, 3]
[10, 20, 30]
[1, 2, 3, 10, 20, 30]
```

7.8
```
[1, 2, 3]
[10, 20, 30, 1, 2, 3]
```

7.9
```
[2, 3]
```

7.10
```
[2, 3]
```

7.11
```
[1]
```

7.12
```
[1, 2, 3, 4, 5]
```

7.13
```
[3, 4, 5]
```

7.14
```
Jasmine's family:
['Jim', 'Jill', 'John', 'Jasmine']
```

7.15 The `remove` method searches for and removes an element containing a specific value. The `del` statement removes an element at a specific index.

7.16 You can use the built-in `min` and `max` functions.

7.17 You would use statement b, `names.append('Wendy')`. This is because element 0 does not exist. If you try to use statement a, an error will occur.

7.18 a) The `index` method searches for an item in the list and returns the index of the first element containing that item.
b) The `insert` method inserts an item into the list at a specified index.
c) The `sort` method sorts the items in the list to appear in ascending order.
d) The `reverse` method reverses the order of the items in the list.

7.19 The list contains 4 rows and 2 columns.

7.20
```
mylist = [[0, 0, 0, 0], [0, 0, 0, 0],
          [0, 0, 0, 0], [0, 0, 0, 0]]
```

7.21
```
for r in range(4):
    for c in range(2):
        print(numbers[r][c])
```

7.22 The primary difference between tuples and lists is that tuples are immutable. That means that once a tuple is created, it cannot be changed.

7.23 Here are three reasons:
- Processing a tuple is faster than processing a list, so tuples are good choices when you are processing lots of data and that data will not be modified.
- Tuples are safe. Because you are not allowed to change the contents of a tuple, you can store data in one and rest assured that it will not be modified (accidentally or otherwise) by any code in your program.
- There are certain operations in Python that require the use of a tuple.

7.24
```
my_tuple = tuple(my_list)
```

7.25
```
my_list = list(my_tuple)
```

Chapter 8

8.1
```
for letter in name:
    print(letter)
```

8.2 0

8.3 9

8.4 An `IndexError` exception will occur if you try to use an index that is out of range for a particular string.

8.5 Use the built-in `len` function.

8.6 The second statement attempts to assign a value to an individual character in the string. Strings are immutable, however, so the expression `animal[0]` cannot appear on the left side of an assignment operator.

8.7
```
cde
```

8.8
```
defg
```

8.9
```
abc
```

8.10
```
abcdefg
```

8.11
```
if 'd' in mystring:
    print('Yes, it is there.')
```

8.12
```
little = big.upper()
```

8.13
```
if ch.isdigit():
    print('Digit')
else:
    print('No digit')
```

8.14
```
a A
```

8.15
```
again = input('Do you want to repeat ' + \
              'the program or quit? (R/Q) ')
while again.upper() != 'R' and again.upper() != 'Q':
    again = input('Do you want to repeat the ' +
                  'program or quit? (R/Q) ')
```

8.16
```
$
```

8.17
```
for letter in mystring:
    if letter.isupper():
        count += 1
```

8.18
```
my_list = days.split()
```

8.19
```
my_list = values.split('$')
```

Chapter 9

9.1 Key and value

9.2 The key

9.3 The string `'start'` is the key, and the integer 1472 is the value.

9.4 It stores the key-value pair `'id' : 54321` in the `employee` dictionary.

9.5 `ccc`

9.6 You can use the `in` operator to test for a specific key.

9.7 It deletes the element that has the key 654.

9.8 `3`

9.9
```
1
2
3
```

9.10 The `pop` method accepts a key as an argument, returns the value that is associated with that key, and removes that key-value pair from the dictionary. The `popitem` method returns a randomly selected key-value pair, as a tuple, and removes that key-value pair from the dictionary.

9.11 It returns all a dictionary's keys and their associated values as a sequence of tuples.

9.12 It returns all the keys in a dictionary as a sequence of tuples.

9.13 It returns all the values in the dictionary as a sequence of tuples.

9.14 Unordered

9.15 No

9.16 You call the built-in `set` function.

9.17 The set will contain these elements (in no particular order): `'J'`, `'u'`, `'p'`, `'i'`, `'t'`, `'e'`, and `'r'`.

9.18 The set will contain one element: 25.

9.19 The set will contain these elements (in no particular order): `'w'`, `' '`, `'x'`, `'y'`, and `'z'`.

9.20 The set will contain these elements (in no particular order): 1, 2, 3, and 4.

9.21 The set will contain these elements (in no particular order): `'www'`, `'xxx'`, `'yyy'`, and `'zzz'`.

9.22 You pass the set as an argument to the `len` function.

9.23 The set will contain these elements (in no particular order): 10, 9, 8, 1, 2, and 3.

9.24 The set will contain these elements (in no particular order): 10, 9, 8, `'a'`, `'b'`, and `'c'`.

9.25 If the specified element to delete is not in the set, the `remove` method raises a `KeyError` exception, but the `discard` method does not raise an exception.

9.26 You can use the `in` operator to test for the element.

9.27 `{10, 20, 30, 100, 200, 300}`

9.28 `{3, 4}`

9.29 `{1, 2}`

9.30 `{5, 6}`

9.31 `{'a', 'd'}`

9.32 `set2` is a subset of `set1`, and `set1` is a superset of `set2`.

9.33 The process of converting the object to a stream of bytes that can be saved to a file for later retrieval.

9.34 `'wb'`

9.35 `'rb'`

9.36 The `pickle` module

9.37 `pickle.dump`

9.38 `pickle.load`

Chapter 10

10.1 An object is a software entity that contains both data and procedures.

10.2 Encapsulation is the combining of data and code into a single object.

10.3 When an object's internal data is hidden from outside code and access to that data is restricted to the object's methods, the data is protected from accidental corruption. In addition, the programming code outside the object does not need to know about the format or internal structure of the object's data.

10.4 Public methods can be accessed by entities outside the object. Private methods cannot be accessed by entities outside the object. They are designed to be accessed internally.

10.5 The metaphor of a blueprint represents a class.

10.6 Objects are the cookies.

10.7 Its purpose is to initialize an object's data attributes. It executes immediately after the object is created.

10.8 When a method executes, it must have a way of knowing which object's data attributes it is supposed to operate on. That's where the `self` parameter comes in. When a method is called, Python automatically makes its `self` parameter reference the specific object that the method is supposed to operate on.

10.9 By starting the attribute's name with two underscores

10.10 It returns a string representation of the object.

10.11 By passing the object to the built-in `str` method

10.12 An attribute that belongs to a specific instance of a class

10.13 10

10.14 A method that returns a value from a class's attribute but does not change it is known as an accessor method. A method that stores a value in a data attribute or changes the value of a data attribute in some other way is known as a mutator method.

10.15 The top section is where you write the name of the class. The middle section holds a list of the class's fields. The bottom section holds a list of the class's methods.

10.16 A written description of the real-world objects, parties, and major events related to the problem

10.17 If you adequately understand the nature of the problem you are trying to solve, you can write a description of the problem domain yourself. If you do not thoroughly understand the nature of the problem, you should have an expert write the description for you.

10.18 First, identify the nouns, pronouns, and pronoun phrases in the problem domain description. Then, refine the list to eliminate duplicates, items that you do not need to be concerned with in the problem, items that represent objects instead of classes, and items that represent simple values that can be stored in variables.

10.19 The things that the class is responsible for knowing and the actions that the class is responsible for doing

10.20 In the context of this problem, what must the class know? What must the class do?

10.21 No, not always

Chapter 11

11.1 A superclass is a general class, and a subclass is a specialized class.

11.2 When one object is a specialized version of another object, there is an "is a" relationship between them. The specialized object "is a" version of the general object.

11.3 It inherits all the superclass's attributes.

11.4 `Bird` is the superclass, and `Canary` is the subclass.

11.5 I'm a vegetable.
I'm a potato.

Chapter 12

12.1 A recursive algorithm requires multiple method calls. Each method call requires several actions to be performed by the JVM. These actions include allocating memory for parameters and local variables and storing the address of the program location where control returns after the method terminates. All these actions are known as overhead. In an iterative algorithm, which uses a loop, such overhead is unnecessary.

12.2 A case in which the problem can be solved without recursion

12.3 Cases in which the problem is solved using recursion

12.4 When it reaches the base case

12.5 In direct recursion, a recursive method calls itself. In indirect recursion, method A calls method B, which in turn calls method A.

Chapter 13

13.1 The part of a computer and its operating system with which the user interacts

13.2 A command line interface typically displays a prompt, and the user types a command, which is then executed.

13.3 The program

13.4 A program that responds to events that take place, such as the user clicking a button

13.5 a) `Label`—An area that displays one line of text or an image
b) `Entry`—An area in which the user may type a single line of input from the keyboard
c) `Button`—A button that can cause an action to occur when it is clicked
d) `Frame`—A container that can hold other widgets

13.6 You create an instance of the `tkinter` module's `Tk` class.

13.7 This function runs like an infinite loop until you close the main window.

13.8 The `pack` method arranges a widget in its proper position, and it makes the widget visible when the main window is displayed.

13.9 One will be stacked on top of the other.

13.10 `side=—left—`

13.11 You use an `Entry` widget's `get` method to retrieve the data that the user has typed into the widget.

13.12 It is a string.

13.13 `tkinter`

13.14 Any value that is stored in the `StringVar` object will automatically be displayed in the `Label` widget.

13.15 You would use radio buttons.

13.16 You would use check buttons.

13.17 When you create a group of `Radiobuttons`, you associate them all with the same `IntVar` object. You also assign a unique integer value to each `Radiobutton` widget. When one of the `Radiobutton` widgets is selected, it stores its unique integer value in the `IntVar` object.

13.18 You associate a different `IntVar` object with each `Checkbutton`. When a `Checkbutton` is selected, its associated `IntVar` object will hold the value 1. When a `Checkbutton` is deselected, its associated `IntVar` object will hold the value 0.