Starting Out with Python, 5th Edition

Answers to Review Questions

Chapter 1

Multiple Choice

1. b
2. a
3. d
4. b
5. c
6. a
7. c
8. b
9. a
10. a
11. d
12. b
13. c
14. b
15. c
16. a
17. b
18. d
19. b
20. b
21 c
22. a
23. d
24. a
25. b

True or False

1. False
2. True
3. True
4. False
5. True
6. False
7. True
8. False
9. False
10. False

Short Answer

1. Because without it, the computer could not run software.

2. A bit that is turned on represents 1, and a bit that is turned off represents 0.

3. A digital device

4. Keywords

5. Mnemonics

6. A compiler is a program that translates a high-level language program into a separate machine language program. The machine language program can then be executed any time it is needed. An interpreter is a program that both translates and executes the instructions in a high-level language program. As the interpreter reads each individual instruction in the program, it converts it to a machine language instruction and then immediately executes it. Because interpreters combine translation and execution, they typically do not create separate machine language programs.

7. Operating system

Exercises

1. *No solution -- This is a hands-on exercise to help you learn how to work with the Python interpreter in interactive mode.*

2. *No solution -- This is a hands-on exercise to help you learn how to work with the IDLE programming environment.*

3.

Decimal	Binary
11	1011
65	1000001
100	1100100
255	11111111

4.

Binary	Decimal
1101	13
1000	8
101011	43

5. Here is an example: The ASCII codes for the name Marty are:

```
M =  77
a =  97
r = 114
t = 226
y = 121
```

6.

- Guido van Rossum is the creator of the Python programming language.
- Python was created in the late 1980s.
- Benevolent Dictator for Life

Chapter 2

Multiple Choice

1. c
2. b
3. d
4. b
5. a
6. c
7. a
8. b
9. d
10. a
11. b
12. d
13. b
14. a
15. a
16. c
17. a
18. b
19. a
20. b
21. b
22. b

True or False

1. False
2. True
3. False
4. True
5. False

Short Answer

1. Interview the customer

2. 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.

3. (1) Input is received.
   (2) Some process is performed on the input.
   (3) Output is produced.

4. `float`

5. Floating point division returns a floating point number that may include fractions. Integer division returns an integer and ignores any fractional part of the division result.

6. A magic number is an unexplained value that appears in a program's code. Magic numbers can be problematic, for a number of reasons. First, it can be difficult for someone reading the code to

determine the purpose of the number. Second, if the magic number is used in multiple places in the program, it can take painstaking effort to change the number in each location, should the need arise. Third, you take the risk of making a typographical mistake each time you type the magic number in the program's code.

7. The named constant makes the program more self-explanatory. In a math statement, it is evident that `PI` represents the value of pi. Another advantage to using the named constant is that widespread changes can easily be made to the program. Let's say the value of pi appears in several different statements throughout the program. If you need to change the number of decimal places of precision used with the number, the initialization value in the declaration of the named constant is the only value that needs to be modified. For example, to use only two decimal places of precision, the declaration can be changed to:

```
PI = 3.14
```

The new value of 3.14 will then be used in each statement that includes the `PI` constant. Another advantage to using the named constant is that it helps to prevent the typographical errors that are common when using magic numbers. For example, if you accidentally type 31.4159 instead of 3.14159 in a math statement, the program will calculate the wrong value. However, if you misspell `PI`, the Python interpreter will display a message indicating that the name is not defined.

Algorithm Workbench

1.
```
height = int(input('Enter your height: '))
```

2.
```
color = input('Enter your favorite color: ')
```

3.
a.
```
b = a + 2
```
b.
```
a = b * 4
```
c.
```
b = a / 3.14
```
d.
```
a = b - 8
```

4.
a. 12
b. 4
c. 2
d. 6
e. 2

5.
```
total = 10 + 14
```

6.
```
due = total - down_payment
```

7.
```
total = subtotal * 0.15
```

8. 11

9. 5

10.
```
print(f'{sales:.2f}')
```


11.
```
print(f'{number:,.1f}')
```

12.
```
George@John@Paul@Ringo
```

13.
```
turtle.circle(75)
```

14.
```
turtle.fillcolor('blue')
turtle.begin_fill()
turtle.forward(100)
turtle.left(90)
turtle.forward(100)
turtle.left(90)
turtle.forward(100)
turtle.left(90)
turtle.forward(100)
turtle.end_fill()
```

15.
```
turtle.forward(100)
turtle.left(90)
turtle.forward(100)
turtle.left(90)
turtle.forward(100)
turtle.left(90)
turtle.forward(100)
turtle.penup()
turtle.left(90)
turtle.forward(50)
turtle.right(90)
turtle.forward(30)
turtle.setheading(0)
turtle.pendown()
turtle.fillcolor('red')
turtle.begin_fill()
turtle.circle(80)
turtle.end_fill()
```

Chapter 3

Multiple Choice

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

12. a

True or False

1. False
2. False
3. False
4. True
5. True

Short Answer

1. A conditionally executed statement is performed only when a certain condition is true.

2. A dual alternative decision structure

3. The `and` operator connects two Boolean expressions into one compound expression. Both subexpressions must be true for the compound expression to be true.

4. The `or` operator connects two Boolean expressions into one compound expression. One or both sub expressions must be true for the compound expression to be true. It is only necessary for one of the subexpressions to be true, and it does not matter which.

5. The `and` operator.

6. A flag is a variable that signals when some condition exists in the program. When the flag variable is set to `False`, it indicates the condition does not exist. When the flag variable is set to `True`, it means the condition does exist.

Algorithm Workbench

1.
```
if x > 100:
    y = 20
    z = 40
```

2.
```
if a < 10:
    b = 0
    c = 1
```

3.
```
if a < 10:
    b = 0 else:
    b = 99
```

4.
```
if score >= A_score:
    print('Your grade is A.')
else:
    if score >= B_score:
        print('Your grade is B.')
    else:
        if score >= C_score:
            print('Your grade is C.')
        else:
```



```
                    if score >= D_score:
                         print('Your grade is D.')
                    else:
                         print('Your grade is F.')
```

5.
```
if amount1 > 10 and amount2 < 100:
     if amount1 > amount2:
          print (amount1)
     elif amount2 > amount1:
          print (amount2)
     else:
          print('Both values are the same.')
```

6.
```
if speed >= 24 and speed <= 56:
     print ('Speed is normal.')
else:
     print ('Speed is abnormal.')
```

7.
```
if points < 9 or points > 51:
     print ('Invalid points')
else:
    print ('Valid points')
```

8.
```
if turtle.heading() >= 0 and turtle.heading() <= 45:
     turtle.penup()
```

9.
```
if turtle.pencolor() == 'red' or turtle.pencolor() == 'blue':
     turtle.pensize(5)
```

10.
```
if turtle.xcor() > 100 and turtle.xcor() < 200 and
   turtle.ycor() > 100 and turtle.ycor() < 200:
     turtle.hideturtle()
```

Chapter 4

Multiple Choice

1. b
2. d
3. d
4. a
5. c
6. b
7. d
8. a
9. b
10. c
11. d
12. a

True or False

1. False
2. True
3. True
4. False
5. True
6. False
7. False

Short Answer

1. A condition-controlled loop uses a true/false condition to control the number of times that it repeats.

2. A count-controlled loop repeats a specific number of times.

3. An *infinite loop* continues to repeat until the program is interrupted. Infinite loops usually occur when the programmer forgets to write code inside the loop that makes the test condition false. Here is an example of Python code that contains an infinite loop:

```
x = 99
while x > 0:
    print (x)
```

4. If the accumulator starts with any value other than 0, it will not contain the correct total when the loop finishes.

5. You can write a loop that processes a list of data items even though you do not know the number of data items in the list, and without requiring the user to know the number of items in the list in advance.

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

7. This saying, sometimes abbreviated as GIGO, refers to the fact that computers cannot tell the difference between good data and bad data. If a user provides bad data as input to a program, the program will process that bad data and, as a result, will produce bad data as output.

8. When input is given to a program, it should be inspected before it is processed. If the input is invalid, the program should discard it and prompt the user to enter the correct data.

   Specifically, the input is read, and then a loop is executed. If the input data is bad, the loop executes its block of statements. The loop displays an error message so the user will know that the input was invalid, and then it reads the new input. The loop repeats as long as the input is bad.

Algorithm Workbench

1.
```
product = 0
while product < 100:
    number = int(input('Enter a number: '))
    product = number * 10
```

2.
```
again = 'y'
while again == 'y':
```



```
    num1 = float(input('Enter a number: '))
    num2 = float(input('Enter another number: '))
    sum = num1 + num2
    print ('The sum of the numbers you entered is', sum)
    again = input('Do you want to do that again? (y/n): ')
```

3.
```
for number in range(0, 1001, 10):
    print(number)
```

4.
```
total = 0.0
    for counter in range(10):
        number = float(input('Enter a number: '))
        total += number
    print ('The total is', total)
```

5.
```
denominator = 30
total = 0
for numerator in range(1, 31):
    value = numerator / denominator
    total = total + value
    denominator -= 1
print (total)
```

6.
a. `x += 1`
b. `x *= 2`
c. `x /= 10`
d. `x -= 100`

7.
```
for row in range(10):
    for column in range(15):
        print('#', end='')
    print()
```

8.
```
number = float(input('Enter a positive nonzero number: '))
while number <= 0:
    print('That is an invalid value.')
    number = float(input('Enter a positive nonzero number: '))
print ('Thanks!')
```

9.
```
number = int(input('Enter a number between 1 and 100: '))
while number < 1 or number>100:
    print('That is an invalid value.')
    number = int(input('Enter a number between 1 and 100: '))
print ('Thanks!')
```

Chapter 5

Multiple Choice

1. c
2. a
3. d
4. b
5. a
6. d
7. b
8. b
9. c
10. a
11. b
12. d
13. b
14. b
15. b
16. a
17. d
18. d
19. b
20. c
21. c

True or False

1. False
2. True
3. False
4. False
5. False
6. True
7. False
8. False
9. True
10. False
11. True
12. False
13. True
14. True
15. True

Short Answer

1. Functions can reduce the duplication of code within a program. 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. This is known as code reuse because you are writing the code to perform a task once and then reusing it each time you need to perform the task.

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

