# Chapter 2: Model Solutions

Below, you'll find sample solutions to the lab exercises in the book.

## Lab Exercises 2.1

1. Syntax in programming refers to the set of rules that define the structure and composition of a programming language. It specifies how statements, expressions, and other constructs should be written in order to be considered valid and executable. Adhering to syntax rules is important because they ensure that the code is correctly interpreted by the compiler or interpreter. Violating syntax rules can lead to syntax errors, which prevent the code from running or cause unexpected behavior.
2. Reserved words or keywords in Python are words that are part of the language's syntax and have special meanings. They are reserved for specific purposes and cannot be used as variable names or identifiers. The reason for this restriction is to maintain the consistency and clarity of the language. By reserving certain words, Python ensures that they are always interpreted in a specific way, avoiding potential conflicts with user-defined names.
3. Three reserved words in Python and their purposes are:
    - if: Used to define conditional statements. It allows the execution of a block of code only if a certain condition is satisfied.
    - for: Used to create loops. It iterates over a sequence of items and executes a block of code for each item.
    - def: Used to define functions. It allows the creation of reusable blocks of code that can be called from other parts of the program.
4. Python uses indentation to define code blocks. Indentation refers to the spaces or tabs placed at the beginning of lines to indicate the hierarchy and grouping of statements. Code blocks are formed by lines with the same level of indentation. Consistent indentation is important because it is the primary means of structuring code in Python. Inconsistencies in indentation can lead to syntax errors or alter the intended logic of the program.
5. Code snippet that demonstrates the use of indentation

```
def greet(name):
    if name == "Alice":
        print("Hello, Alice!")
    else:
        print("Hello, stranger!")
greet("Bob")
```

6. Comments in programming are non-executable lines used to provide explanations, documentation, or annotations within the code. They are ignored by the interpreter or compiler and are meant for human readers. Comments enhance code clarity and readability by providing additional context, explanations of complex logic, or reminders for future modifications.
7. Comments facilitate code maintenance and collaboration in a team setting by providing information that helps other developers understand the code. They can explain the purpose of functions, describe input/output formats, document algorithms, or provide warnings and

TODOs for future improvements. Comments also allow team members to communicate about specific code sections, discuss potential issues, or leave feedback.

8. During development or debugging, it is common to temporarily disable or comment out blocks of code. This is done to exclude certain code segments from execution without deleting them. Commented-out code can be useful for testing alternative approaches or temporarily removing code that might be causing errors. However, commented-out code should be avoided in production codebases because it can make the code harder to understand and maintain.

## Lab Exercises 2.2

1. The input() function is used to get data from the user. It prompts the user with a message, waits for input, and returns the entered data as a string.
2. To print data to the console in Python, you can use the print() function.
3. Ask the user for their name and prints a welcome message

```
name = input("Enter your name: ")
print("Welcome,", name, "!")
```

## Lab Exercises 2.3

1. Variables in programming are used to store and manipulate data. They act as named containers that hold values in computer memory. Variables play a crucial role in programming by allowing us to store and retrieve data, perform calculations, make decisions based on conditions, and create flexible and dynamic programs.
2. Integers and floats are two different numeric data types:
    a. Integers (int) are whole numbers without decimal points. For example: 42, -10, 0.
    b. Floats (float) represent numbers with decimal points or fractional parts. For example: 3.14, -2.5, 0.0.
3. Strings in programming are sequences of characters enclosed in single quotes ('') or double quotes (""). They are commonly used to represent text and are essential for working with textual data. Strings can contain letters, numbers, symbols, and whitespace.

```
my_string = "Hello, World!"
print(my_string)              # Output: Hello, World!
```

4. String concatenation is the process of combining two or more strings into a single string. In Python, concatenation can be achieved using the + operator.

```
str1 = "Hello"
str2 = "World"
result = str1 + " " + str2
print(result)                 # Output: Hello World
```

5. Individual characters within a string can be accessed using indexing. Indexing in Python starts from 0, where the first character has an index of 0, the second character has an index of 1, and so on. Negative indexing starts from -1, where the last character has an index of -1, the second-to-last character has an index of -2, and so forth.

```
my_string = "Hello"
print(my_string[0])   # Output: H
print(my_string[3])   # Output: l
print(my_string[-1])  # Output: o
```

6. String slicing in Python allows you to extract a portion of a string by specifying a range of indices. It is done using the slicing operator [:], where the start index is inclusive, and the end index is exclusive.

```
my_string = "Python Programming"
sliced_string = my_string[7:18]
print(sliced_string)              # Output: Programming
```

7. String formatting in Python is a way to create formatted output by substituting placeholders with values using the .format() method.

```
name = "Alice"
age = 25
message = "My name is {} and I'm {} years old."
                                .format(name, age)
print(message)
```

8. Three commonly used string methods in Python are:

```
my_string = "hello"
print(my_string.upper())               # Output: HELLO

my_string = "banana"
print(my_string.replace("a", "e"))     # Output: benene

my_string = "Hello, World!"
print(my_string.startswith("Hello"))  # Output: True
```

9. The boolean data type in Python represents logical values. It has two possible values: True and False. Booleans are used for making logical comparisons, conditional statements, and determining the truthfulness or falseness of an expression.
10. age = 27
11. height = 1.75
12. name = "John Doe"
13. Concatenating the "name" and "age"

```
message = "My name is " + name + " and I am " + str(age) + "
           years old."
print(message)
```

14. Accessing the third character

```
my_string = "Hello, World!"
third_character = my_string[2]
print(third_character)
```

15. Slicing the string "Python Programming"

```
my_string = "Python Programming"
sliced_string = my_string[7:]
print(sliced_string)
```

16. Using string formatting to create a message that includes the values of the variables "name" and "height":

```
message = "My name is {} and my height is {}
                                  meters.".format(name, height)
print(message)
```

17. Using the .upper() method to convert the string "hello" to uppercase

```
my_string = "hello"
uppercase_string = my_string.upper()
print(uppercase_string)
```

18. Using the .replace() method

```
my_string = "banana"
replaced_string = my_string.replace("a", "e")
print(replaced_string)
```

19. Checking if the string "Hello, World!" starts with the substring "Hello"

```
my_string = "Hello, World!"
starts_with_hello = my_string.startswith("Hello")
```

```
print(starts_with_hello)
```

20. is_raining = False

## Lab Exercises 2.4

1. Arithmetic operators perform mathematical operations, comparison operators compare values, and assignment operators assign values to variables. Examples of these operators are:
    - Arithmetic operators: + (addition), - (subtraction), * (multiplication), / (division), % (modulus), ** (exponentiation), // (floor division).
    - Comparison operators: == (equality), != (inequality), > (greater than), < (less than), >= (greater than or equal to), <= (less than or equal to).
    - Assignment operators: = (assignment), += (addition assignment), -= (subtraction assignment), *= (multiplication assignment), /= (division assignment), %= (modulus assignment), **= (exponentiation assignment), //= (floor division assignment).
2. sum_result = x + y
3. result = a * b – c
4. is_adult = age >= 18
5. is_long_name = len(name) > 5
6. remainder = numerator % denominator
7. is_valid = (x == 10) or (y == 20)
8. num = int(num_str)
9. pi_str = str(pi)
10. is_valid_bool = bool(is_valid)
11. numbers_tuple = tuple(numbers)
12. Type casting, also known as type conversion, is the process of changing the data type of a value or variable from one type to another. It allows you to perform operations or assign values between different data types. Examples of type casting include converting an integer to a float, a string to an integer, or a list to a tuple.

## Lab Exercises 2.5

1. Program to determine if the user is an adult or a minor based on their age:

```
age = int(input("Enter your age: "))
if age >= 18:
    print("You are an adult")
else:
    print("You are a minor")
```

2. Program to determine if a number is positive, negative, or zero:

```
number = float(input("Enter a number: "))
if number > 0:
    print("Positive")
elif number < 0:
    print("Negative")
else:
    print("Zero")
```

3. Program to check if a given year is a leap year:

```
year = int(input("Enter a year: "))
if (year % 4 == 0 and year % 100 != 0) or year % 400 == 0:
    print("Leap year")
else:
    print("Not a leap year")
```

4. Program to determine the letter grade based on a numeric grade:

```
grade = float(input("Enter the grade: "))
if grade >= 90:
    print("A")
elif grade >= 80:
    print("B")
elif grade >= 70:
    print("C")
elif grade >= 60:
```

```
    print("D")
else:
    print("F")
```

5. Program to find the maximum among three numbers:

```
num1 = float(input("Enter the first number: "))
num2 = float(input("Enter the second number: "))
num3 = float(input("Enter the third number: "))
maximum = max(num1, num2, num3)
print("The maximum number is:", maximum)
```

6. Program to print the corresponding month name based on the input month number:

```
month = int(input("Enter a month number: "))
if month == 1:
    print("January")
elif month == 2:
    print("February")
elif month == 3:
    print("March")
elif month == 4:
    print("April")
elif month == 5:
    print("May")
elif month == 6:
    print("June")
elif month == 7:
    print("July")
elif month == 8:
    print("August")
elif month == 9:
    print("September")
elif month == 10:
```

```
    print("October")
elif month == 11:
    print("November")
elif month == 12:
    print("December")
else:
    print("Invalid month number")
```

7. Program to check if a given string is a palindrome:

```
word = input("Enter a word: ")
if word == word[::-1]:
    print("Palindrome")
else:
    print("Not a palindrome")
```

8. Program to check if a number is divisible by both 3 and 5:

```
number = int(input("Enter a number: "))
if number % 3 == 0 and number % 5 == 0:
    print("Divisible by 3 and 5")
else:
    print("Not divisible by 3 and 5")
```

9. Program to calculate the discount based on the total amount of a shopping cart:

```
total_amount = float(input("Enter the total amount: "))
if total_amount >= 100:
    discount = total_amount * 0.1
elif total_amount >= 50:
    discount = total_amount * 0.05
else:
    discount = 0
discounted_amount = total_amount - discount
print("Discounted amount:", discounted_amount)
```

10. Program to determine if a year is a leap year, century leap year, or neither:

```
year = int(input("Enter a year: "))


if (year % 4 == 0 and year % 100 != 0) or year % 400 == 0:
    if year % 100 == 0:
        print("Century leap year")
    else:
        print("Leap year")
else:
    print("Not a leap year or century leap year")
```

## Lab Exercises 2.6

1. Program to print each number in a list multiplied by 2 using a for loop:

```
numbers = [1, 2, 3, 4, 5]
for number in numbers:
    result = number * 2
    print(result)
```

2. Program to print each character of a string in reverse order using a while loop:

```
string = input("Enter a string: ")
length = len(string)
index = length - 1
while index >= 0:
    print(string[index])
    index -= 1
```

3. Program to count the number of vowels in a string using a for loop:

```
string = input("Enter a string: ")
vowels = 'aeiou'
count = 0
for char in string:
    if char.lower() in vowels:
        count += 1
print("Total count of vowels:", count)
```

4. Program to calculate the sum of positive numbers entered by the user using a while loop:

```
sum_positive = 0
while True:
    number = int(input("Enter a number (negative to quit): "))
    if number < 0:
        break
    sum_positive += number
```

```
print("Sum of positive numbers:", sum_positive)
```

5. Program to greet names based on the first letter using a for loop:

```
names = ['Alice', 'Bob', 'Anna', 'David']
for name in names:
    if name[0] == 'A':
        print("Hello,", name + "!")
    else:
        print("Greetings,", name + "!")
```

6. Program to guess a random number generated by the computer using a while loop:

```
import random
random_number = random.randint(1, 10)
while True:
    guess = int(input("Guess the number (1-10): "))
    if guess == random_number:
        print("Congratulations! You guessed correctly.")
        break
    elif guess > random_number:
        print("Too high")
    else:
        print("Too low")
```

7. Program to print words in lowercase or uppercase based on their length using a for loop:

```
words = ['apple', 'banana', 'cat', 'elephant']
for word in words:
    if len(word) > 5:
        print(word.upper())
    else:
        print(word.lower())
```

8. Program to print the square of each number in a range using a for loop:

```
for number in range(1, 11):
    square = number ** 2
    print(square)
```

9. Program to count the number of words with more than 3 characters in a sentence using a for loop:

```
sentence = input("Enter a sentence: ")
words = sentence.split()
count = 0
for word in words:
    if len(word) > 3:
        count += 1
print("Number of words with more than 3 characters:", count)
```