



**Module Title**

**Fundamentals of Data Science**

**Assessment Weightage & Type**

**Weekly Assignment 3 and 4 - Coursework & Regular**

**Year**

**2025**

**Student Name: NIRVIK K.C.**

**UWE ID: 25024649**

**Assignment Due Date: July 7, 2025**

**Assignment Submission Date: July 7, 2025**



## **Bi-weekly assignment**

### **Module Details**

<b>Module Code</b>	<b>UFCFK1-15-0</b>
<b>Module Title</b>	<b>Fundamentals of Data Science</b>
<b>Module Tutors</b>	<b>Saurav Gautam</b>
<b>Year</b>	<b>2024-2025</b>
<b>Component/Element Number</b>	<b>PSA/Bi-weekly assignment/Regular</b>
<b>Weighting</b>	<b>10%</b>

### **Dates**

<b>Submission Date</b>	<b>07-July-2025</b>
<b>Submission Place</b>	<b>Backboard</b>
<b>Submission Time</b>	<b>23:59</b>
<b>Submission Notes</b>	<b>Submit Gitlab URL</b>

## **Assignment 1**

This assignment consists of the programming questions related to the topics of week 3 and week 4. The main topics of questions are: Python Basics, Operators, and Conditional Statements.

All the students are required to follow the format of the program as specified in the guideline below.

1. All the programs should have initial **doc string** comment (‘’ description of program’’) mentioning what your program will do.
2. Try to maintain single/multi-line comments in the place where needed to make the program understandable.
3. Maintain proper indention and newline spaces to increase the readability of the program.
4. The deliverable are 2 type of files (a single word file and multiple python program files):
  - a) Separate python program files with **.py** extension (e.g. program\_name.py). Provide a relevant name to your program file on the basis of functionality of the program.
  - b) A word file describing the working of all the programs according to their number. The details required in this is the description of program, screenshot of the testing (input given and output obtained in the execution environment such as IDLE or Command prompt or terminal whichever you prefer.). It is preferred that you work with multiple inputs and outputs.

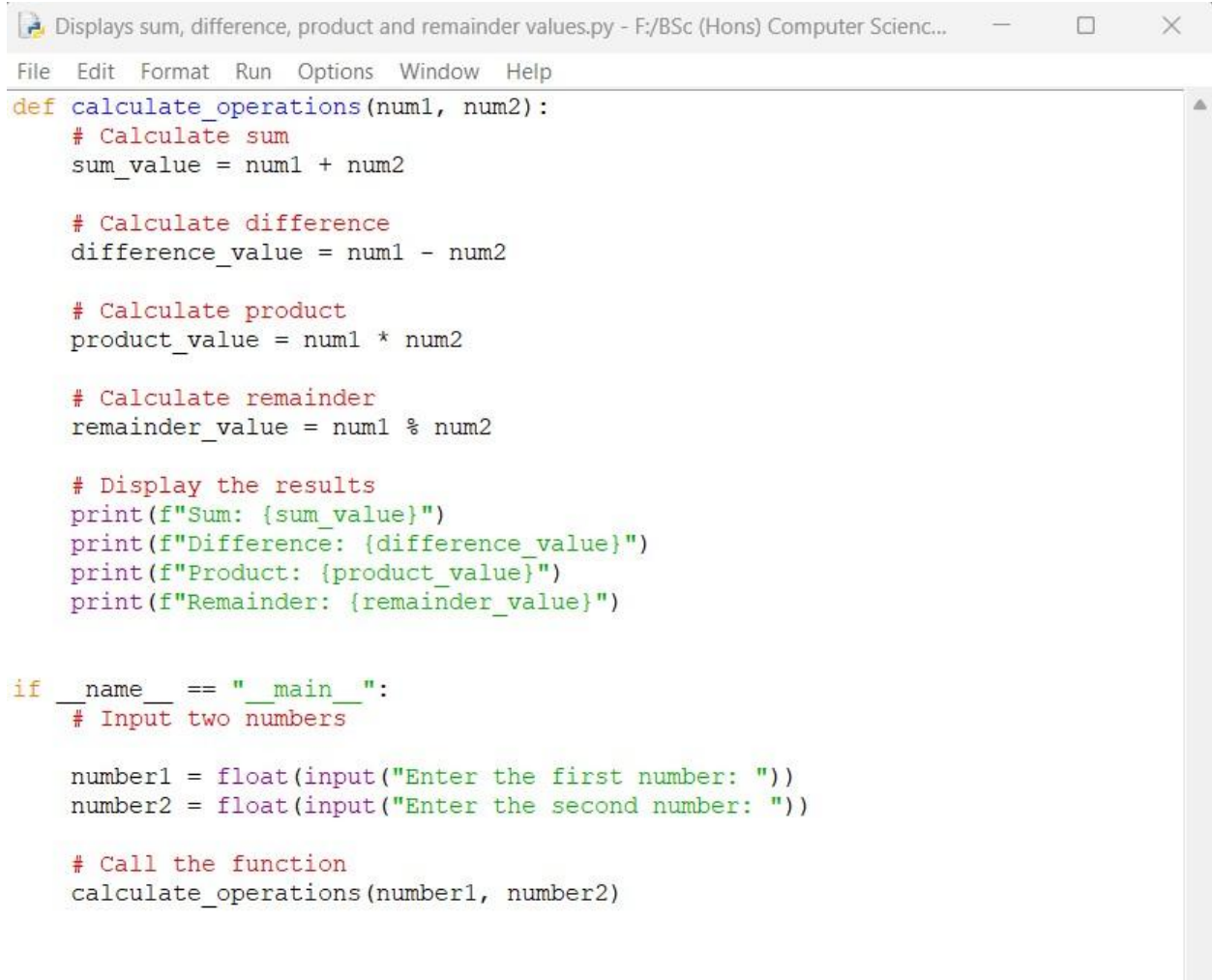
## Questions

1. Write a program to create a function which accepts 2 numbers and displays the sum, difference, product and the remainder values.

### Answer:

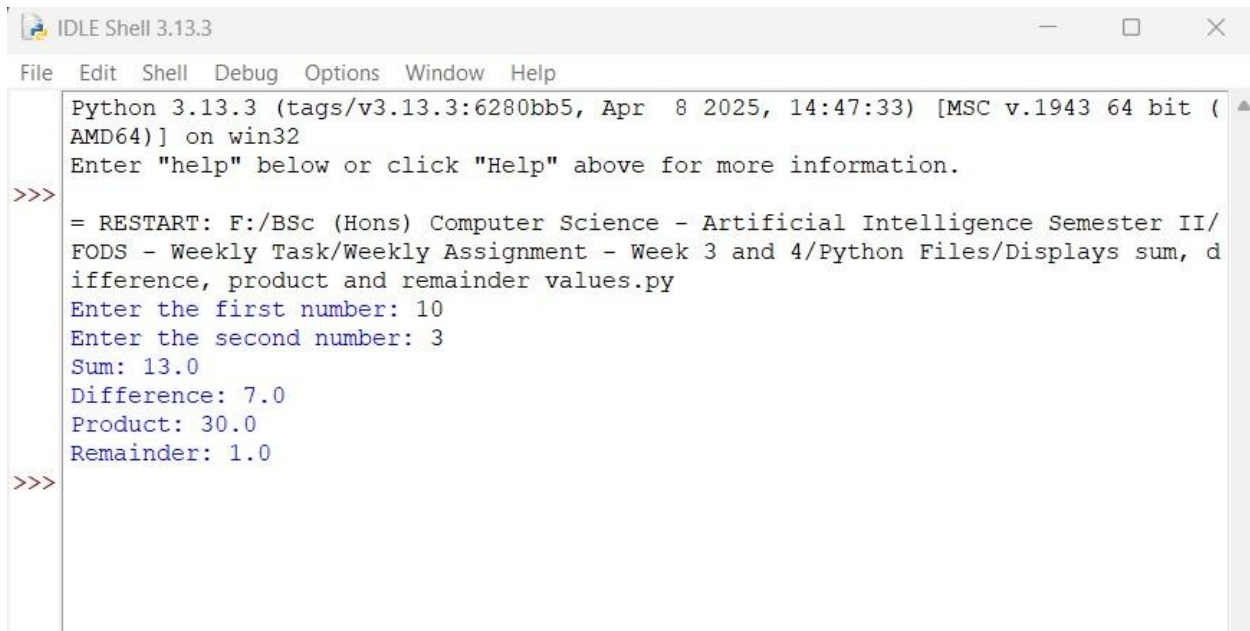
The given python program below defines a function to accept two numbers and displays their sum, difference, product, and remainder.

### Following code for input:

A screenshot of a Python IDE window. The title bar reads "Displays sum, difference, product and remainder values.py - F:/BSc (Hons) Computer Scienc...". The menu bar includes "File", "Edit", "Format", "Run", "Options", "Window", and "Help". The code editor contains the following Python code:

```
def calculate_operations(num1, num2):  
    # Calculate sum  
    sum_value = num1 + num2  
  
    # Calculate difference  
    difference_value = num1 - num2  
  
    # Calculate product  
    product_value = num1 * num2  
  
    # Calculate remainder  
    remainder_value = num1 % num2  
  
    # Display the results  
    print(f"Sum: {sum_value}")  
    print(f"Difference: {difference_value}")  
    print(f"Product: {product_value}")  
    print(f"Remainder: {remainder_value}")  
  
if __name__ == "__main__":  
    # Input two numbers  
  
    number1 = float(input("Enter the first number: "))  
    number2 = float(input("Enter the second number: "))  
  
    # Call the function  
    calculate_operations(number1, number2)
```

## Output obtained in execution:



```

Python 3.13.3 (tags/v3.13.3:6280bb5, Apr 8 2025, 14:47:33) [MSC v.1943 64 bit (AMD64)] on win32
Enter "help" below or click "Help" above for more information.
>>>
= RESTART: F:/BSc (Hons) Computer Science - Artificial Intelligence Semester II/
FODS - Weekly Task/Weekly Assignment - Week 3 and 4/Python Files/Displays sum, d
ifference, product and remainder values.py
Enter the first number: 10
Enter the second number: 3
Sum: 13.0
Difference: 7.0
Product: 30.0
Remainder: 1.0
>>>

```

Python Program File: “Saved the above code in a file named “Displays sum, difference, product, and remainder values.py.”

## Explanation of code:

Function Definition:

Def calculate\_operations(num1, num2):

This line defines a function named ‘calculate\_operations’ that takes two parameters, ‘num1’ and ‘num2’. The parameters will hold the two numbers for we want to perform the required operations.

Calculating the Sum:

The sum of two parameters ‘num1’ and ‘num2’ is calculated and the result is stored in the variable named ‘sum\_value’.

### Calculating the Difference:

The difference between 'num1' and 'num2' is calculated and the result is stored in the variable named 'difference\_value'.

### Calculating the Product:

The product of 'num1' and 'num2' is calculated and the result is stored in the variable in the variable named 'product\_value'.

### Calculating the Remainder:

The remainder when 'num1' is divided by 'num2' is calculated and the result is stored in 'remainder\_value'.

### Display the Result:

The results of the calculations in a formatted manner. The 'f' before the string allows for formatted string literals, making it easy to include variable values directly in the output.

### Main Program Execution:

If `__name__ == "__main__"`:

This line checks if the script is being run directly ( as opposite to being imported as a module in another script). If it is, the code block under it will execute.

### User Input:

The program prompts the user to enter two numbers. The 'input()' function reads the input as a string, and 'float()' converts it to a floating-point number.

### Function Call:

The function 'calculation\_operations' call the two numbers provided by the user triggering the calculation and displaying the results

The program displays the results of the calculations. The sum, difference, product, and remainder of two numbers are shown in the output.

#### Conclusion of the Program:

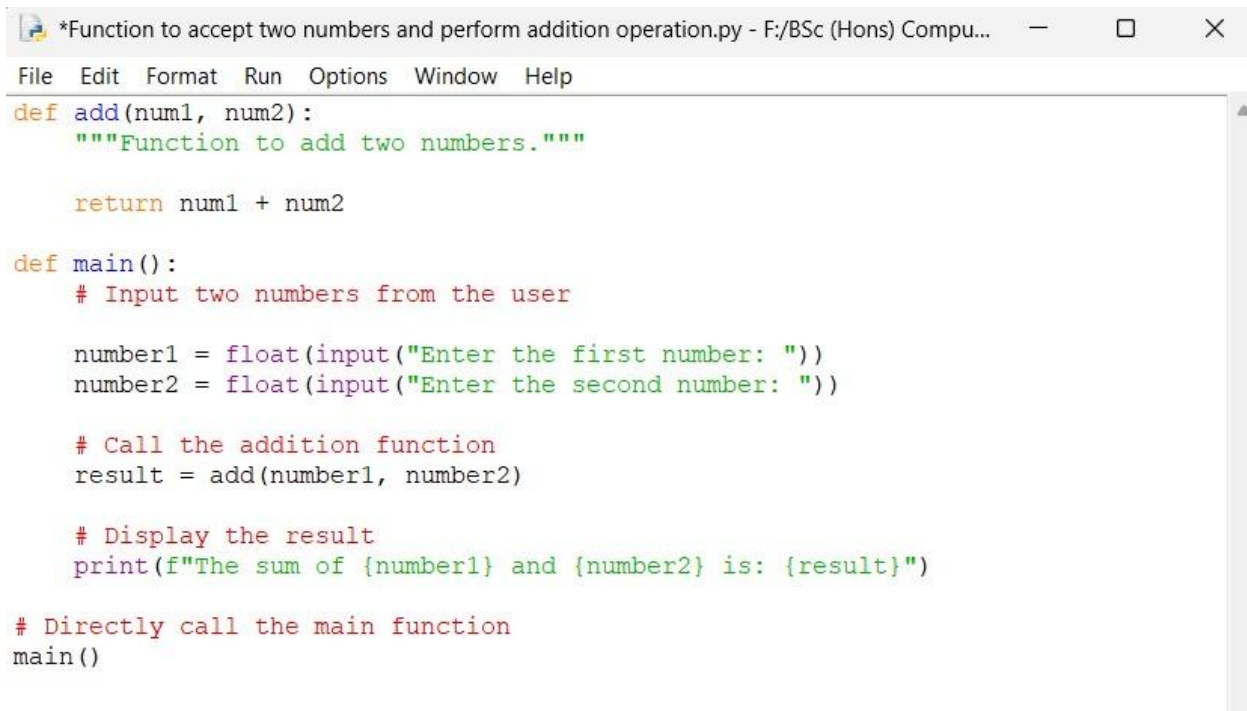
The given program is an example of how users can input two numbers and then calculates and displays their sum, difference, product, and remainder values. It demonstrates basic arithmetic operations in python.

2. Write a program to create separate functions for below mentioned mathematical calculations which would return the values back to the program. The functions should accept the 2 number which are inputs from the user and passed to them. The program should display the output in a proper format.

#### I. Addition

##### **Answer:**

The given python program below defines a separate function for operation like addition which accepts two numbers as input, performs the addition, and returns the result.

**Following code for input:**

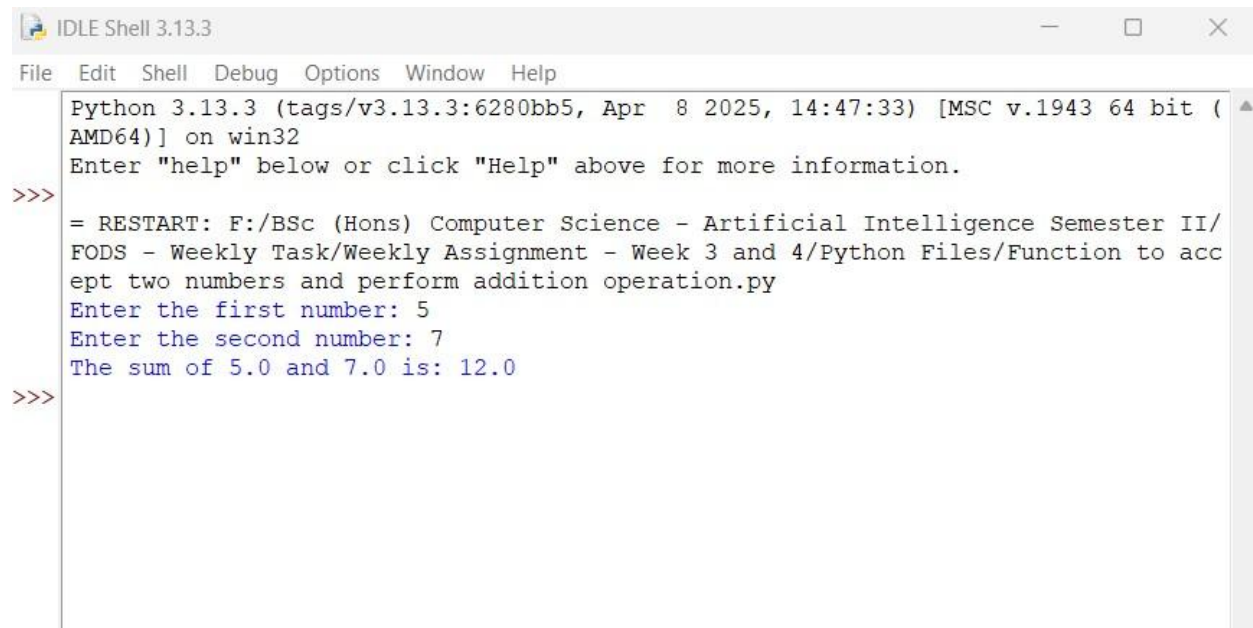
```
*Function to accept two numbers and perform addition operation.py - F:/BSc (Hons) Compu...
File Edit Format Run Options Window Help
def add(num1, num2):
    """Function to add two numbers."""
    return num1 + num2

def main():
    # Input two numbers from the user
    number1 = float(input("Enter the first number: "))
    number2 = float(input("Enter the second number: "))

    # Call the addition function
    result = add(number1, number2)

    # Display the result
    print(f"The sum of {number1} and {number2} is: {result}")

# Directly call the main function
main()
```

**Output obtained in execution:**

```
IDLE Shell 3.13.3
File Edit Shell Debug Options Window Help
Python 3.13.3 (tags/v3.13.3:6280bb5, Apr 8 2025, 14:47:33) [MSC v.1943 64 bit (AMD64)] on win32
Enter "help" below or click "Help" above for more information.
>>> = RESTART: F:/BSc (Hons) Computer Science - Artificial Intelligence Semester II/
FODS - Weekly Task/Weekly Assignment - Week 3 and 4/Python Files/Function to acc
ept two numbers and perform addition operation.py
Enter the first number: 5
Enter the second number: 7
The sum of 5.0 and 7.0 is: 12.0
>>>
```



Python Program File: “Function to accept two numbers and perform addition operation.py.”

### **Explanation of code:**

Function Definition:

```
def add(num1, num2):
```

The function ‘add’ takes two parameters, ‘num1’ and ‘num2’, and returns their sum.

Main Function:

The function ‘def main():’ acts as the main entry point for the program.

User Input:

The program prompts the user to enter two numbers, which are converted to floating-point numbers.

Function Call:

The ‘add’ function is called with the numbers provided by the user, and the result is stored in the variable ‘result’.

Main Program Execution:

The ‘main()’ function is called directly at the end of the script, which executes the program without the need for the ‘if name == ‘\_\_name\_\_’ == “\_\_main\_\_”:' construct.

Displaying the Result:

The program prints the result in a formatted string, showing the two input numbers and their sum.

## Conclusion of the Program:

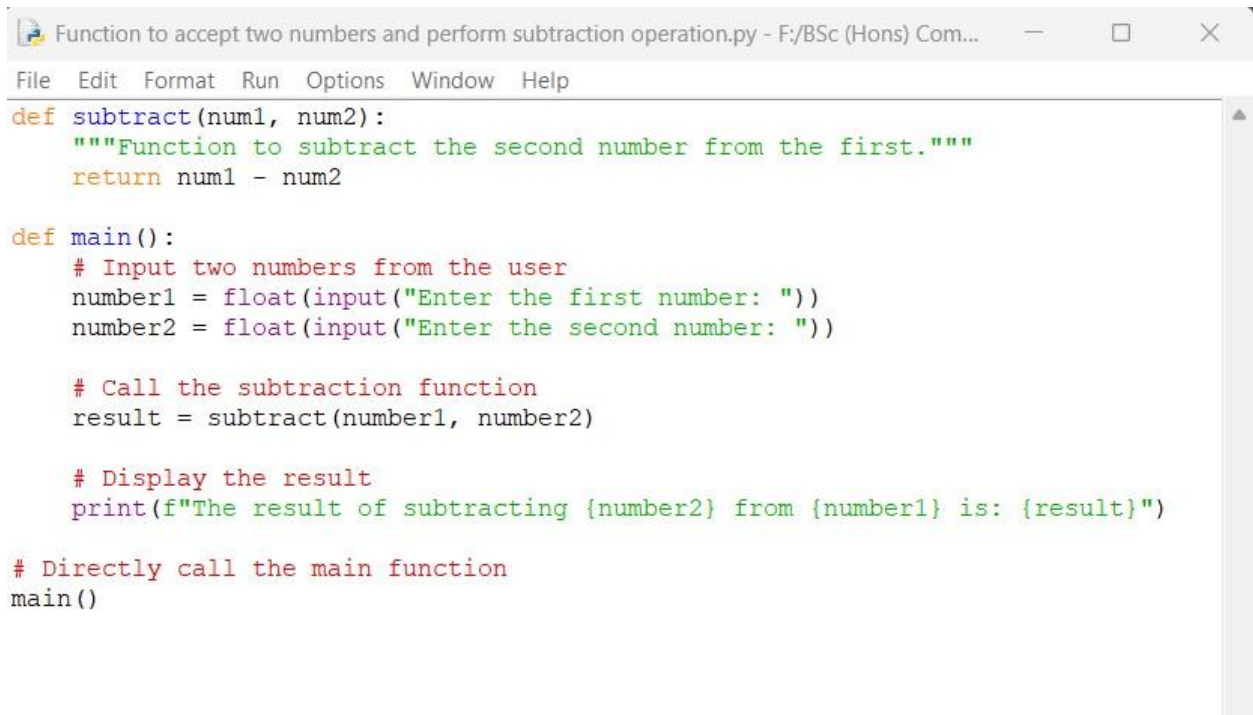
The given program demonstrates how to create a function for addition operation, accept user input, and display the result.

## II. Subtraction

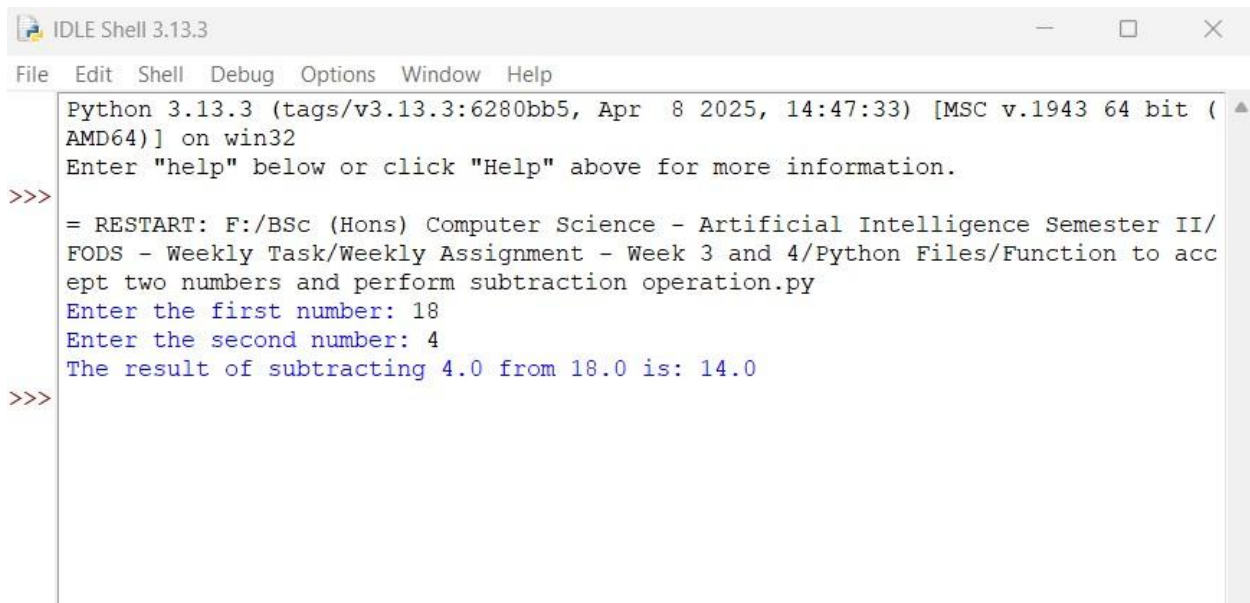
### Answer:

The given python below defines a separate function for subtraction. The program accepts two numbers as input, performs the subtraction, and returns the result. The program displays the output in a proper format.

### Following code for input:

A screenshot of a Python IDE window titled "Function to accept two numbers and perform subtraction operation.py - F:/BSc (Hons) Com...". The window has a menu bar with "File", "Edit", "Format", "Run", "Options", "Window", and "Help". The code is as follows:

```
def subtract(num1, num2):  
    """Function to subtract the second number from the first."""  
    return num1 - num2  
  
def main():  
    # Input two numbers from the user  
    number1 = float(input("Enter the first number: "))  
    number2 = float(input("Enter the second number: "))  
  
    # Call the subtraction function  
    result = subtract(number1, number2)  
  
    # Display the result  
    print(f"The result of subtracting {number2} from {number1} is: {result}")  
  
# Directly call the main function  
main()
```

**Output obtained in execution:**

```
Python 3.13.3 (tags/v3.13.3:6280bb5, Apr 8 2025, 14:47:33) [MSC v.1943 64 bit (AMD64)] on win32
Enter "help" below or click "Help" above for more information.
>>> = RESTART: F:/BSc (Hons) Computer Science - Artificial Intelligence Semester II/
FODS - Weekly Task/Weekly Assignment - Week 3 and 4/Python Files/Function to accept two numbers and perform subtraction operation.py
Enter the first number: 18
Enter the second number: 4
The result of subtracting 4.0 from 18.0 is: 14.0
>>>
```

Python Program File: “Function to accept two numbers and perform subtraction operation.py.”

**Explanation of code:**

Function Definition:

The function ‘subtract’, takes two parameters as ‘num1’ and ‘num2’, and returns the result of subtracting ‘num2’ from ‘num1’.

Main Function:

The function def main(): acts as the main entry point for the program.

User Input:

The program prompts the user to enter two numbers, which are converted to floating-point numbers.

Function Call:

```
def subtract(num1, num2):
```

This function takes two parameters, 'num1' and 'num2' and returns the result of subtracting 'num2' from 'num1'.

Main Function:

```
def main():
```

This function serves as the main entry point for the program.

User Input:

The program prompts the user to enter two numbers, which are converted to floating-point numbers.

Function Call:

```
result = subtract(number1, number2)
```

The 'subtract' function accepts number from the user, and the result is stored in the variable 'result'.

Displaying the Result:

The program outputs a formatted string displaying the two numbers and the subtraction result.

Direct Function Call:

The 'main()' function is called directly at the end of the script, which executes the program.

Conclusion of the Program:

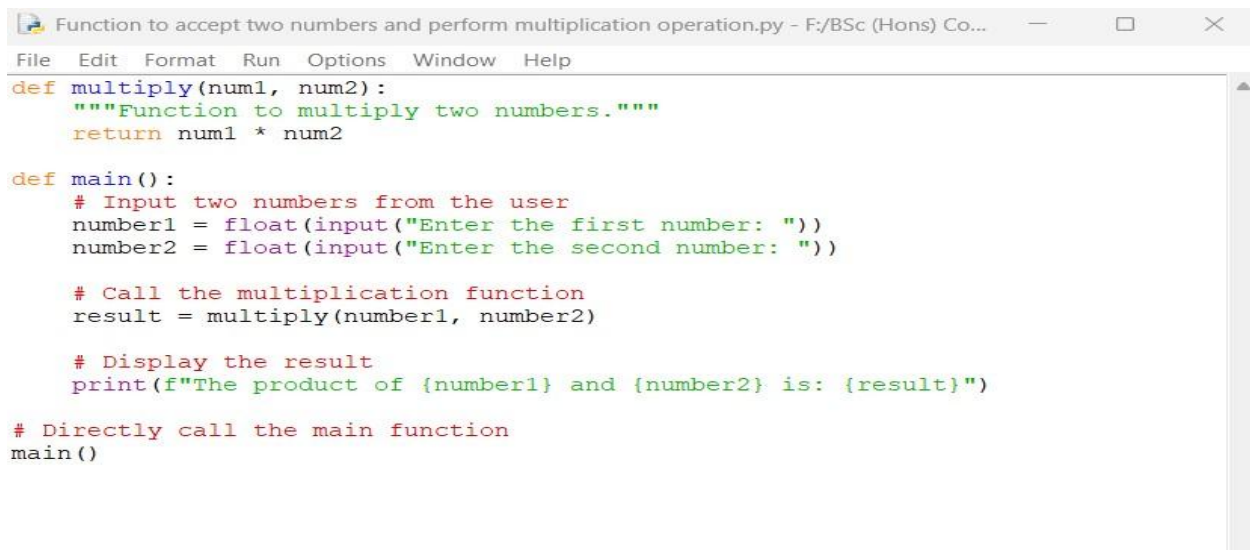
The program demonstrates how to create a separate function for subtraction, accept user input, and display the result in a clear format.

### III. Multiplication

Answer:

The given python program below creates a function for multiplication, which accepts two input numbers, performs the multiplication, and returns the result.

**Following code for input:**



```
Function to accept two numbers and perform multiplication operation.py - F:/BSc (Hons) Co...
File Edit Format Run Options Window Help
def multiply(num1, num2):
    """Function to multiply two numbers."""
    return num1 * num2

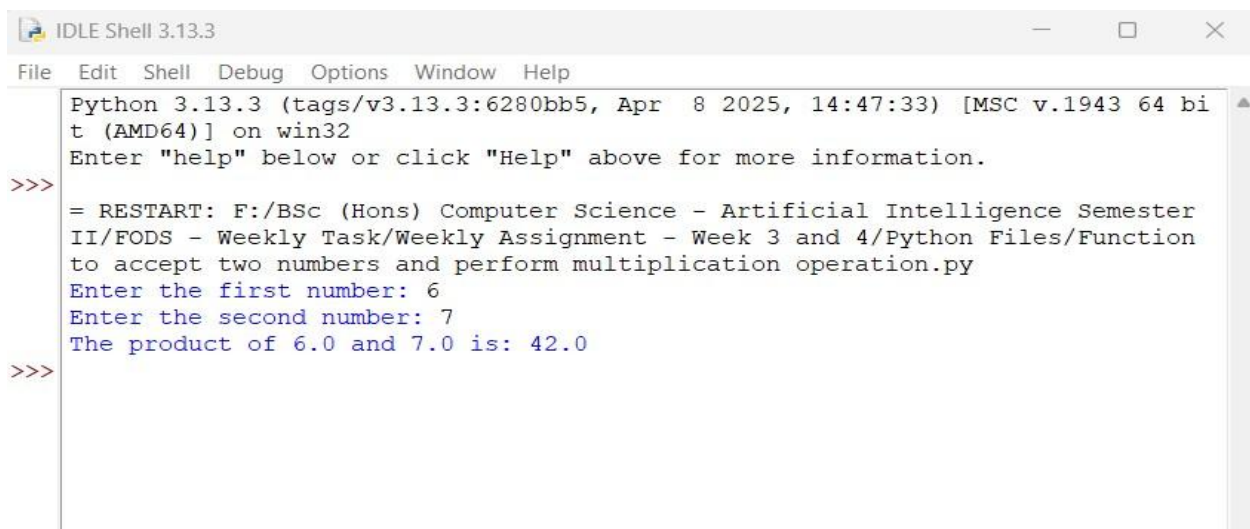
def main():
    # Input two numbers from the user
    number1 = float(input("Enter the first number: "))
    number2 = float(input("Enter the second number: "))

    # Call the multiplication function
    result = multiply(number1, number2)

    # Display the result
    print(f"The product of {number1} and {number2} is: {result}")

# Directly call the main function
main()
```

**Output obtained in execution:**



```
IDLE Shell 3.13.3
File Edit Shell Debug Options Window Help
Python 3.13.3 (tags/v3.13.3:6280bb5, Apr 8 2025, 14:47:33) [MSC v.1943 64 bit (AMD64)] on win32
Enter "help" below or click "Help" above for more information.

>>> = RESTART: F:/BSc (Hons) Computer Science - Artificial Intelligence Semester
II/FODS - Weekly Task/Weekly Assignment - Week 3 and 4/Python Files/Function
to accept two numbers and perform multiplication operation.py
Enter the first number: 6
Enter the second number: 7
The product of 6.0 and 7.0 is: 42.0
>>>
```

Python Program File: “Function to accept two numbers and perform multiplication operation.py.”

### **Explanation of code:**

Function Definition:

```
def multiply(num1, num2):
```

This line defines a function named ‘multiply’ that two parameters, ‘num1’ and ‘num2’. The function calculates the product of the two numbers using the multiplication operator ‘\*’ and returns the result.

Main Function:

The main function acts the main entry point for the program.

User Input:

The program prompts the user to enter two number. The ‘input()’ function reads the input as a string, and ‘float()’ converts it to a floating-point number. This allows for user to input decimal values.

Function Call:

```
result = multiply(number1, number2)
```

The ‘multiply’ function is called with the numbers provided by the user, and the result of the multiplication is stored in the variable ‘result’.

Displaying the Result:

The program prints the result in a formatted string, which makes it easy to include variable values directly in the output.

### Direct Function Call:

The 'main' function is called directly at the end of the program, which executes the program.

### Conclusion of the Program:

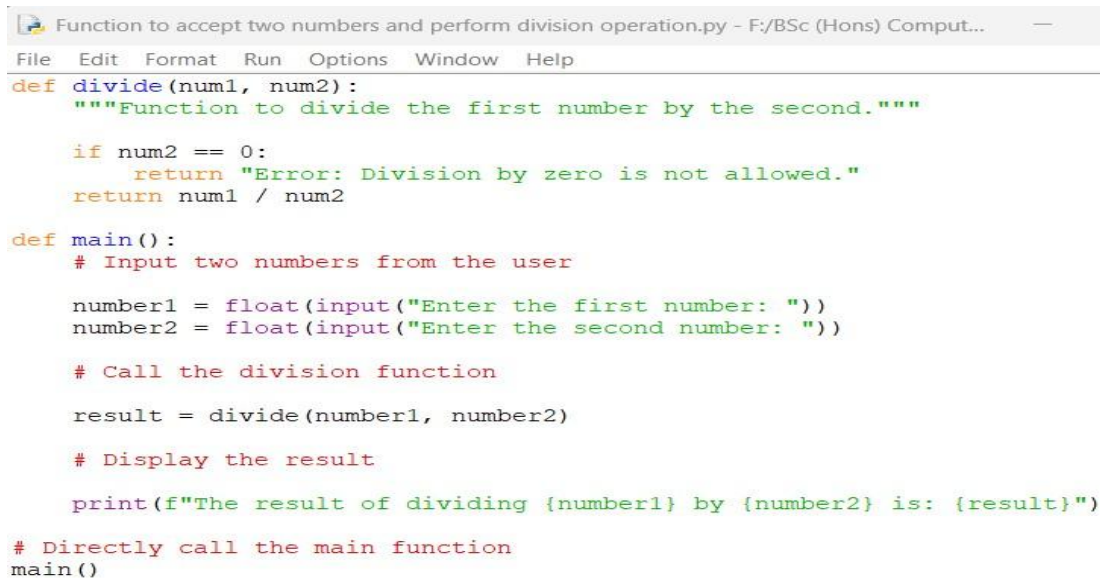
The program demonstrates how to create function for multiplication, accept the user input, and display the result in a clear format.

## IV. Division

### Answer:

The given python program below defines a separate function for division, which accepts two numbers as input from the user, performs the division calculation, and returns the result.

### Following code for input:



```
Function to accept two numbers and perform division operation.py - F:/BSc (Hons) Comput...
File Edit Format Run Options Window Help
def divide(num1, num2):
    """Function to divide the first number by the second."""
    if num2 == 0:
        return "Error: Division by zero is not allowed."
    return num1 / num2

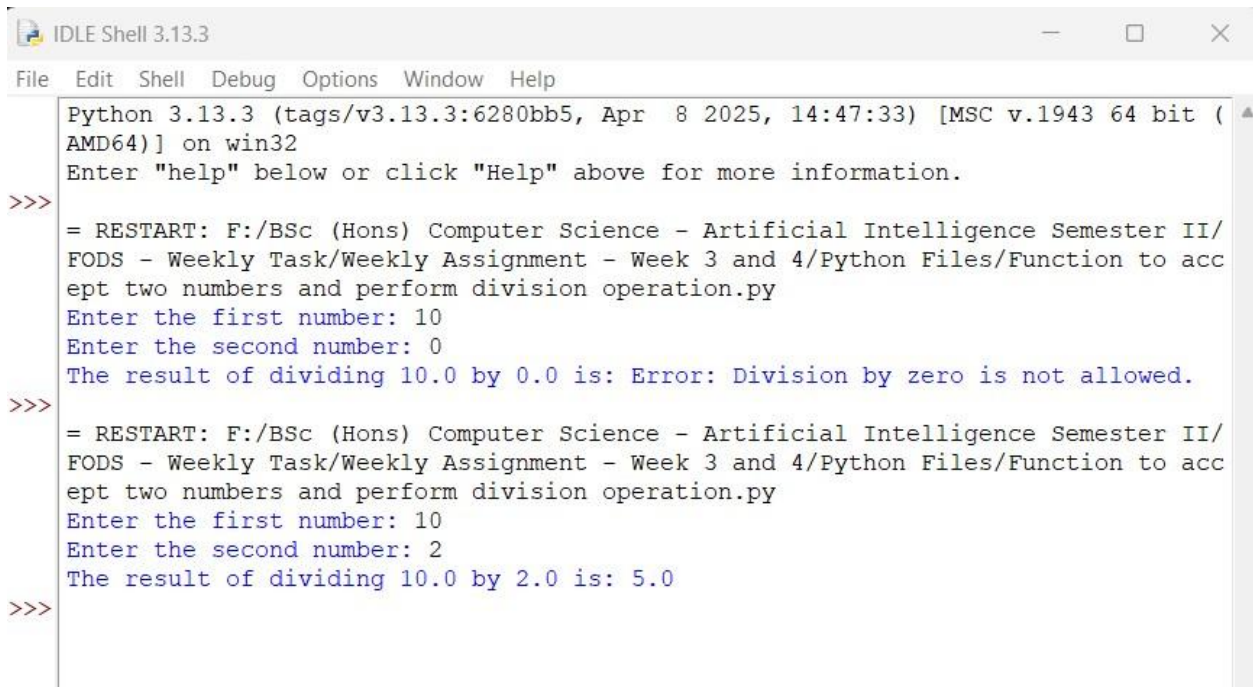
def main():
    # Input two numbers from the user
    number1 = float(input("Enter the first number: "))
    number2 = float(input("Enter the second number: "))

    # Call the division function
    result = divide(number1, number2)

    # Display the result
    print(f"The result of dividing {number1} by {number2} is: {result}")

# Directly call the main function
main()
```

## Output obtained in execution:



```

Python 3.13.3 (tags/v3.13.3:6280bb5, Apr 8 2025, 14:47:33) [MSC v.1943 64 bit (AMD64)] on win32
Enter "help" below or click "Help" above for more information.
>>>
= RESTART: F:/BSc (Hons) Computer Science - Artificial Intelligence Semester II/
FODS - Weekly Task/Weekly Assignment - Week 3 and 4/Python Files/Function to acc
ept two numbers and perform division operation.py
Enter the first number: 10
Enter the second number: 0
The result of dividing 10.0 by 0.0 is: Error: Division by zero is not allowed.
>>>
= RESTART: F:/BSc (Hons) Computer Science - Artificial Intelligence Semester II/
FODS - Weekly Task/Weekly Assignment - Week 3 and 4/Python Files/Function to acc
ept two numbers and perform division operation.py
Enter the first number: 10
Enter the second number: 2
The result of dividing 10.0 by 2.0 is: 5.0
>>>

```

Python Program File: “Function to accept two numbers and perform division operation.py.”

## Explanation of code:

Function Definition:

```
def divide(num1, num2):
```

The ‘divide’ function takes two parameters, ‘num1’ and ‘num2’. Then, the function first checks if ‘num2’ is zero to prevent undefined division by zero. If the ‘num2’ is zero, it returns an error message. If the ‘num2’ is not zero, it calculates the result of dividing ‘num1’ by ‘num2’ using the division operator ‘/’ and returns the result.



### Main Function:

The 'main' function acts as the main entry point for the program.

### User Input:

The program prompts the user to enter two numbers as input. The 'input()' function reads the input as a string, and 'float()' converts it to a floating-point number.

### Function Call:

```
result = divide(num1, num2)
```

The 'divide' function is called when the numbers are provided by the user, and the result of the division ( or error message) is stored in the variable 'result'.

### Displaying the Result:

The program prints the result of calculation in a formatted string. The 'f' before the string allows for formatted string literals, which makes it easy to include variable values directly in the output.

### Direct Function Call:

The 'main()' function is directly called at the end, which executes the program.

### Conclusion of the Program:

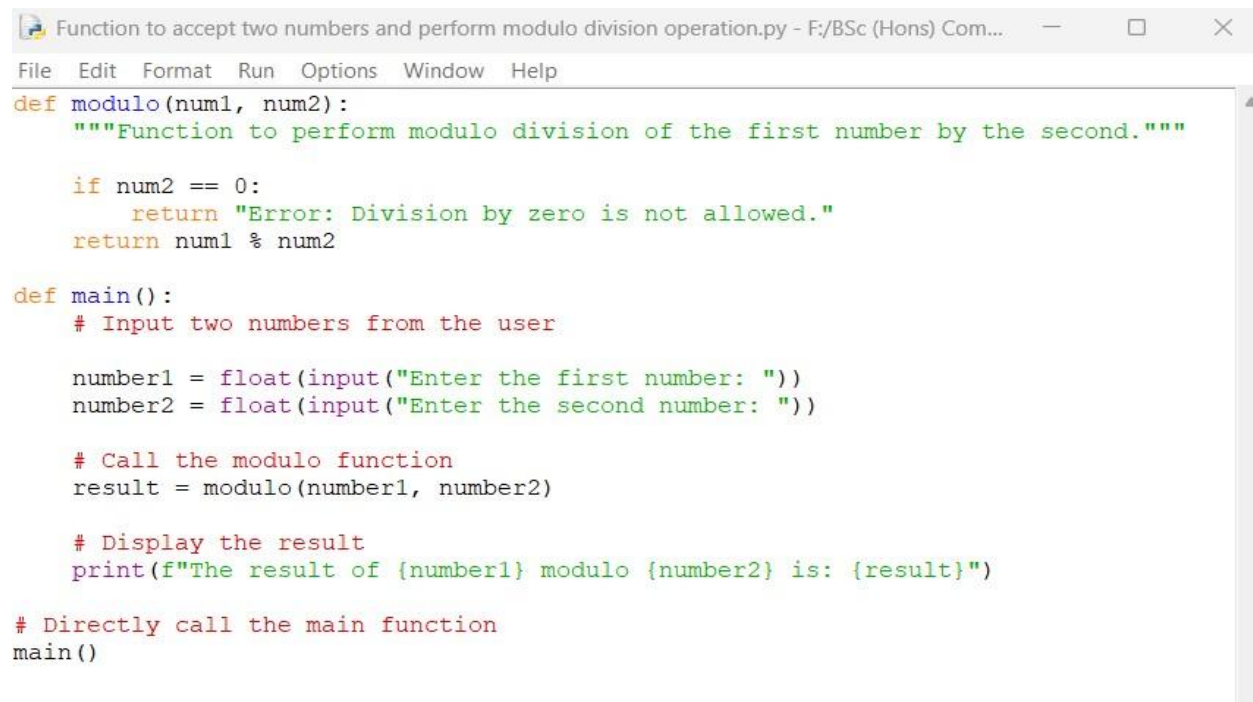
The given python program demonstrates how to create a function for division calculation, handle division by zero, accept user input, and display the result in a formatted way.

## V. Modulo division

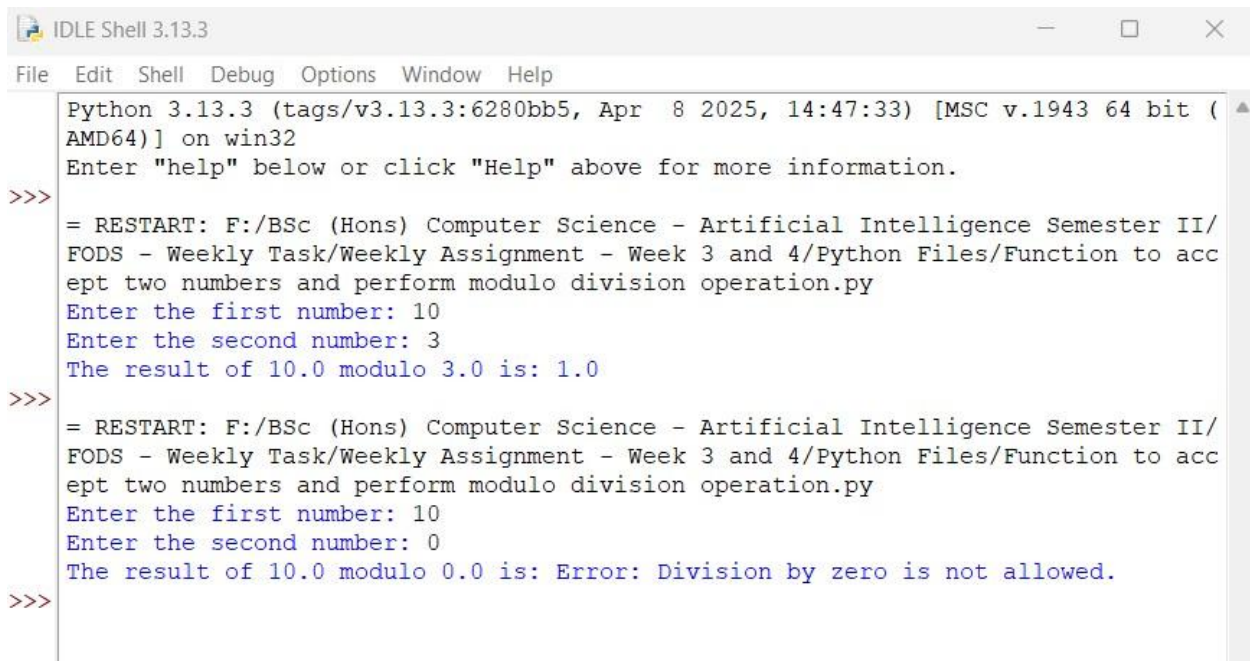
### Answer:

The given python program below defines a separate function for modulo operation, which accepts numbers as input, performs the modulo operation, and returns the result.

### Following code for input:

A screenshot of a Python IDE window titled "Function to accept two numbers and perform modulo division operation.py - F:/BSc (Hons) Com...". The window contains a Python script that defines a 'modulo' function and a 'main' function. The 'modulo' function takes two arguments, 'num1' and 'num2', and returns 'num1 % num2' after checking for a zero divisor. The 'main' function prompts the user for two numbers, calls the 'modulo' function, and prints the result. The code is as follows:

```
def modulo(num1, num2):  
    """Function to perform modulo division of the first number by the second."""  
  
    if num2 == 0:  
        return "Error: Division by zero is not allowed."  
    return num1 % num2  
  
def main():  
    # Input two numbers from the user  
  
    number1 = float(input("Enter the first number: "))  
    number2 = float(input("Enter the second number: "))  
  
    # Call the modulo function  
    result = modulo(number1, number2)  
  
    # Display the result  
    print(f"The result of {number1} modulo {number2} is: {result}")  
  
# Directly call the main function  
main()
```

**Output obtained in execution:**


```

IDLE Shell 3.13.3
File Edit Shell Debug Options Window Help
Python 3.13.3 (tags/v3.13.3:6280bb5, Apr 8 2025, 14:47:33) [MSC v.1943 64 bit (AMD64)] on win32
Enter "help" below or click "Help" above for more information.
>>>
= RESTART: F:/BSc (Hons) Computer Science - Artificial Intelligence Semester II/
FODS - Weekly Task/Weekly Assignment - Week 3 and 4/Python Files/Function to acc
ept two numbers and perform modulo division operation.py
Enter the first number: 10
Enter the second number: 3
The result of 10.0 modulo 3.0 is: 1.0
>>>
= RESTART: F:/BSc (Hons) Computer Science - Artificial Intelligence Semester II/
FODS - Weekly Task/Weekly Assignment - Week 3 and 4/Python Files/Function to acc
ept two numbers and perform modulo division operation.py
Enter the first number: 10
Enter the second number: 0
The result of 10.0 modulo 0.0 is: Error: Division by zero is not allowed.
>>>

```

Python Program File: “Function to accept two numbers and perform modulo division operation.py.”

**Explanation of code:**

Function Definition:

```
def modulo(num1, num2):
```

The ‘modulo’ function takes two parameters, ‘num1’ and ‘num2’. The function checks if the ‘num2’ is zero to prevent division by zero, which is undefined. If ‘num2’ is zero, then it returns an error message. If ‘num2’ is not zero, it calculates the result of the modulo operation using the modulo operator ‘%’ and returns the result.

Main Function:

The ‘main’ function acts as the main entry point for the program.

### User Input:

The program prompts the user to enter two numbers. The 'input()' function reads the input from the user as a string, and 'float()' converts it to a floating-point number.

### Function Call:

The 'module' function is called with the number provided by the user, and the result of the module operation ( or the error message) is stored in the variable 'result'.

### Displaying the Result:

The line prints the result in a formatted string. The 'f' before the string allows for formatted string literals, which makes it easy to include variable values directly in the output.

### Direct Function Call:

The 'main' function is called directly at the end of the program, which executes it.

### Conclusion of the Program:

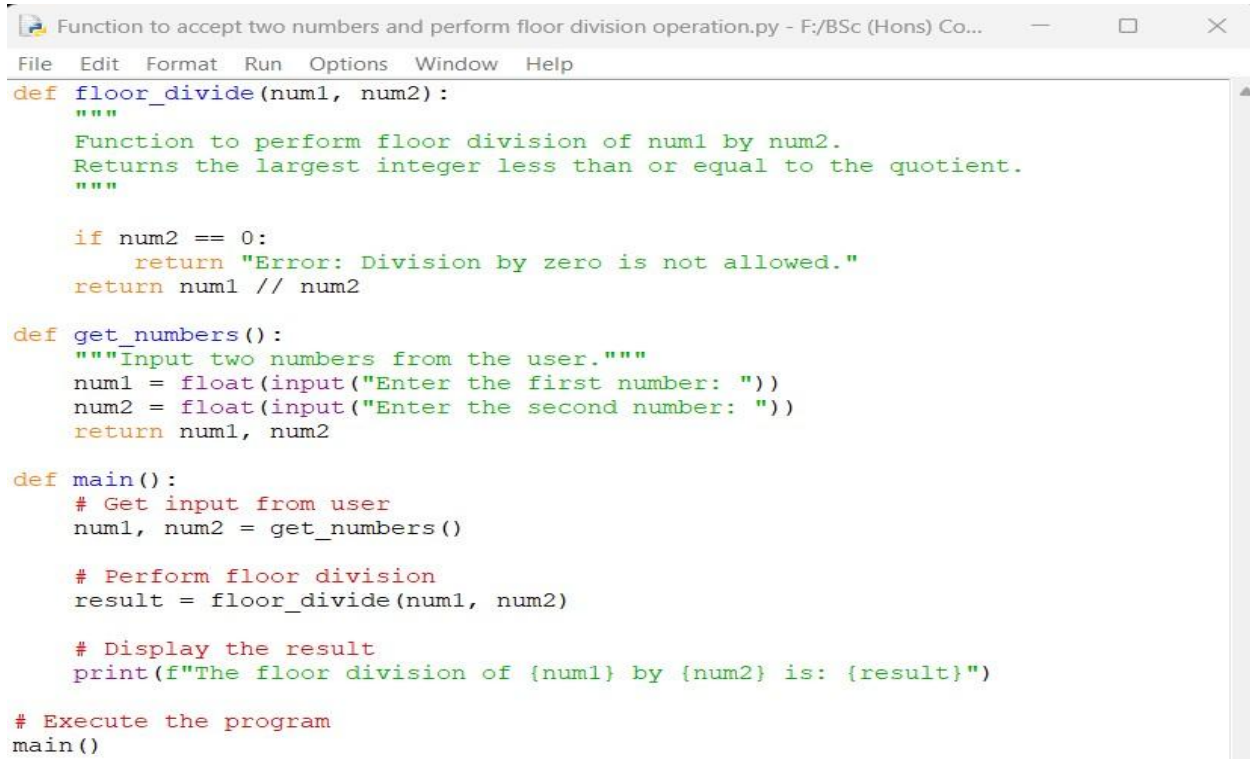
The given python program demonstrates how to create a function for modulo division, handle the division operation by zero, accept user input numbers, and display the result in a clear format.

## VI. Floor division

### **Answer:**

The given python program below performs floor division which accepts numbers as input, performs the modulo operation, and prints the statement at the end of the program.

## Following code for input:



```

Function to accept two numbers and perform floor division operation.py - F:/BSc (Hons) Co...
File Edit Format Run Options Window Help
def floor_divide(num1, num2):
    """
    Function to perform floor division of num1 by num2.
    Returns the largest integer less than or equal to the quotient.
    """

    if num2 == 0:
        return "Error: Division by zero is not allowed."
    return num1 // num2

def get_numbers():
    """Input two numbers from the user."""
    num1 = float(input("Enter the first number: "))
    num2 = float(input("Enter the second number: "))
    return num1, num2

def main():
    # Get input from user
    num1, num2 = get_numbers()

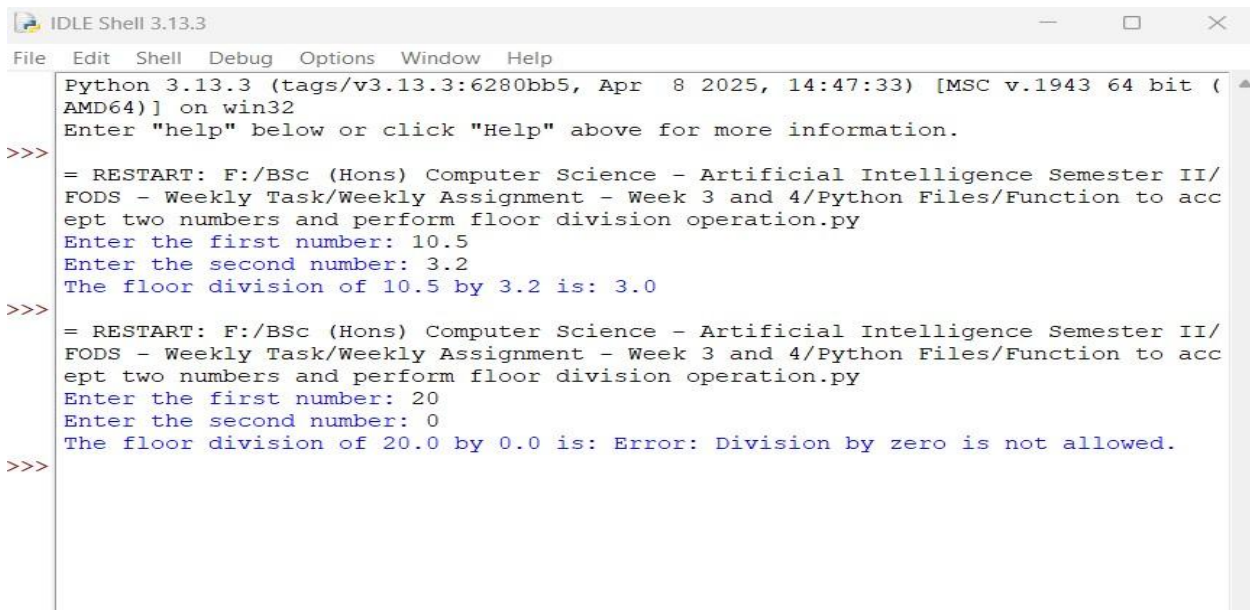
    # Perform floor division
    result = floor_divide(num1, num2)

    # Display the result
    print(f"The floor division of {num1} by {num2} is: {result}")

# Execute the program
main()

```

## Output obtained in execution:



```

IDLE Shell 3.13.3
File Edit Shell Debug Options Window Help
Python 3.13.3 (tags/v3.13.3:6280bb5, Apr 8 2025, 14:47:33) [MSC v.1943 64 bit (AMD64)] on win32
Enter "help" below or click "Help" above for more information.
>>>
= RESTART: F:/BSc (Hons) Computer Science - Artificial Intelligence Semester II/
FODS - Weekly Task/Weekly Assignment - Week 3 and 4/Python Files/Function to acc
ept two numbers and perform floor division operation.py
Enter the first number: 10.5
Enter the second number: 3.2
The floor division of 10.5 by 3.2 is: 3.0
>>>
= RESTART: F:/BSc (Hons) Computer Science - Artificial Intelligence Semester II/
FODS - Weekly Task/Weekly Assignment - Week 3 and 4/Python Files/Function to acc
ept two numbers and perform floor division operation.py
Enter the first number: 20
Enter the second number: 0
The floor division of 20.0 by 0.0 is: Error: Division by zero is not allowed.
>>>

```

Python Program File: “Function to accept two numbers and perform floor division operation.py.”

### **Explanation of code:**

Function Definition:

```
def floor_division(num1, num2):
```

This function takes two parameters. ‘num1’ and ‘num2’, and perform floor division using the ‘//’ operator. It checks if ‘num2’ is zero to prevent division by zero and returns an error message if it is.

### **Input Function:**

```
def get_numbers():
```

The function prompts the user for two input numbers and returns them as floating-point values using ‘float()’.

Main Function:

The ‘main’ function serves as the main entry point for the program.

User Input and Calculation:

```
num1, num2 = get_numbers()  
result = floor_divide(num1, num2)
```

The program retrieves user input and performs floor division operation.

Displaying the Result:

The result is printed directly in the ‘main()’ function, providing a clear output format.

Direct Function Call:

The 'main()' function is called to execute program.

Conclusion of the Program:

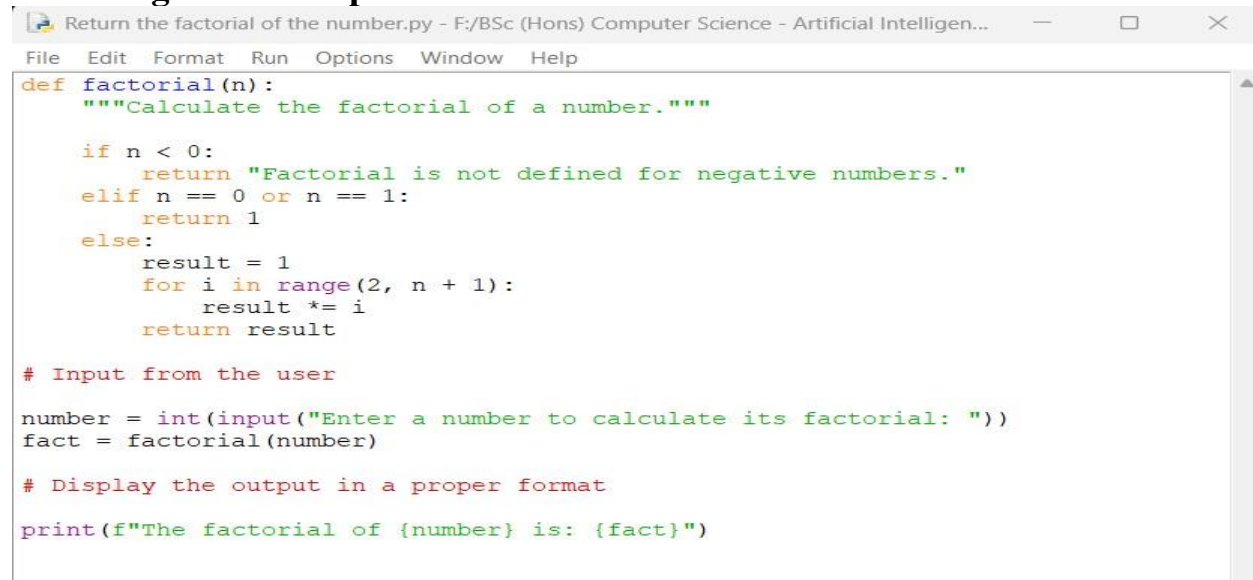
The given program is an example of how to gather user input and perform floor division in python. First, we must set up the variables, utilize the floor division operator (//), and display the results with the print() function.

3. Write a program to create a function which will accept a parameter and return the factorial of the number. The output should be displayed in a proper format.

**Answer:**

The given python program below that defines a function to calculate the factorial of a given number.

**Following code for input:**



```

Return the factorial of the number.py - F:/BSc (Hons) Computer Science - Artificial Intelligen...
File Edit Format Run Options Window Help
def factorial(n):
    """Calculate the factorial of a number."""
    if n < 0:
        return "Factorial is not defined for negative numbers."
    elif n == 0 or n == 1:
        return 1
    else:
        result = 1
        for i in range(2, n + 1):
            result *= i
        return result

# Input from the user
number = int(input("Enter a number to calculate its factorial: "))
fact = factorial(number)

# Display the output in a proper format
print(f"The factorial of {number} is: {fact}")

```

**Output obtained in execution:**

```

Python 3.13.3 (tags/v3.13.3:6280bb5, Apr 8 2025, 14:47:33) [MSC v.1943 64 bit (AMD64)] on win32
Enter "help" below or click "Help" above for more information.
>>>
= RESTART: F:/BSc (Hons) Computer Science - Artificial Intelligence Semester II/FODS - Weekly Task/Weekly Assignment - Week 3 and 4/Python Files/Return the factorial of the number.py
Enter a number to calculate its factorial: 5
The factorial of 5 is: 120
>>>
= RESTART: F:/BSc (Hons) Computer Science - Artificial Intelligence Semester II/FODS - Weekly Task/Weekly Assignment - Week 3 and 4/Python Files/Return the factorial of the number.py
Enter a number to calculate its factorial: 0
The factorial of 0 is: 1
>>>

```

Python Program File: “Return the factorial of the number.py.”

**Explanation of code:**

Function Definition:

```
def factorial(n):
```

This function named ‘factorial’ that takes a single parameter ‘n’, which is expected to be an integer.

Handling Negative input:

The conditional statement checks if the input number ‘n’ is less than zero. If ‘n’ is negative, the function returns a message indicating that the factorial is not defined for negative numbers. This prevents further calculations that would be invalid.



### Base Cases for Factorial:

This checks if 'n' is either '0' or '1'. The factorial of both '0' and '1' is defined to be '1' so the function returns '1' in these cases. This shows the factorial definition part.

### Calculating Factorial for Positive Integers:

In else block statement, if 'n' is greater than '1', the function enters this block to calculate the factorial. A variable 'result' is initialized to '1'. This variable will hold the cumulative product of the numbers from '2' to 'n'. In for loop, the loop iterates over from the range from '2' to 'n'. In each iteration, the current value of 'i' is multiplied with 'result', which effectively calculates the factorial by accumulating the product. After the loop complete, the function returns the factorial stored 'result' variable.

### User Input:

The program prompts user to enter a number for which they want to calculate the factorial. The input is read as string for user, then it is converted to an integer using 'int()'.

### Function Call and Result Storage:

```
fact = factorial(number)
```

The program calls the 'factorial' function with the number provided with the user and stores the result in the variable 'fact'. This allows the program to display the output of the factorial for further use.

### Output of the Program:

The program prints the result using f-string to embed values of 'number' and 'fact' directly into the output. The calculated factorial displayed clearly, making the program's output easy to understand.

**Conclusion of the Program:**

This given python program is an example of calculating the factorial of a given number while handling cases such negative numbers 0 and 1. The use of loop ensures that the program can handle any positive integer input.

4. Write a function to accept a list of numbers and print the occurrence of each number. The function should be tested well in the program by calling and sending various list of numbers.

**Answer:**

The given python program below defines a function to accept a list of numbers and print the occurrence of each. The program also includes test cases to demonstrate the function's functionality.

## Following code for input:

```

Print the occurrence of each number.py - F:/BSc (Hons) Computer Science - Artificial Intelligence Semester II/FODS - Weekly Tas...
File Edit Format Run Options Window Help
def count_occurrences(numbers):
    """Count and print the occurrences of each number in the list."""
    occurrences = {}

    for number in numbers:
        if number in occurrences:
            occurrences[number] += 1
        else:
            occurrences[number] = 1

    # Print the occurrences in a formatted way
    for number, count in occurrences.items():
        print(f"Number {number} occurs {count} time(s).")

# Test cases
if __name__ == "__main__":
    # Test case 1
    test_list_1 = [1, 2, 2, 3, 4, 4, 4, 5]
    print("Test Case 1:")
    count_occurrences(test_list_1)
    print()

    # Test case 2
    test_list_2 = [10, 20, 10, 30, 20, 10]
    print("Test Case 2:")
    count_occurrences(test_list_2)
    print()

    # Test case 3
    test_list_3 = [5, 5, 5, 5, 5]
    print("Test Case 3:")
    count_occurrences(test_list_3)
    print()

```

## Continue:

```

# Test case 4
test_list_4 = []
print("Test Case 4:")
count_occurrences(test_list_4)
print()

# Test case 5
test_list_5 = [1, 2, 3, 4, 5, 1, 2, 1]
print("Test Case 5:")
count_occurrences(test_list_5)

```

**Output obtained in execution:**

```

Python 3.13.3 (tags/v3.13.3:6280bb5, Apr 8 2025, 14:47:33) [MSC v.1943 64 bit (AMD64)] on win32
Enter "help" below or click "Help" above for more information.
>>>
= RESTART: F:/BSc (Hons) Computer Science - Artificial Intelligence Semester II/FODS - Weekly Task/Weekly Assignment - Week 3 and 4/Python Files/Print the occurrence of each number.py
Test Case 1:
Number 1 occurs 1 time(s).
Number 2 occurs 2 time(s).
Number 3 occurs 1 time(s).
Number 4 occurs 3 time(s).
Number 5 occurs 1 time(s).

Test Case 2:
Number 10 occurs 3 time(s).
Number 20 occurs 2 time(s).
Number 30 occurs 1 time(s).

Test Case 3:
Number 5 occurs 5 time(s).

Test Case 4:

Test Case 5:
Number 1 occurs 3 time(s).
Number 2 occurs 2 time(s).
Number 3 occurs 1 time(s).
Number 4 occurs 1 time(s).
Number 5 occurs 1 time(s).
>>>

```

Python Program File: “Print the occurrence of each number.py.”

**Explanation of code:****Function Definition:**

The function ‘count\_occurrences’ takes a list of numbers as input and is designed to count how many times each number appears in that list.

### Counting Occurrences:

An empty dictionary 'occurrences' is initialized to store the count of each number. The function iterates through each number in the input list, if the number is already a key in the dictionary, its count is incremented by 1. If the number is not in the dictionary, it is added with a count of 1.

### Output of the Program:

The function iterates through the dictionary and prints the number along with its occurrence count in a formatted manner.

### Testing the Function:

'if \_\_name\_\_ == "\_\_main\_\_":' block ensures that the test cases are executed only when the script is run directly.

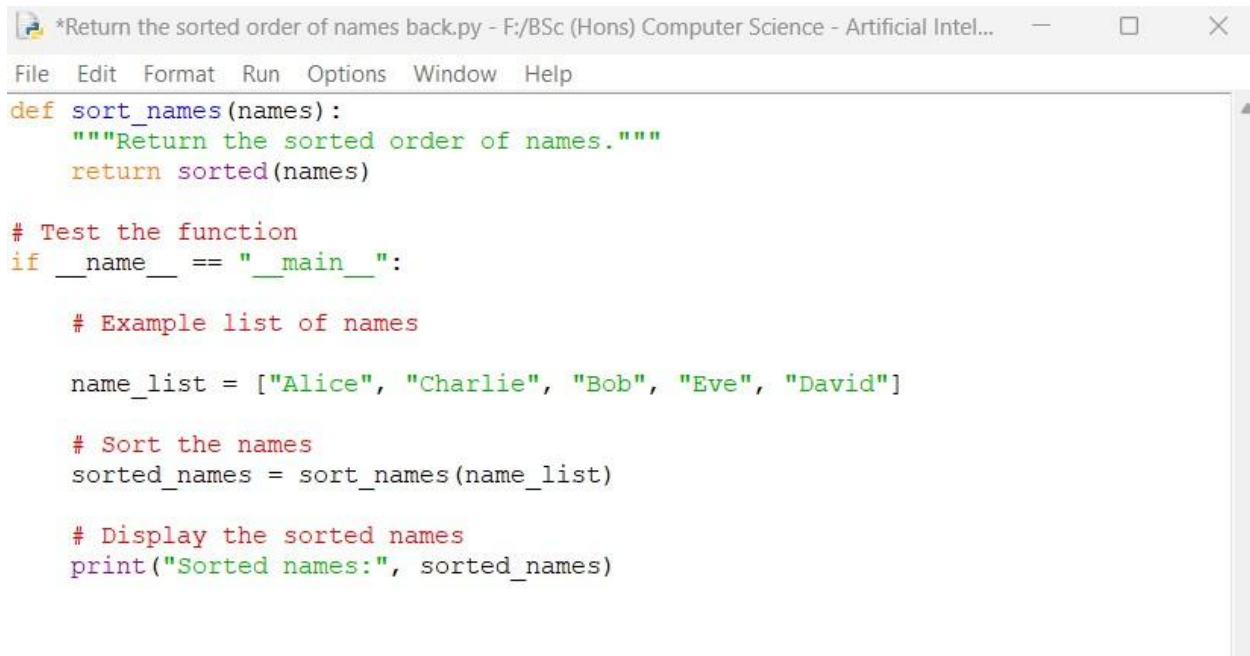
### Conclusion of the Program:

The given python program is an example of implementing a function to count and display the occurrences of each number in a given list.

5. Write a function to accept a list of names and return the sorted order of names back.

### **Answer:**

The given python program defines a function to accept a list of names and return the sorted order of those names.

**Following code for input:**


```

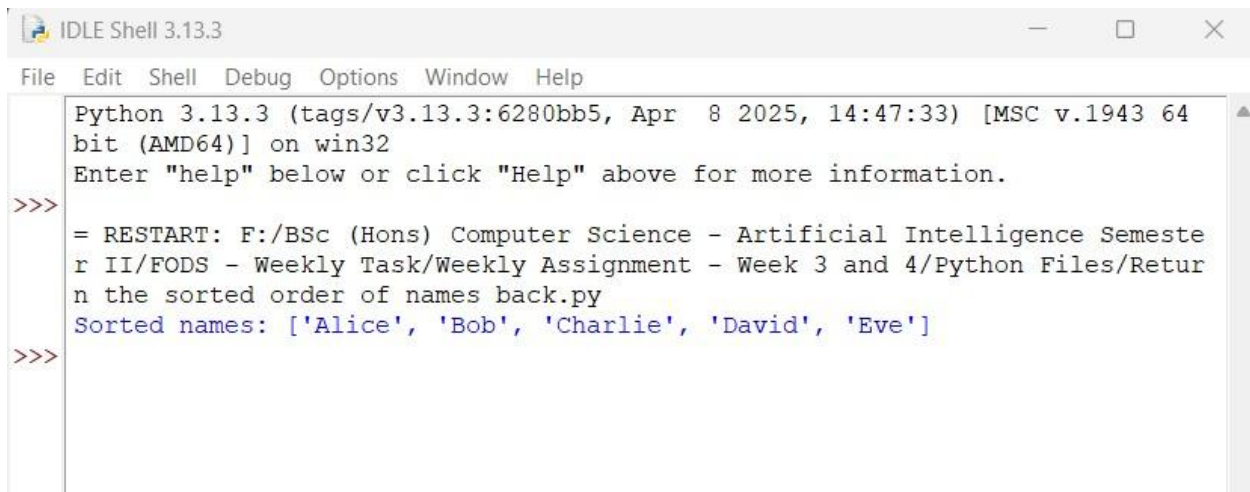
*Return the sorted order of names back.py - F:/BSc (Hons) Computer Science - Artificial Intel...
File Edit Format Run Options Window Help
def sort_names(names):
    """Return the sorted order of names."""
    return sorted(names)

# Test the function
if __name__ == "__main__":
    # Example list of names
    name_list = ["Alice", "Charlie", "Bob", "Eve", "David"]

    # Sort the names
    sorted_names = sort_names(name_list)

    # Display the sorted names
    print("Sorted names:", sorted_names)

```

**Output obtained in execution:**


```

IDLE Shell 3.13.3
File Edit Shell Debug Options Window Help
Python 3.13.3 (tags/v3.13.3:6280bb5, Apr 8 2025, 14:47:33) [MSC v.1943 64
bit (AMD64)] on win32
Enter "help" below or click "Help" above for more information.
>>>
= RESTART: F:/BSc (Hons) Computer Science - Artificial Intelligence Semeste
r II/FODS - Weekly Task/Weekly Assignment - Week 3 and 4/Python Files/Retur
n the sorted order of names back.py
Sorted names: ['Alice', 'Bob', 'Charlie', 'David', 'Eve']
>>>

```

Python Program File: “Return the sorted order of names back.py.”

**Explanation of code:****Function Definition:**

```
def sort_names(names):
```

The function 'sort\_names' takes a single parameter 'names', which is expected to a list of names in form of strings.

**Sorting the Names:**

The built-in 'sorted()' function is used to sort the list of names. The function returns a new list containing all items from the original list in ascending order (alphabetically).

**Testing the Function:**

The 'if \_\_name\_\_ == "\_\_main\_\_":' block ensures that code runs only when the program is executed directly.

**Displaying the Sorted Names:**

The 'sort\_names' function is called with the list, and the sorted result is stored in 'sorted\_names'. The sorted names are printed to the console.

**Conclusion of the Program:**

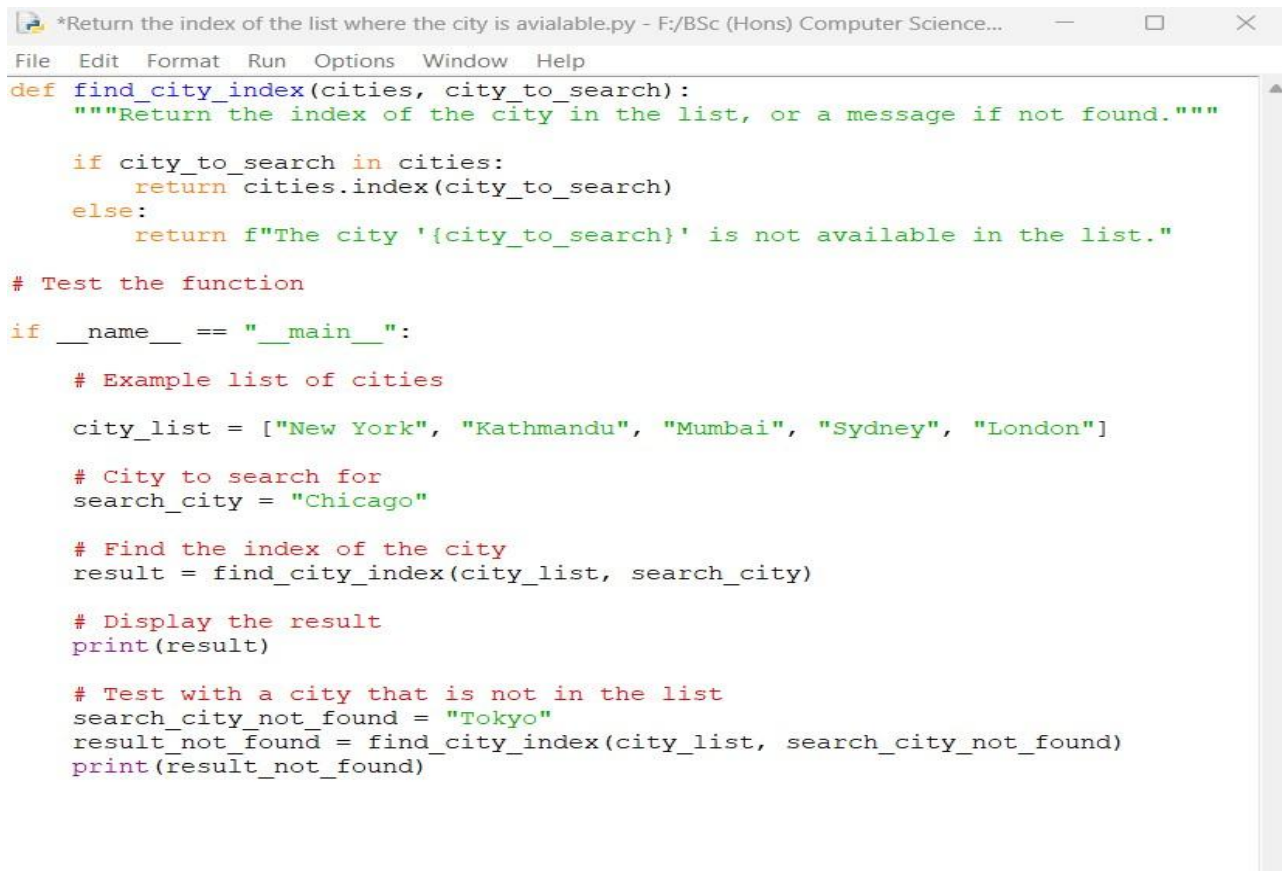
The given python program is an example of how to sort a list of names in Python. The use of function 'sorted()' simplifies the sorting process, making the code concise and efficient. The program can be easily modified to accept input from the user or handle different types of data.

6. Write a function to accept 2 parameters, one is the list of cities and another is the city that the user wants to search. The function should search the city in the list of cities and return the index of the list where the city is available. If the city is not available, the program should return a proper message.

### Answer:

The given python program defines a function to accept a list of cities and a city name to search for. The function returns the index of the city in the list if it exists, or a message indicating that the city is found.

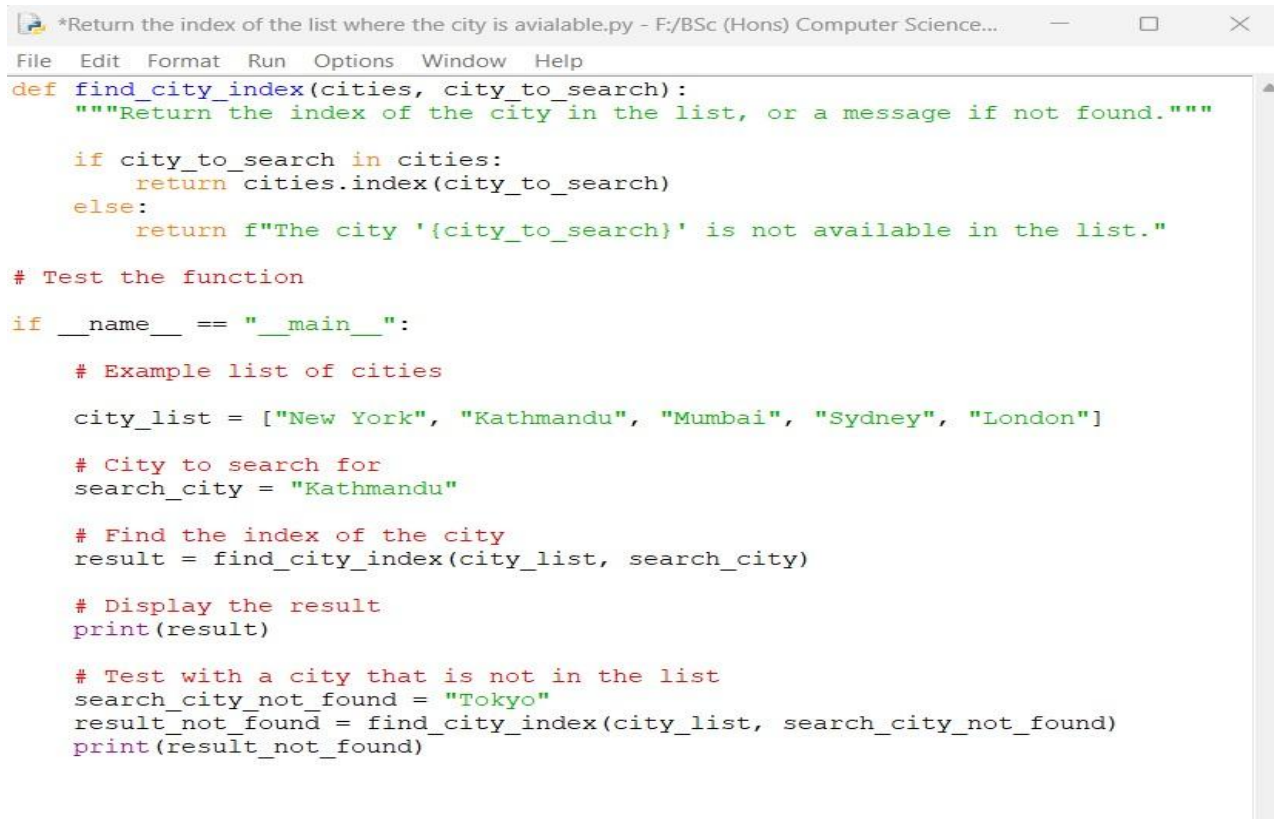
### Following code for input:

A screenshot of a Python IDE window titled '\*Return the index of the list where the city is available.py - F:/BSc (Hons) Computer Science...'. The window contains a Python script with the following code:

```
def find_city_index(cities, city_to_search):  
    """Return the index of the city in the list, or a message if not found."""  
  
    if city_to_search in cities:  
        return cities.index(city_to_search)  
    else:  
        return f"The city '{city_to_search}' is not available in the list."  
  
# Test the function  
  
if __name__ == "__main__":  
    # Example list of cities  
    city_list = ["New York", "Kathmandu", "Mumbai", "Sydney", "London"]  
  
    # City to search for  
    search_city = "Chicago"  
  
    # Find the index of the city  
    result = find_city_index(city_list, search_city)  
  
    # Display the result  
    print(result)  
  
    # Test with a city that is not in the list  
    search_city_not_found = "Tokyo"  
    result_not_found = find_city_index(city_list, search_city_not_found)  
    print(result_not_found)
```



Continue:



```

*Return the index of the list where the city is available.py - F:/BSc (Hons) Computer Science...
File Edit Format Run Options Window Help
def find_city_index(cities, city_to_search):
    """Return the index of the city in the list, or a message if not found."""
    if city_to_search in cities:
        return cities.index(city_to_search)
    else:
        return f"The city '{city_to_search}' is not available in the list."

# Test the function

if __name__ == "__main__":
    # Example list of cities
    city_list = ["New York", "Kathmandu", "Mumbai", "Sydney", "London"]

    # City to search for
    search_city = "Kathmandu"

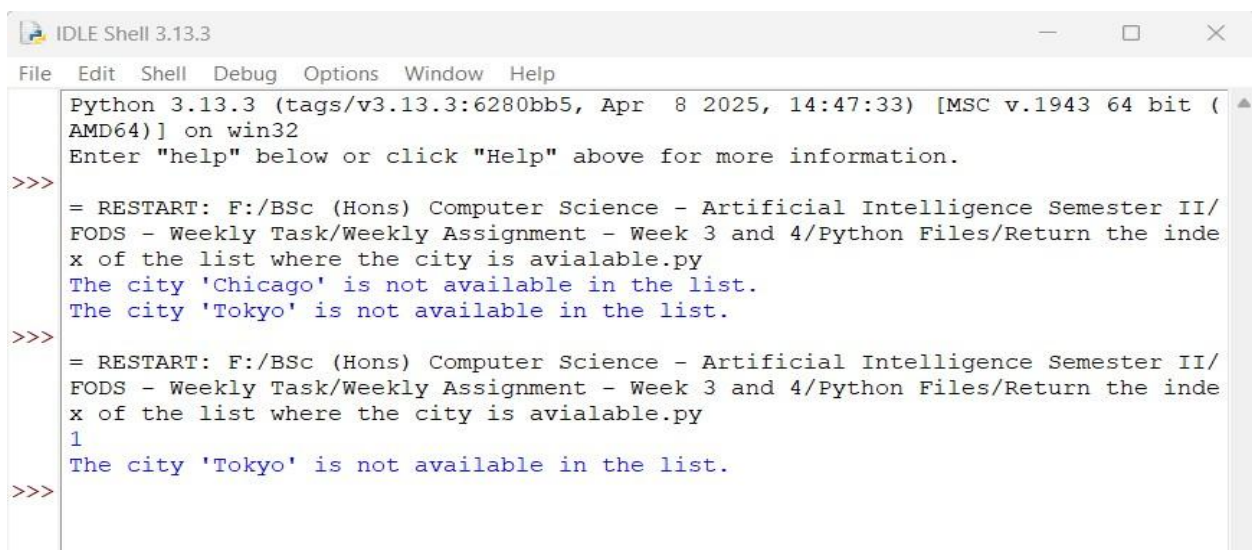
    # Find the index of the city
    result = find_city_index(city_list, search_city)

    # Display the result
    print(result)

    # Test with a city that is not in the list
    search_city_not_found = "Tokyo"
    result_not_found = find_city_index(city_list, search_city_not_found)
    print(result_not_found)

```

Output obtained in execution:



```

IDLE Shell 3.13.3
File Edit Shell Debug Options Window Help
Python 3.13.3 (tags/v3.13.3:6280bb5, Apr 8 2025, 14:47:33) [MSC v.1943 64 bit (AMD64)] on win32
Enter "help" below or click "Help" above for more information.

>>> = RESTART: F:/BSc (Hons) Computer Science - Artificial Intelligence Semester II/
FODS - Weekly Task/Weekly Assignment - Week 3 and 4/Python Files/Return the index of the list where the city is available.py
The city 'Chicago' is not available in the list.
The city 'Tokyo' is not available in the list.

>>> = RESTART: F:/BSc (Hons) Computer Science - Artificial Intelligence Semester II/
FODS - Weekly Task/Weekly Assignment - Week 3 and 4/Python Files/Return the index of the list where the city is available.py
1
The city 'Tokyo' is not available in the list.

>>>

```

Python Program File: “Return the index of list where the city is available.py.”

### **Explanation of code:**

Function Definition:

```
def find_city_index(cities, city_to_search):
```

The function ‘find\_city\_index’ takes two parameters: ‘cities’, which is a list of city names, and ‘city\_to\_search’, which is the name of the city the user wants to find.

Searching for the City:

The function checks if ‘city\_to\_search’ is present in the ‘cities’ list using the ‘in’ keyword. If the required city is found, the function returns its index using ‘index()’ method.

Handling City Not Found:

If the city is not found in the list, the function returns a message indicating that the city is not available.

Testing the Function:

The ‘if \_\_name\_\_ == “\_\_main\_\_”:' block ensures that the test code runs only when the script is executed directly. An example list of cities is created to demonstrate the function.

Displaying the Result:

The ‘find\_ciy\_index’ function is called with the example list and a city to search for, and the result is printed. The program also texts the function with a city that is not in the list to demonstrate the handling of that case.

### Conclusion of the Program:

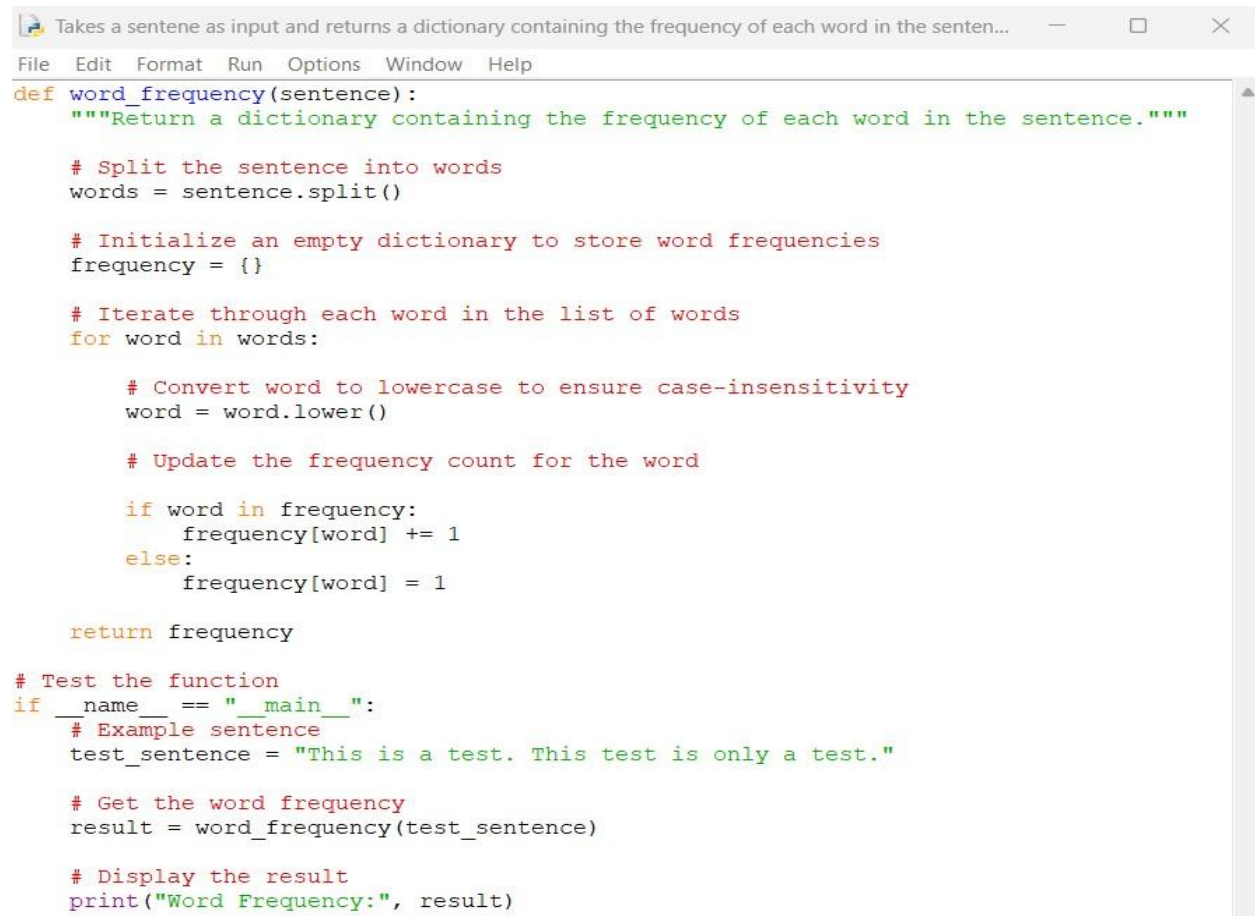
The given program effectively demonstrates how to search for a city in a list and return its index or a message if the city is found. The use of list methods like 'in' and 'index()' makes the implementation straightforward and efficient. The program can be easily modified to accept user input for the list of cities and the city to search, enhancing its usability.

7. Write a function `word_frequency(sentence)` that takes a sentence as input and returns a dictionary containing the frequency of each word in the sentence. [Hint: split the sentence into words and iterate to check the word frequency.]

### Answer:

The given python program below defines a function '`word_frequency(sentence)`' to calculate the frequency of each word in a given sentence. The function splits the sentence into words and returns a dictionary containing the word counts.

## Following code for input:



```

def word_frequency(sentence):
    """Return a dictionary containing the frequency of each word in the sentence."""

    # Split the sentence into words
    words = sentence.split()

    # Initialize an empty dictionary to store word frequencies
    frequency = {}

    # Iterate through each word in the list of words
    for word in words:

        # Convert word to lowercase to ensure case-insensitivity
        word = word.lower()

        # Update the frequency count for the word
        if word in frequency:
            frequency[word] += 1
        else:
            frequency[word] = 1

    return frequency

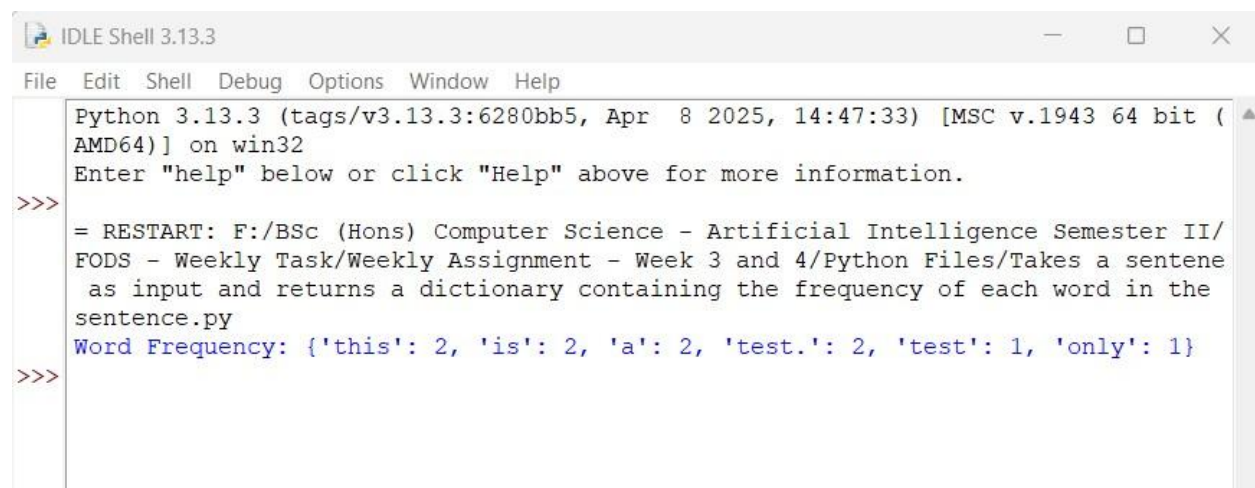
# Test the function
if __name__ == "__main__":
    # Example sentence
    test_sentence = "This is a test. This test is only a test."

    # Get the word frequency
    result = word_frequency(test_sentence)

    # Display the result
    print("Word Frequency:", result)

```

## Output obtained in execution:



```

Python 3.13.3 (tags/v3.13.3:6280bb5, Apr 8 2025, 14:47:33) [MSC v.1943 64 bit (AMD64)] on win32
Enter "help" below or click "Help" above for more information.

>>> = RESTART: F:/BSc (Hons) Computer Science - Artificial Intelligence Semester II/
FODS - Weekly Task/Weekly Assignment - Week 3 and 4/Python Files/Takes a sentene
as input and returns a dictionary containing the frequency of each word in the
sentence.py
Word Frequency: {'this': 2, 'is': 2, 'a': 2, 'test.': 2, 'test': 1, 'only': 1}
>>>

```

Python Program File: “Takes a sentence as input and returns a dictionary containing the frequency of each word in the sentence.py.”

### **Explanation of code:**

Function Definition:

```
def word_frequeuncy(sentence):
```

The function ‘word\_frequency’ takes a single parameter ‘sentence’, which is expected to be a string.

Splitting the Sentence:

```
words = sentence.split()
```

The ‘split()’ method is used to break the sentence into a list of words based on whitespace.

Initializing the Frequency Dictionary:

An empty dictionary ‘frequency’ is initialized to store the count of each word.

Iterating Through Words:

The function iterates through each word in the list. Then, each word is converted to lowercase to ensure that the counting is case-insensitive (e.g., “This” and “this” are treated as the same word in the program).

Updating Word Counts:

If the word is already a key in the ‘frequency’ dictionary, its count is incremented by 1. If the word is not in the present dictionary, it is added with a count of 1.

Returning the Result:

The function returns the ‘frequency’ dictionary containing the word count after processing all words.

Testing the Function:

The 'if `__name__` == "`__main__`":' block ensures that the test code runs only when the script is executed directly.

Displaying the Result:

The 'word\_frequency' function is called with example sentence, and the result is printed.

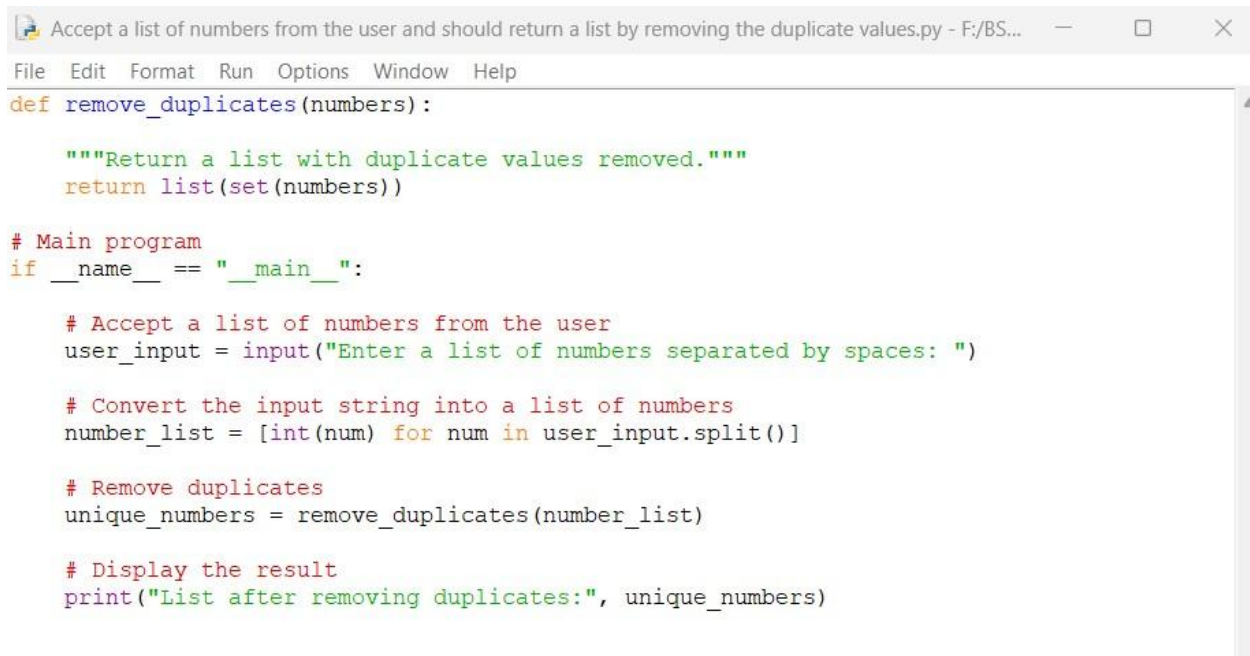
Conclusion of the Program:

The given python program is an example of how to calculate the frequency of each word in a sentence using a dictionary.

8. Write a program to accept a list of numbers from the user and should return a list by removing the duplicate values, if any.

**Answer:**

The given python program below accepts a list of numbers from the user and returns a new list with duplicate values removed. The program uses a set to eliminate duplicate values, as sets do not allow duplicate values.

**Following code for input:**


```

Accept a list of numbers from the user and should return a list by removing the duplicate values.py - F:/BS...
File Edit Format Run Options Window Help
def remove_duplicates(numbers):

    """Return a list with duplicate values removed."""
    return list(set(numbers))

# Main program
if __name__ == "__main__":

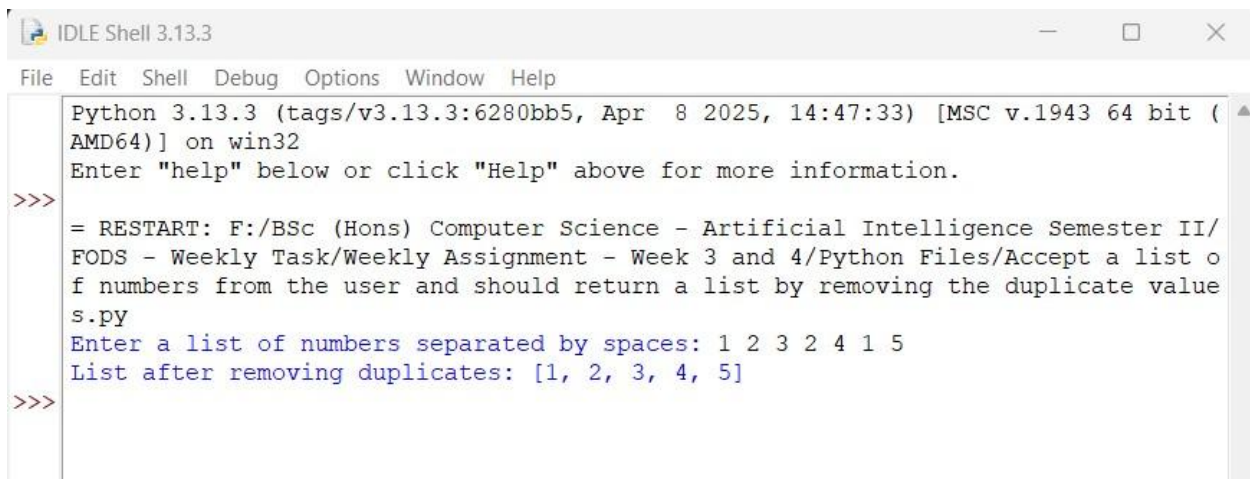
    # Accept a list of numbers from the user
    user_input = input("Enter a list of numbers separated by spaces: ")

    # Convert the input string into a list of numbers
    number_list = [int(num) for num in user_input.split()]

    # Remove duplicates
    unique_numbers = remove_duplicates(number_list)

    # Display the result
    print("List after removing duplicates:", unique_numbers)

```

**Output obtained in execution:**


```

IDLE Shell 3.13.3
File Edit Shell Debug Options Window Help
Python 3.13.3 (tags/v3.13.3:6280bb5, Apr 8 2025, 14:47:33) [MSC v.1943 64 bit (AMD64)] on win32
Enter "help" below or click "Help" above for more information.

>>> = RESTART: F:/BSc (Hons) Computer Science - Artificial Intelligence Semester II/
FODS - Weekly Task/Weekly Assignment - Week 3 and 4/Python Files/Accept a list o
f numbers from the user and should return a list by removing the duplicate value
s.py
Enter a list of numbers separated by spaces: 1 2 3 2 4 1 5
List after removing duplicates: [1, 2, 3, 4, 5]

>>>

```

Python Program File: "Accept a list of numbers from the user and should return a list by removing the duplicate values.py."

**Explanation of code:**

## Function Definition:

```
def remove_duplicates(numbers):
```

The function 'remove\_duplicates' takes a single parameter 'numbers', which is expected to be a list of numbers.

## Removing Duplicates:

```
    return list(set(numbers))
```

The function converts the list of numbers into a set using 'set(numbers)' which automatically removes any duplicate values. Then, the set is converted back into a list using 'list()', and this new list is returned.

## Main Program:

The 'if \_\_name\_\_ == "\_\_main\_\_":' block ensures that the following code runs only when the program is executed directly.

## User Input:

The program prompts the user to enter a list of numbers, separated by spaces.

## Converting Input to List of Numbers:

The input string is split into individual string representations of numbers using 'split()'. A comprehension form of list is used to convert each string into an integer, resulting in a list of numbers.

## Removing Duplicates:

The 'remove\_duplicates' function is called with the list of numbers, and the result is stored in 'unique\_numbers'.



Displaying the Result:

The program prints the list of unique numbers.

Conclusion of the Program:

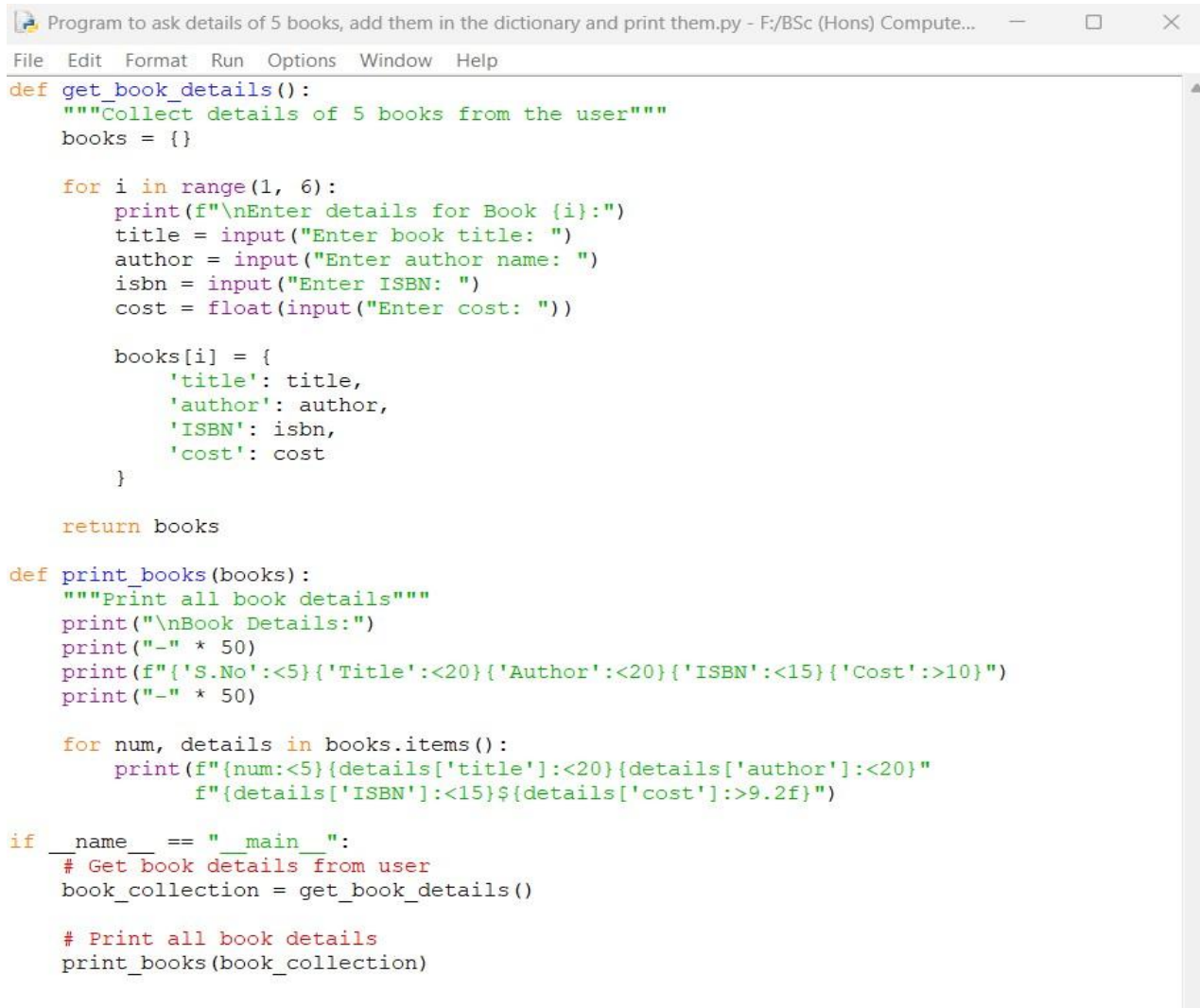
The given python program is an example of how to remove duplicate values from a list of numbers provided by the user. There is use of set for duplicate removal which makes the implementation efficient. It can easily be modified to handle different types of input or to include additional features such as sorting.

9. Write a program to ask the details of 5 books (title, author, ISBN, cost), add them in the dictionary and print them.

Answer:

The given python program below collects details of 5 books (title, author, ISBN, cost) from the user, stores them in a dictionary, and then prints all the book details.

## Following code for input:



```

Program to ask details of 5 books, add them in the dictionary and print them.py - F:/BSc (Hons) Compute...
File Edit Format Run Options Window Help

def get_book_details():
    """Collect details of 5 books from the user"""
    books = {}

    for i in range(1, 6):
        print(f"\nEnter details for Book {i}:")
        title = input("Enter book title: ")
        author = input("Enter author name: ")
        isbn = input("Enter ISBN: ")
        cost = float(input("Enter cost: "))

        books[i] = {
            'title': title,
            'author': author,
            'ISBN': isbn,
            'cost': cost
        }

    return books

def print_books(books):
    """Print all book details"""
    print("\nBook Details:")
    print("-" * 50)
    print(f"{'S.No':<5}{'Title':<20}{'Author':<20}{'ISBN':<15}{'Cost':>10}")
    print("-" * 50)

    for num, details in books.items():
        print(f"{num:<5}{details['title']:<20}{details['author']:<20}"
              f"{details['ISBN']:<15}${details['cost']:>9.2f}")

if __name__ == "__main__":
    # Get book details from user
    book_collection = get_book_details()

    # Print all book details
    print_books(book_collection)

```

## Output obtained in execution:

```

Python 3.13.3 (tags/v3.13.3:6280bb5, Apr 8 2025, 14:47:33) [MSC v.1943 64 bit (AMD64)] on win32
Enter "help" below or click "Help" above for more information.
>>>
= RESTART: F:\BSc (Hons) Computer Science - Artificial Intelligence Semester II\FODS - Weekly Task
\Weekly Assignment - Week 3 and 4\Python Files\Program to ask details of 5 books, add them in the
dictionary and print them.py

Enter details for Book 1:
Enter book title: Python Programming
Enter author name: John Smith
Enter ISBN: 9780123456789
Enter cost: 50.00

Enter details for Book 2:
Enter book title: Data Science Basics
Enter author name: Jane Doe
Enter ISBN: 9789976543210
Enter cost: 59.95

Enter details for Book 3:
Enter book title: Exploring Grammar
Enter author name: Ronald Carter
Enter ISBN: 9789547493932
Enter cost: 44.99

Enter details for Book 4:
Enter book title: Introduction to Algorithms
Enter author name: Aditya Bhargava
Enter ISBN: 9785949493942
Enter cost: 65.00

Enter details for Book 5:
Enter book title: Javascript for Kids : A playful Introduction to Programming
Enter author name: Nick Morgan
Enter ISBN: 97589035954049
Enter cost: 69.99

Book Details:
-----
S.No Title Author ISBN Cost
-----
1 Python Programming John Smith 9780123456789 $ 50.00
2 Data Science Basics Jane Doe 9789976543210 $ 59.95
3 Exploring Grammar Ronald Carter 9789547493932 $ 44.99
4 Introduction to Algorithms Aditya Bhargava 9785949493942 $ 65.00
5 Javascript for Kids : A playful Introduction to Programming Nick Morgan 97589035954049 $ 69.99
>>>

```

Python Program File: “Program to ask the details of 5 books, add them in the dictionary and print them.py.”

**Explanation of code:****Function Definitions:**

The program defines two functions: `'get_book_details()'` and `'print_books(books)'`.

**1. get\_book\_details() Function:**

This function collects the details for 5 books from the user and stores them in a dictionary.

**Initialization:**

An empty dictionary `'books'` is created to hold the book details.

**Loop:**

A `'for'` loop iterates from 1 to 5, prompting the user to enter details for each book. For each iteration, the user is prompted to enter the title, author, ISBN, and cost of the book. The cost is converted to a floating-value to ensure it can handle decimal values.

**Storing Data:**

Book's details are stored in the `'books'` dictionary using the loop index `'i'` as the key. The value is another dictionary containing the book's title, author, ISBN, and cost.

**Return Value:**

The function returns the `'books'` dictionary.

**2. print\_books(books) Function**

This function takes the `'books'` dictionary as input and prints the details of each book in a formatted manner.

### Header:

It prints a header for the book details, including column titles : Serial Number (S.No), Title, Author, ISBN, and Cost. The use of string alignment to ensure that the columns are properly aligned.

### Looping through Books:

The function iterates over the 'books dictionary using 'items()', which provided both the key (book number) and the value (book details). For each book in the list, it prints the details in a formatted string.

### Main Program:

```
if __name__ == "__main__":
```

This block ensures that the code runs only when the program is directly, not when imported as a module.

### Function Call:

The 'get\_book\_details()' function is called to collect book information from the user, and returned dictionary is stored in 'book\_collection'. The 'print\_books(book\_collection)' function is then called to display the collected book details.

### Conclusion of the Program:

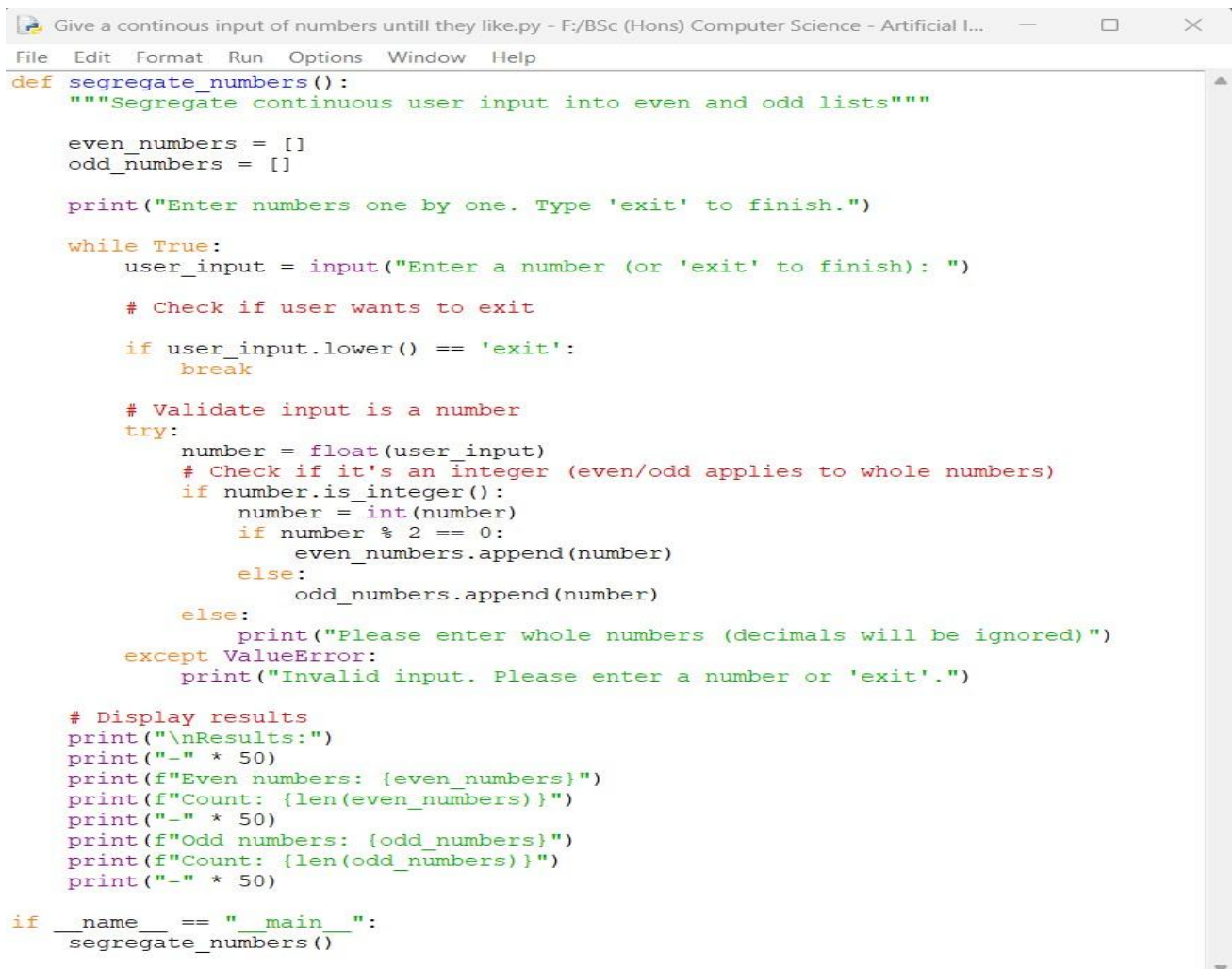
The given python program is an example of how to collect, store, and display structured data in Python, It showcases concepts such as functions, loops, dictionaries, and formatted output.

10. Create a program to ask a user to give continuous input of numbers until they like. The program should keep on segregating the user input numbers into even and odd lists separately. Once the user completes the input and opts for exiting the program, the program should display the separate list of even and odd lists in a proper format.

### Answer:

The given python program below segregates user input numbers into even and odd lists, along with the flow of execution and key components.

### Following code for input:



```

Give a continous input of numbers untill they like.py - F:/BSc (Hons) Computer Science - Artificial I...
File Edit Format Run Options Window Help
def segregate_numbers():
    """Segregate continuous user input into even and odd lists"""

    even_numbers = []
    odd_numbers = []

    print("Enter numbers one by one. Type 'exit' to finish.")

    while True:
        user_input = input("Enter a number (or 'exit' to finish): ")

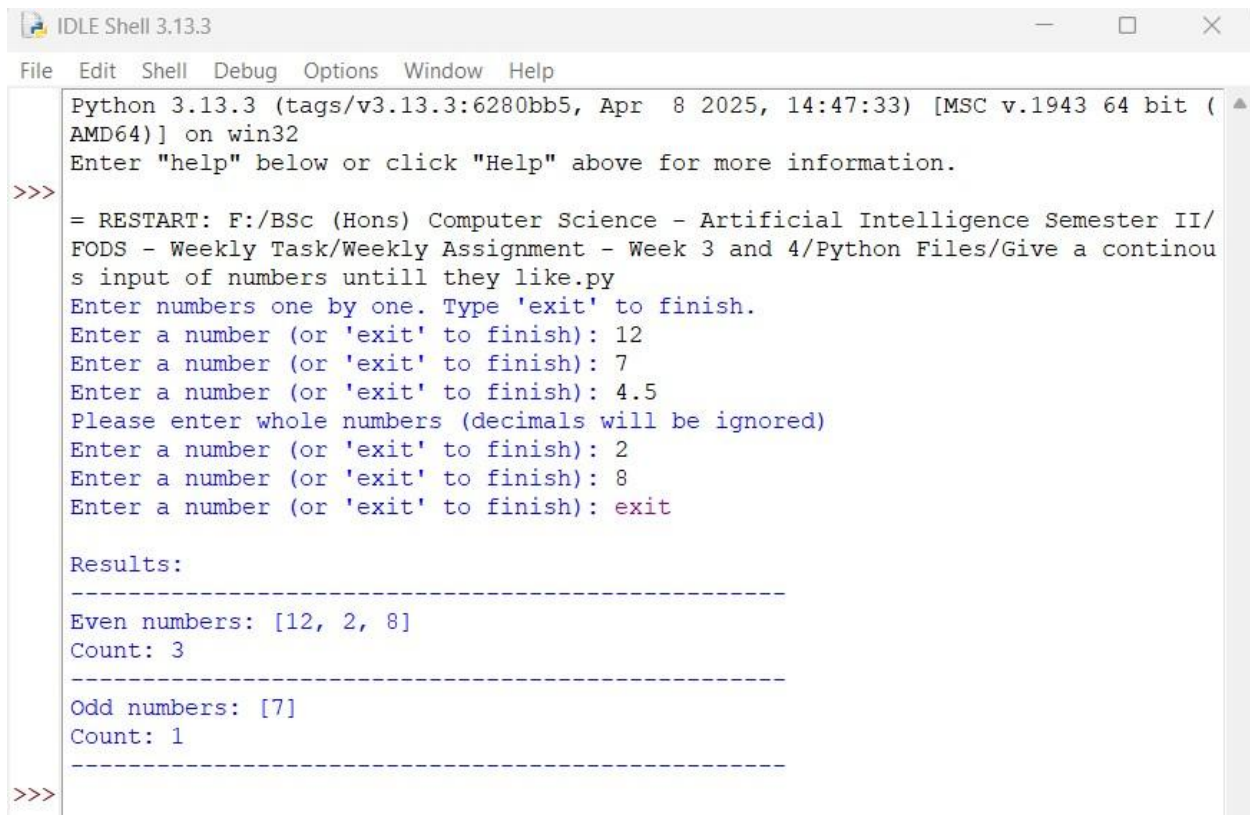
        # Check if user wants to exit
        if user_input.lower() == 'exit':
            break

        # Validate input is a number
        try:
            number = float(user_input)
            # Check if it's an integer (even/odd applies to whole numbers)
            if number.is_integer():
                number = int(number)
                if number % 2 == 0:
                    even_numbers.append(number)
                else:
                    odd_numbers.append(number)
            else:
                print("Please enter whole numbers (decimals will be ignored)")
        except ValueError:
            print("Invalid input. Please enter a number or 'exit'.")

    # Display results
    print("\nResults:")
    print("-" * 50)
    print(f"Even numbers: {even_numbers}")
    print(f"Count: {len(even_numbers)}")
    print("-" * 50)
    print(f"Odd numbers: {odd_numbers}")
    print(f"Count: {len(odd_numbers)}")
    print("-" * 50)

if __name__ == "__main__":
    segregate_numbers()

```

**Output obtained in execution:**


```

Python 3.13.3 (tags/v3.13.3:6280bb5, Apr 8 2025, 14:47:33) [MSC v.1943 64 bit (AMD64)] on win32
Enter "help" below or click "Help" above for more information.
>>>
= RESTART: F:/BSc (Hons) Computer Science - Artificial Intelligence Semester II/
FODS - Weekly Task/Weekly Assignment - Week 3 and 4/Python Files/Give a continuou
s input of numbers untill they like.py
Enter numbers one by one. Type 'exit' to finish.
Enter a number (or 'exit' to finish): 12
Enter a number (or 'exit' to finish): 7
Enter a number (or 'exit' to finish): 4.5
Please enter whole numbers (decimals will be ignored)
Enter a number (or 'exit' to finish): 2
Enter a number (or 'exit' to finish): 8
Enter a number (or 'exit' to finish): exit

Results:
-----
Even numbers: [12, 2, 8]
Count: 3
-----
Odd numbers: [7]
Count: 1
-----
>>>

```

Python Program File: “Give a continuous input of numbers until they like.py.”

**Explanation of code:**

Function Definition:

```
def segregate_numbers():
```

The function ‘segregate\_numbers()’ is defined to encapsulate the logic for collecting and processing user input.

Initialization:

There are two empty lists, ‘even\_number’ and ‘odd\_number’, are initialized to store the respective numbers as they are entered by the user.

### User Instructions:

A message is printed to inform the user how to interact with the program.

### Continuous Input Loop:

A 'while True' loop is initiated to allow the continuous input from the user. The loop will run indefinitely until the user types 'exit'.

### Exit Condition:

The program checks if the user input is 'exit'. If so, the loop is terminated using 'break()' statement.

### Input Validation:

A 'try' block is used to attempt converting the user input to a float value. This allows the program to handle numeric input, including decimal numbers.

### Whole Number Check:

The program checks if the number is a whole number using 'is\_integer()'. If it is, then it converts the float to an integer.

### Segregation Logic:

The program uses the modulus operator (%) to determine if the number is even or odd number. If 'number % 2 == 0', the number is even, and it is added to the 'even\_numbers' list. Otherwise, it is an odd number and added to the 'odd\_numbers' list.

### Handling Invalid Input:

If the input cannot be converted to a float value, a 'ValueError' is raised, and an error message is printed.



### Main Program Execution:

If `__name__ == "__main__"`: block ensures that the 'segregate\_numbers()' function is called only when the script is executed directly, no when imported as a module.

### Output of the Program:

After the user exist the loop, the program prints the results. It displays the list of even numbers and their count. It also displays the list of odd numbers and their count. The output is formatted with separators.

### Conclusion of the Program:

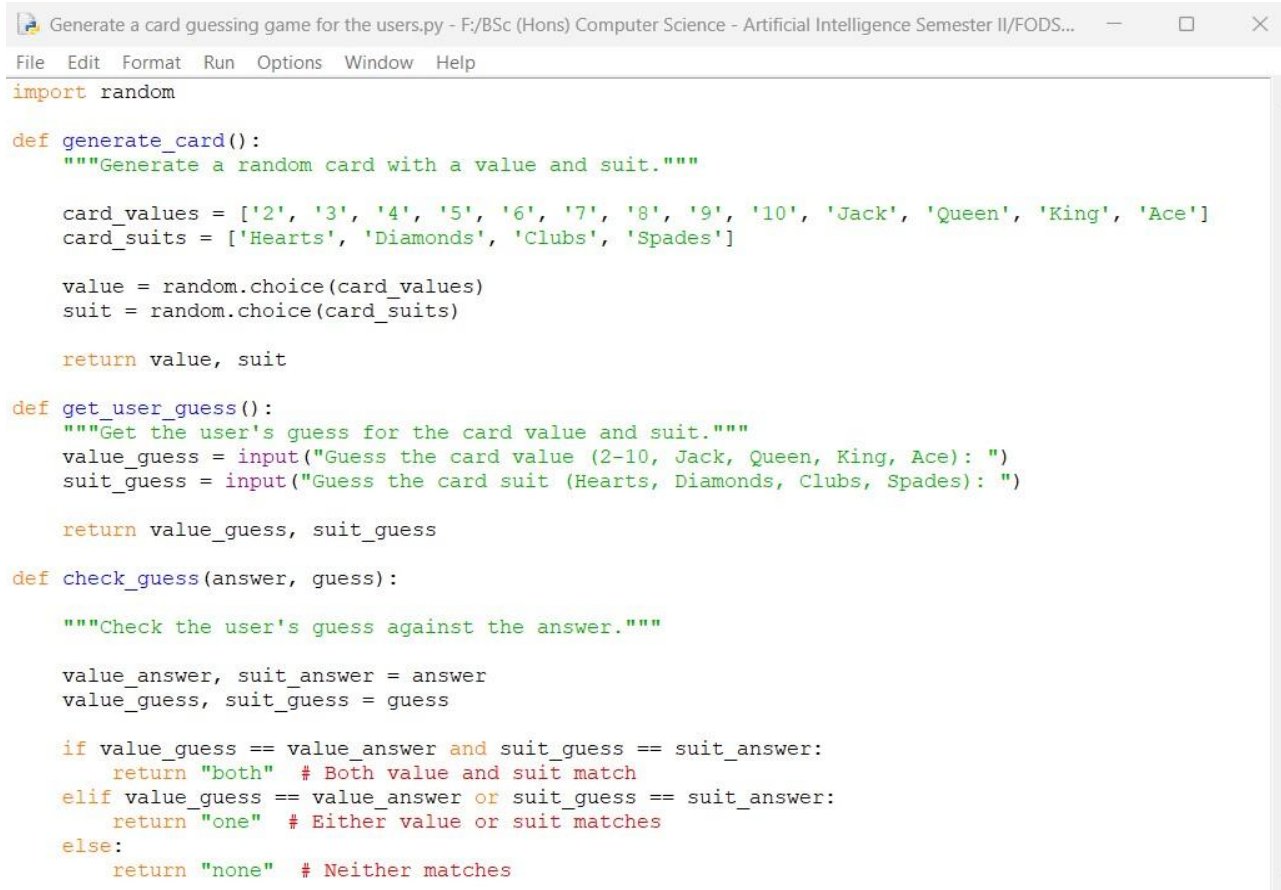
The given python program is an example of how to handle user input, perform data segregation, and display results in python. It shows concepts such as loops, conditionals, error handling, and list operations.

11. Write a program to generate a card guessing game for the users in an interesting way. The card should have property such as name and value (e.g. ace 10). Specifications are as mentioned below.

- I. The program should have a list of card values like 2, 3, 4,..., Jack, Queen, King, Ace
- II. The program should have a list of card suits like heart, diamond, club, spades.
- III. The program should randomly pick up a number and a suit and keep as an answer in a separate list.
- IV. The program should ask the player to guess the card value and the suit.
- V. The program should check the player guessed value with the computer answer value. If both the parts don't match, the program should show a broken heart and game over to the player. If any one of the part of answer matches, the program should show a smiley face to the player. If both the guesses of the player match with the program answer, the program should show a heart and a smiley face to the user.

**Answer:**

The given python program below implements a card guessing game based on user specifications. The program randomly selects a card value and suit, and then prompts the user to guess them. It provided different feedback depending on the user's guesses.

**Following code for input:**


```

Generate a card guessing game for the users.py - F:/BSc (Hons) Computer Science - Artificial Intelligence Semester II/FODS...
File Edit Format Run Options Window Help
import random

def generate_card():
    """Generate a random card with a value and suit."""

    card_values = ['2', '3', '4', '5', '6', '7', '8', '9', '10', 'Jack', 'Queen', 'King', 'Ace']
    card_suits = ['Hearts', 'Diamonds', 'Clubs', 'Spades']

    value = random.choice(card_values)
    suit = random.choice(card_suits)

    return value, suit

def get_user_guess():
    """Get the user's guess for the card value and suit."""
    value_guess = input("Guess the card value (2-10, Jack, Queen, King, Ace): ")
    suit_guess = input("Guess the card suit (Hearts, Diamonds, Clubs, Spades): ")

    return value_guess, suit_guess

def check_guess(answer, guess):

    """Check the user's guess against the answer."""

    value_answer, suit_answer = answer
    value_guess, suit_guess = guess

    if value_guess == value_answer and suit_guess == suit_answer:
        return "both" # Both value and suit match
    elif value_guess == value_answer or suit_guess == suit_answer:
        return "one" # Either value or suit matches
    else:
        return "none" # Neither matches

```

Continue:

```
def display_result(result):
    """Display the result based on the user's guess."""
    if result == "both":
        print("❤️ 🎉 Congratulations! You guessed the card correctly!")
    elif result == "one":
        print("🎉 You guessed one part correctly!")
    else:
        print("❤️ Game Over! Better luck next time!")

def play_game():
    """Main function to play the card guessing game."""
    print("Welcome to the Card Guessing Game!")

    # Generate a random card
    answer = generate_card()
    print("A card has been drawn. Can you guess it?")

    # Get user's guess
    guess = get_user_guess()

    # Check the guess
    result = check_guess(answer, guess)

    # Display the result
    display_result(result)

if __name__ == "__main__":
    play_game()
```

## Explanation of code:

```
IDLE Shell 3.13.3
File Edit Shell Debug Options Window Help
Python 3.13.3 (tags/v3.13.3:6280bb5, Apr 8 2025, 14:47:33) [MSC v.1943 64 bit (AMD64)] on win32
Enter "help" below or click "Help" above for more information.
>>> = RESTART: F:/BSc (Hons) Computer Science - Artificial Intelligence Semester II/FODS - Weekly Task/Weekly Assignment - Week 3 and 4/Python Files/Generate a card guessing game for the users.py
Welcome to the Card Guessing Game!
A card has been drawn. Can you guess it?
Guess the card value (2-10, Jack, Queen, King, Ace): Ace
Guess the card suit (Hearts, Diamonds, Clubs, Spades): Spades
❤️ Game Over! Better luck next time!
>>> = RESTART: F:/BSc (Hons) Computer Science - Artificial Intelligence Semester II/FODS - Weekly Task/Weekly Assignment - Week 3 and 4/Python Files/Generate a card guessing game for the users.py
Welcome to the Card Guessing Game!
A card has been drawn. Can you guess it?
Guess the card value (2-10, Jack, Queen, King, Ace): King
Guess the card suit (Hearts, Diamonds, Clubs, Spades): Diamonds
❤️ Game Over! Better luck next time!
>>>
```

Python Program File: “Generate a card guessing game for the users.py.”

### **Explanation of code:**

#### **Imports:**

The ‘random’ module is imported to allow the program to randomly select card values and suits.

#### **Function Definitions:**

‘generate\_card()’:

This function creates a random card by selecting value and suit from predefined lists. It returns a tuple containing the selected value and suit.

‘get\_user\_guess()’:

This function prompts the user to guess the card value and suit. Then, it returns the user’s guesses as a tuple.

‘check\_guess(answer, guess)’:

This function compares the user’s guesses with the randomly generated card. It returns “both” if both the values and suit match, “one” if either the value or suit matches, and “none” if neither match.

‘display\_result(result)’:

This function displays the message based on the result of the guess. It uses emojis to enhance the user experience.

### Main Game Function:

```
def play_game():
```

In this function, it shows the game flow where it includes interfaces like it welcomes the player, it generates a random card, it prompts the user for their guess, and displays the result.

### Execution Block:

This block ensures that the 'play\_game()' function is called only when the program is executed directly.

### Conclusion of the Program:

The given python program is an example of how to create a card guessing game with a engaging application that demonstrates several key programming concepts and principles.