**A red and blue logo

Description automatically generated**

**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**

**1**

****

**Bi-weekly assignment**

**Module Details**

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

**Dates**

|  |  |
| --- | --- |
| **Submission Date** | **07-July-2025** |
| **Submission Place** | **Backboard** |
| **Submission Time** | **23:59** |
| **Submission Notes** | **Submit Gitlab URL** |

**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.

**2**

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):
   1. 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.
   2. 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.

**3**

25024649

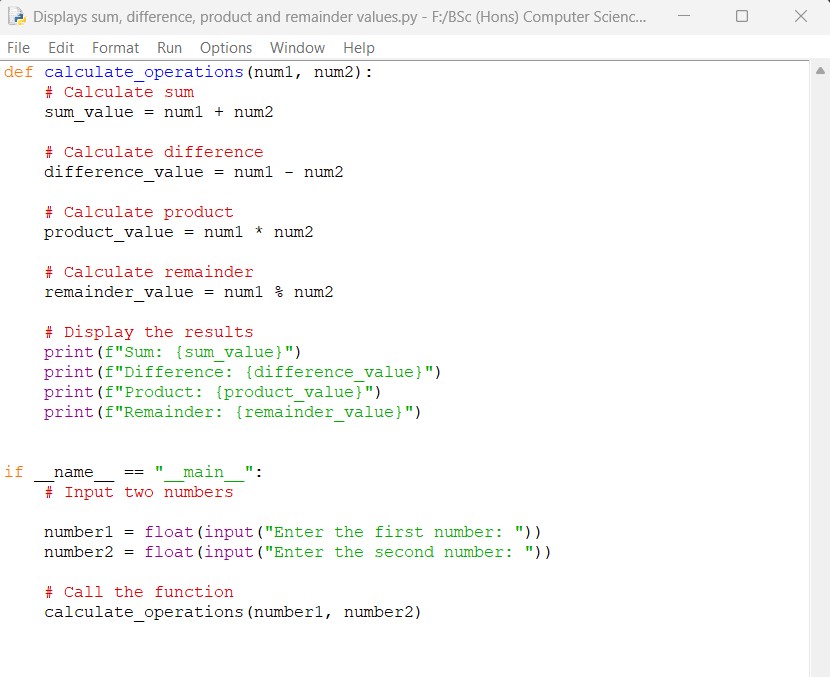
**Questions**

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.

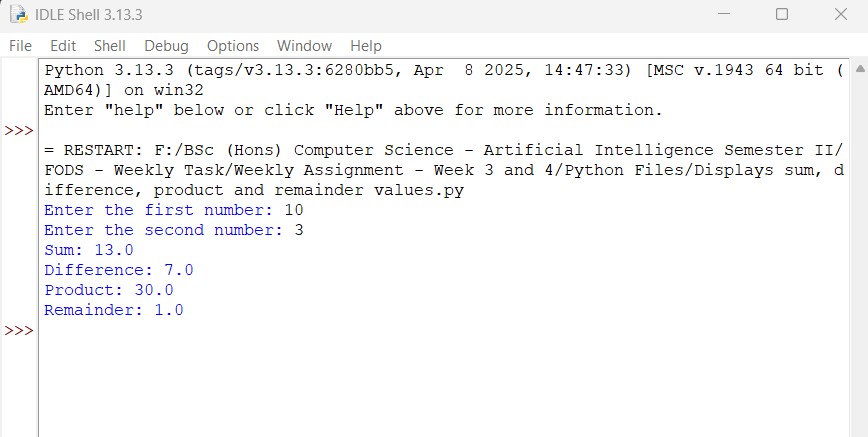
**Following code for input:**



**4**

25024649

**Output obtained in execution:**

****

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(num2, 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’.

**5**

25024649

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

**6**

25024649

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.

1. 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

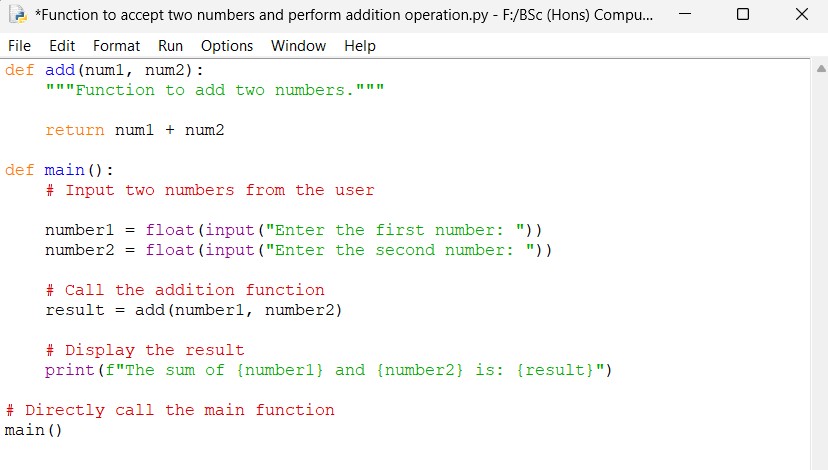
**Answer:**

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.

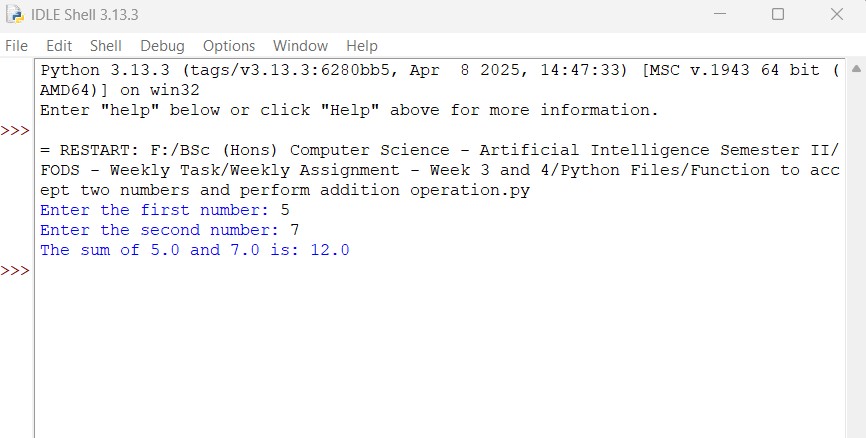
**7**

25024649

**Following code for input:**



**Output obtained in execution:**



**8**

25024649

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.

**9**

25024649

Conclusion of the Program:

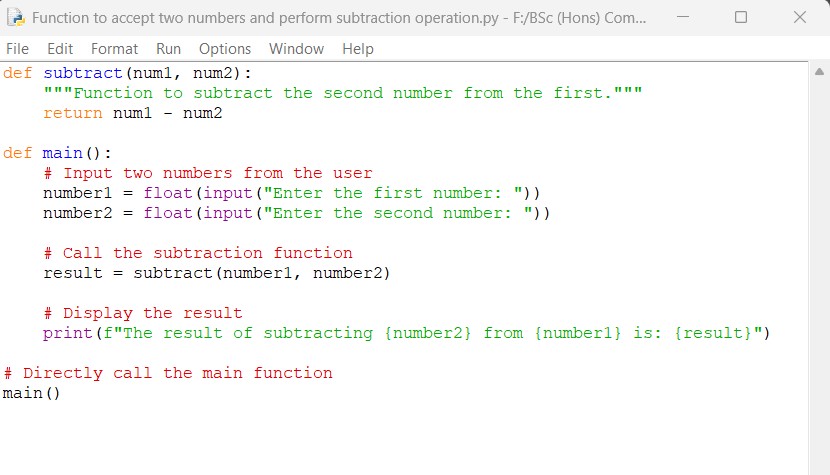
The given program demonstrates how to create a function for addition operation, accept user input, and display the result.

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.

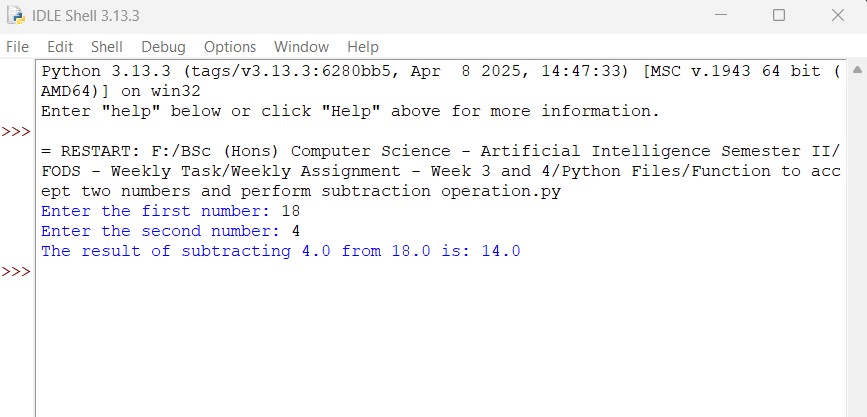
**Following code for input:**



**10**

25024649

**Output obtained in execution:**



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.

**11**

25024649

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.

**12**

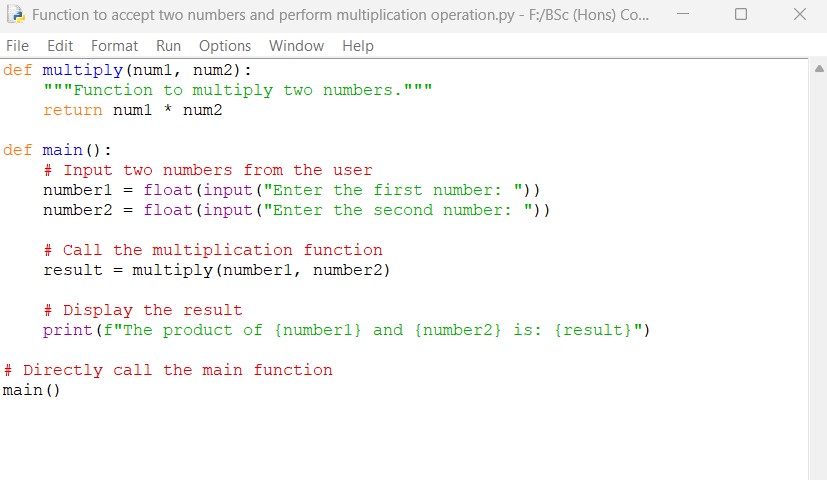
25024649

III. Multiplication

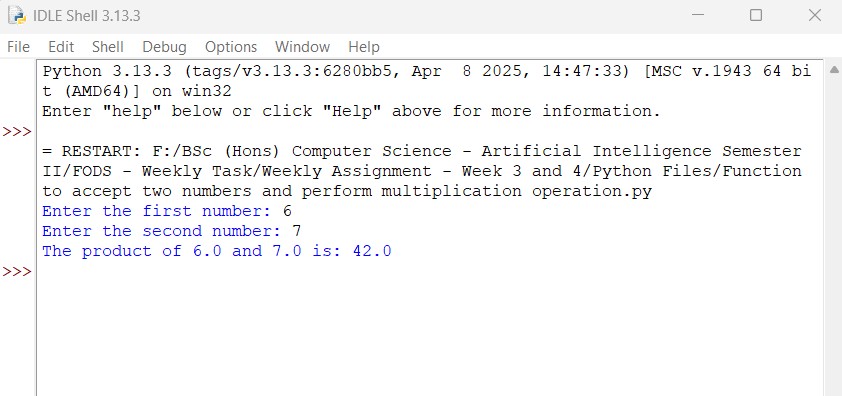
Answer:

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:**



**Output obtained in execution:**



**13**

25024649

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.

**14**

25024649

Direct Function Call:

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

Conclusion of the Program:

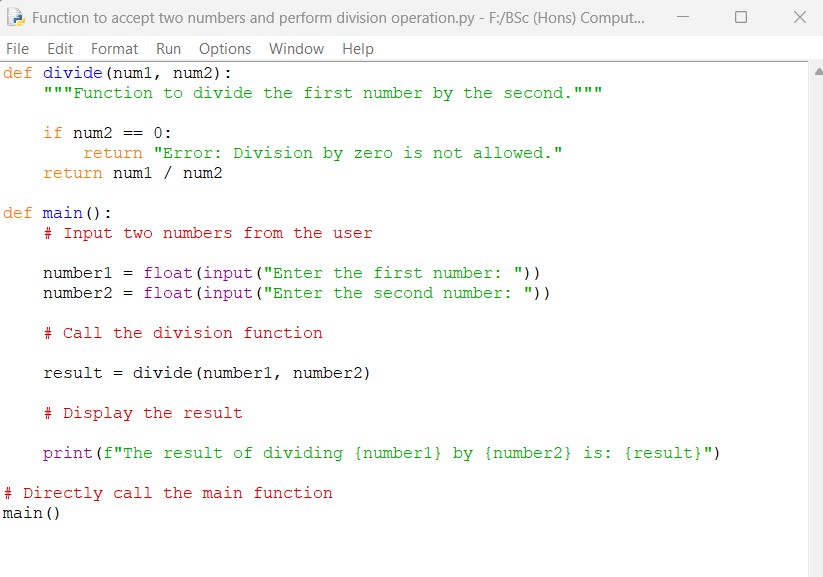
The program demonstrates how to create function for multiplication, accept the user input, and display the result in a clear format.

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.

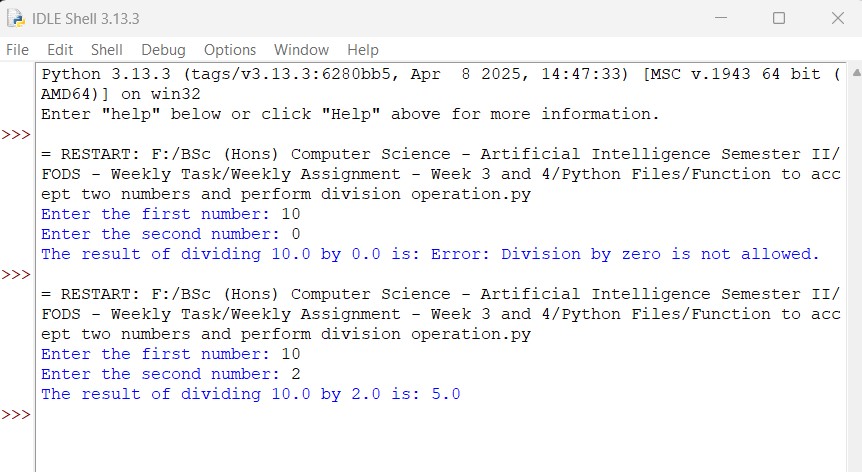
**Following code for input:**



**15**

25024649

**Output obtained in execution:**



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.

**16**

25024649

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.

**17**

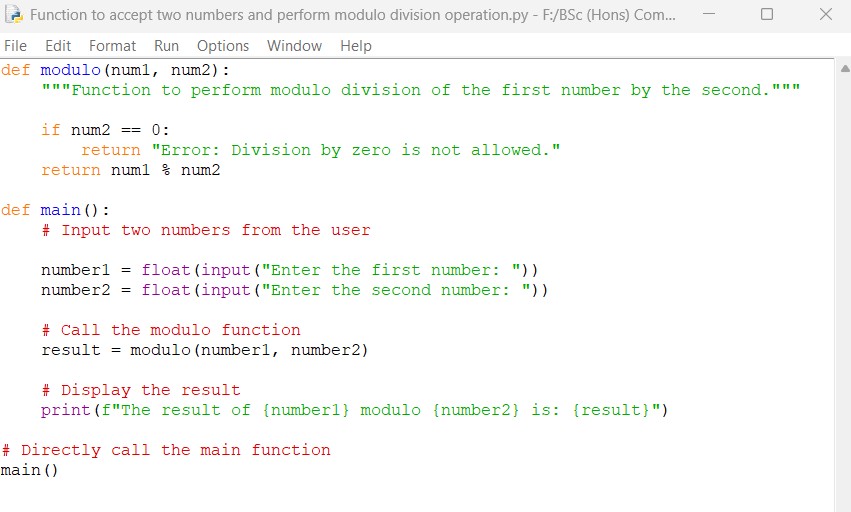
25024649

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.

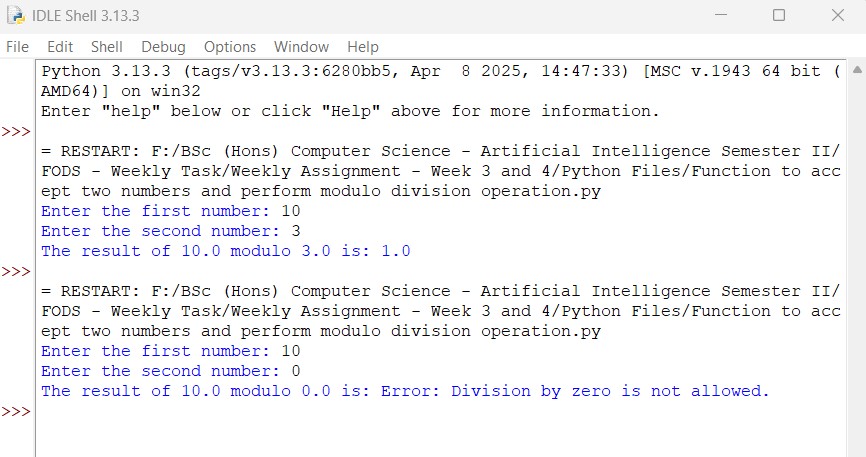
**Following code for input:**



**18**

25024649

**Output obtained in execution:**



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 and 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.

**19**

25024649

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

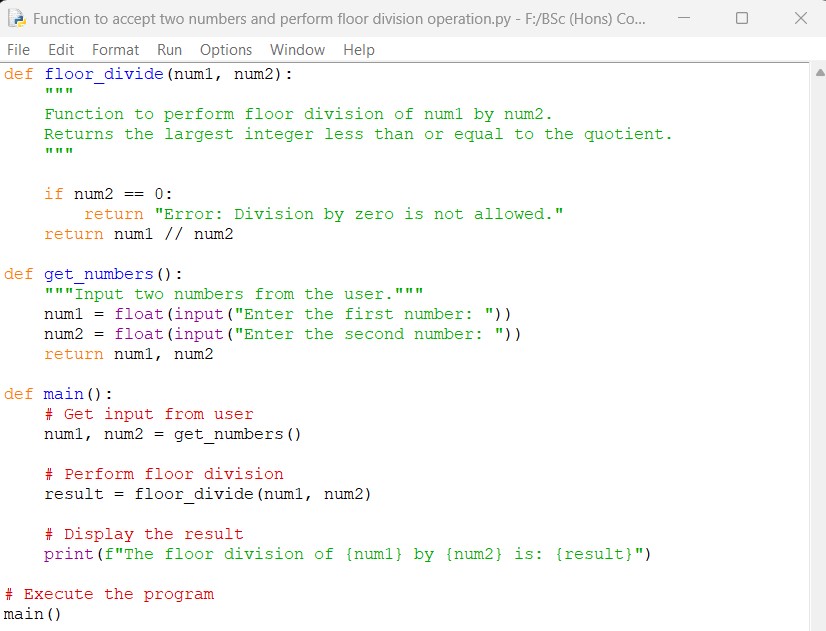
**Answer:**

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.

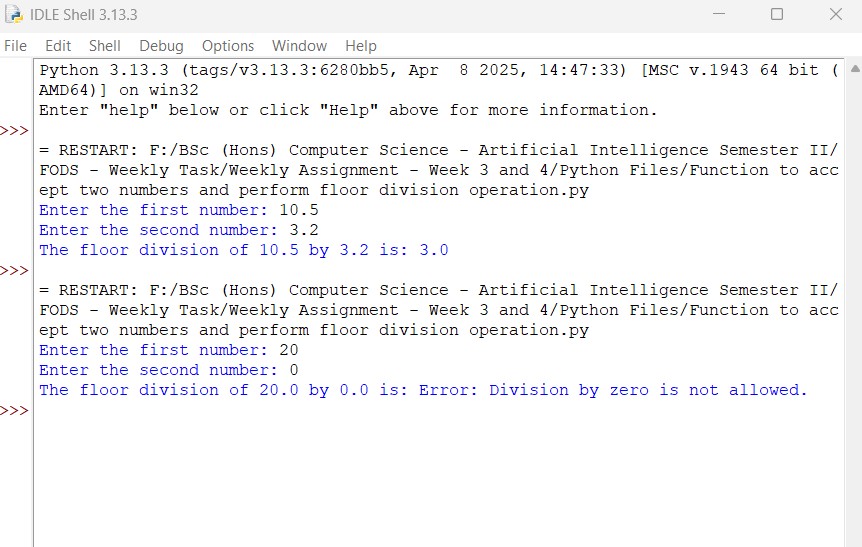
**20**

25024649

**Following code for input:**

****

**Output obtained in execution:**



**21**

25024649

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

**23**

25024649

Direct Function Call:

The ‘main()’ function is called to execute program.

Conclusion of the Program:

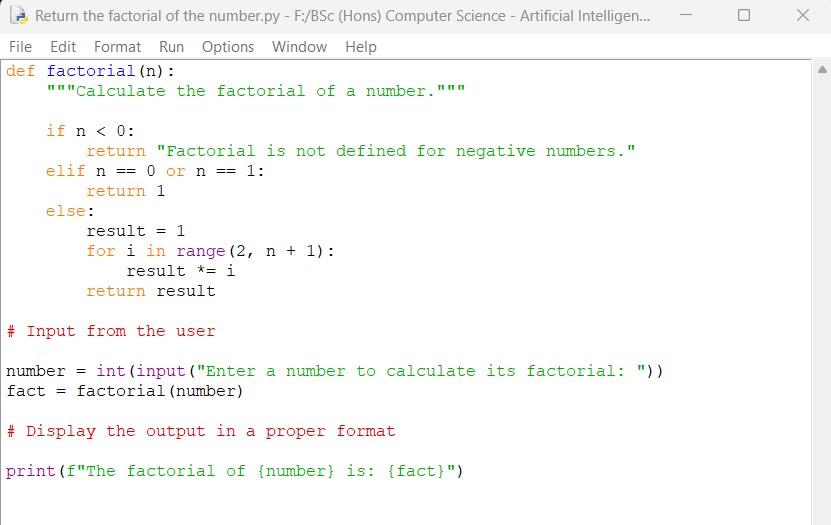
The given program is an example of how to gather user input and perform floor division in python. First, we must set up the variables, utilize the floor division operator (//), and display the results with the print() function.

1. 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.

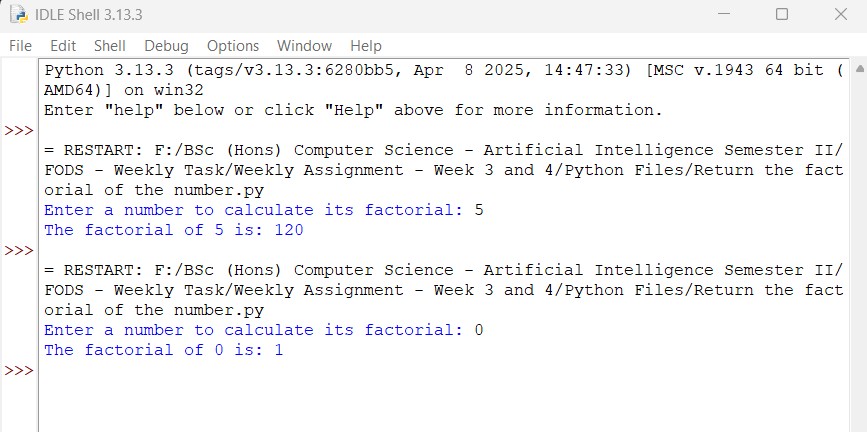
**Following code for input:**



**23**

25024649

**Output obtained in execution:**

****

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 i0s not defined for negative numbers. This prevents further calculations that would be invalid.

**24**

25024649

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 literates 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 interger 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.

**25**

25024649

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.

1. 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.

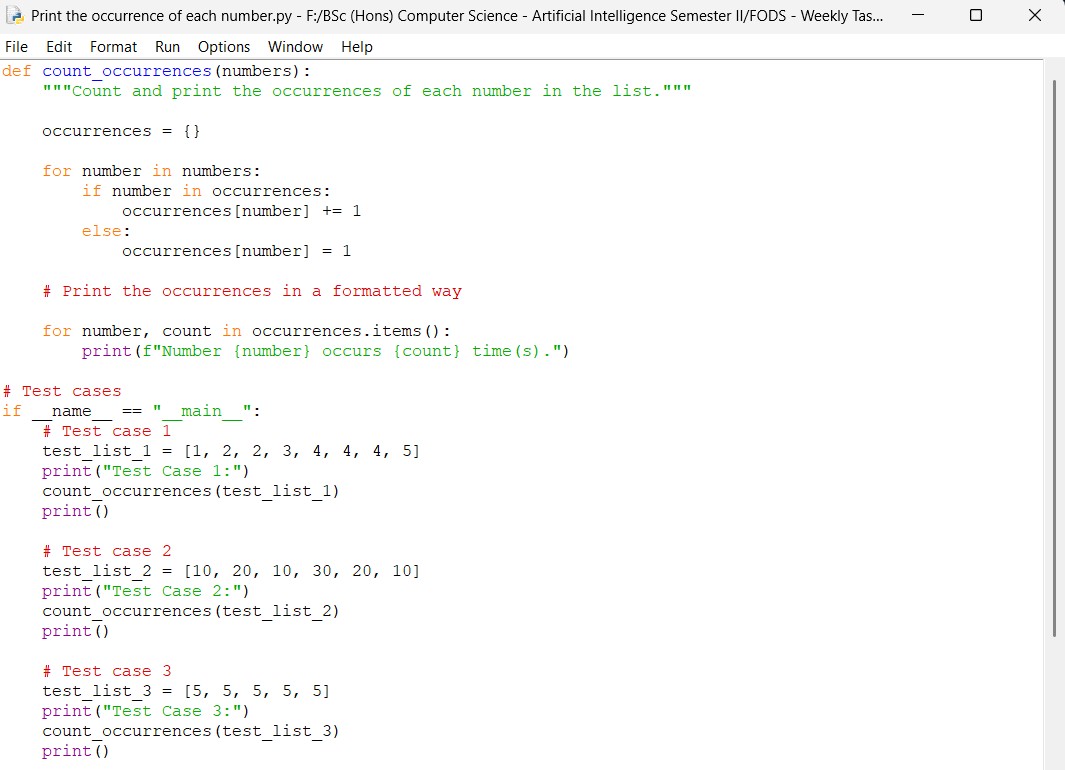
**Answer:**

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.

**27**

25024649

**Following code for input:**



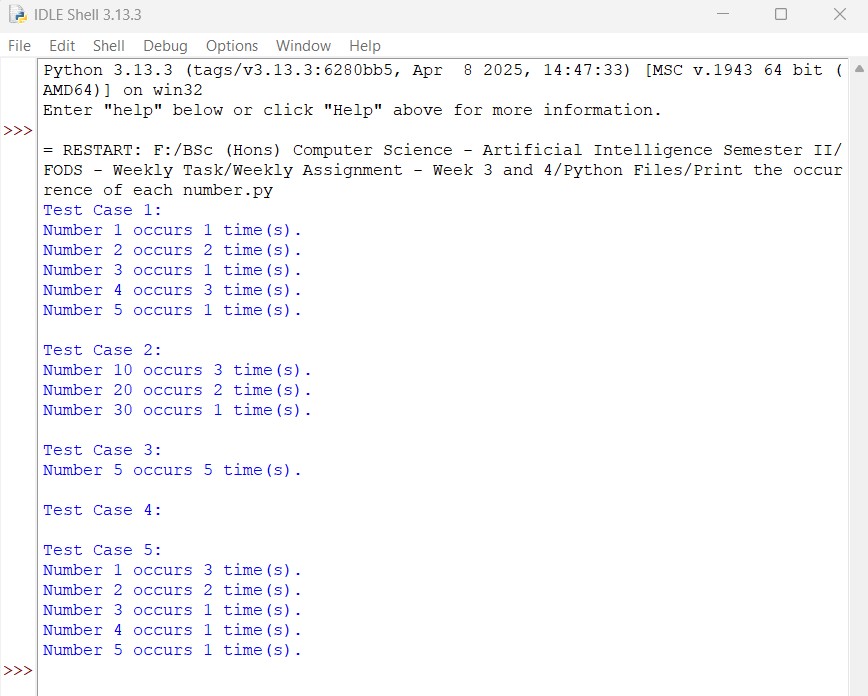
Continue:



**28**

25024649

**Output obtained in execution:**

****

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.

**28**

25024649

Counting Occurrences:

An empty dictionary ‘occurrences’ is initialized to store the count of each number. The function iterates thought 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.

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

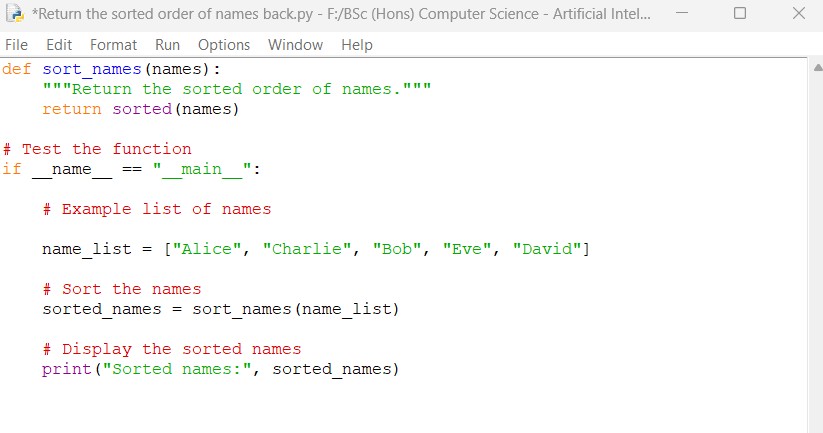
**Answer:**

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

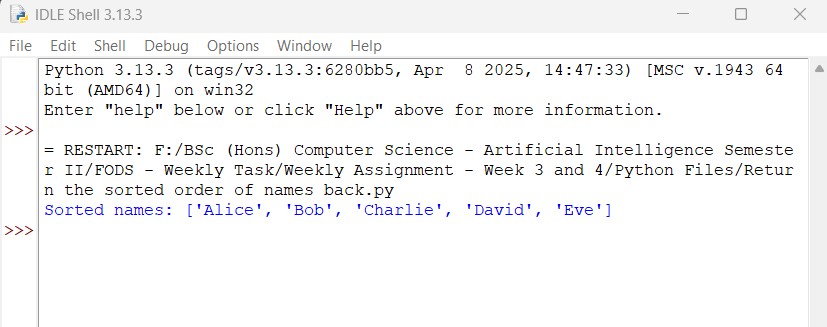
**29**

25024649

**Following code for input:**



**Output obtained in execution:**



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

**30**

25024649

**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.

**31**

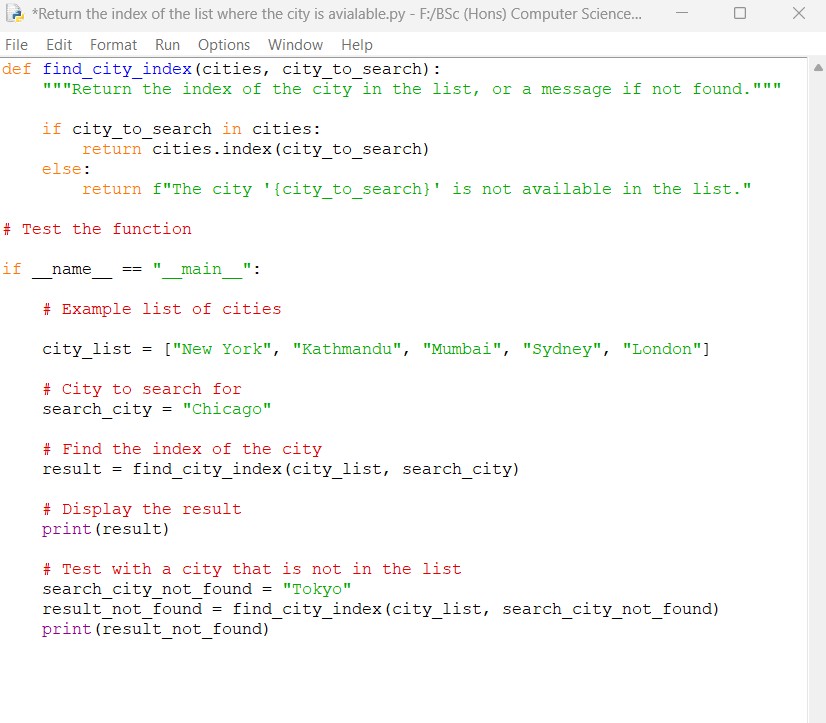
25024649

1. 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:**

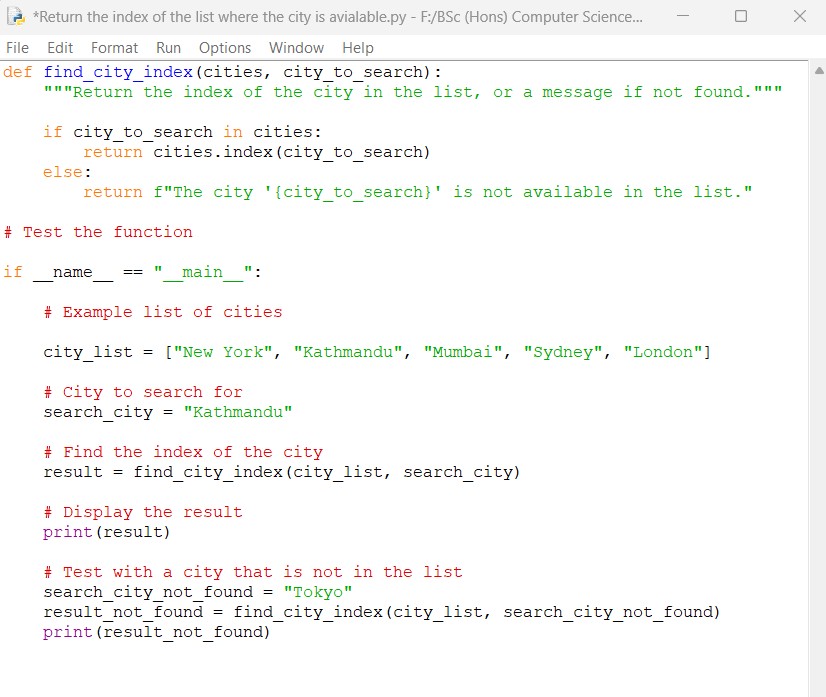
The given python program defines a function to accept a list of cities and a city name to search for. The function returns the index of the city in the list if it exits, or a message indicating that the city is found.

**Following code for input:**

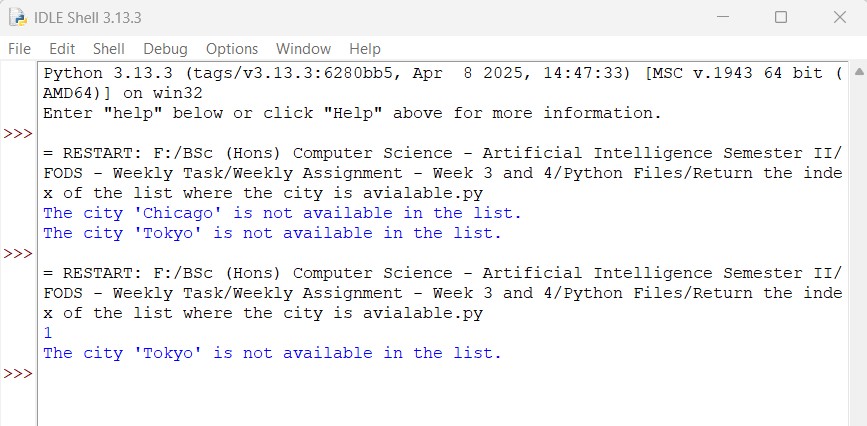
**  
32**

25024649

Continue:

****

**Output obtained in execution:**

****

**33**

25024649

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.

**34**

25024649

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 if cities and the city to search, enhancing its usability.

1. 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.]

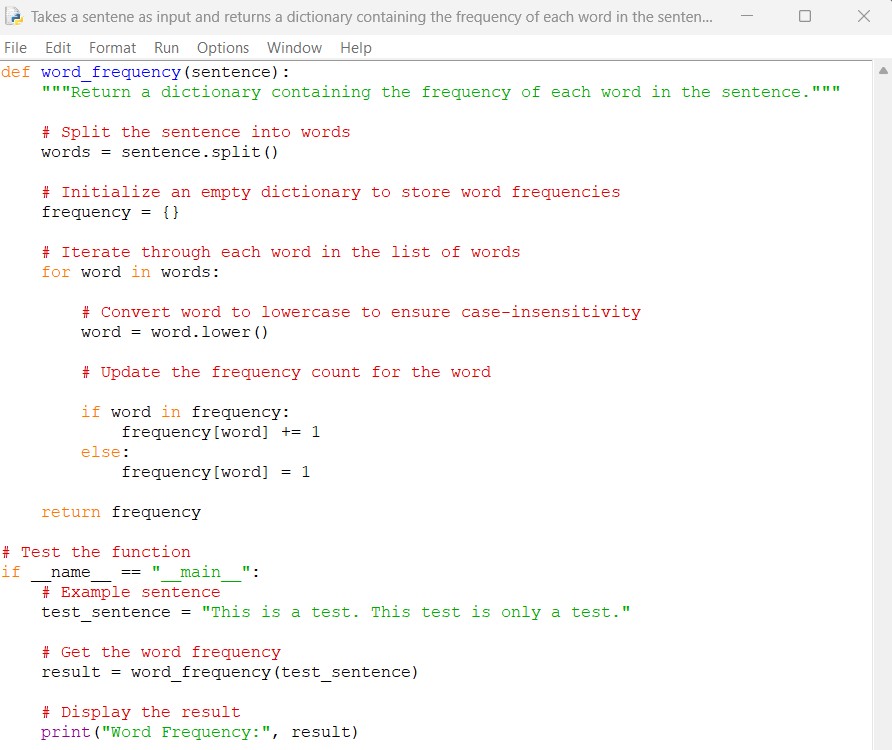
Answer:

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 returns a dictionary containing the word counts.

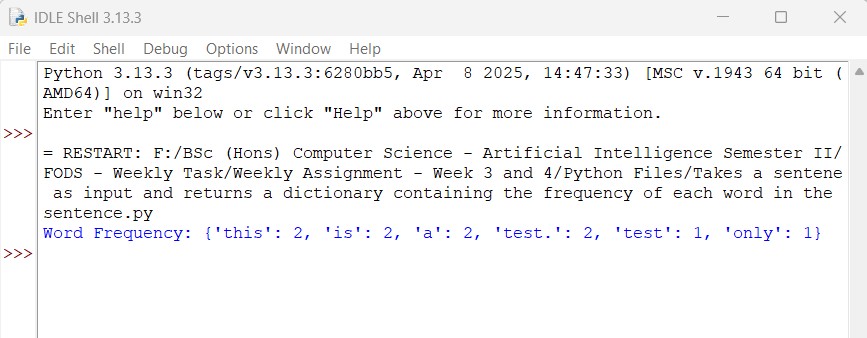
**35**

25024649

**Following code for input:**



**Output obtained in execution:**



**36**

25024649

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\_freqeuncy(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.

**37**

25024649

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.

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

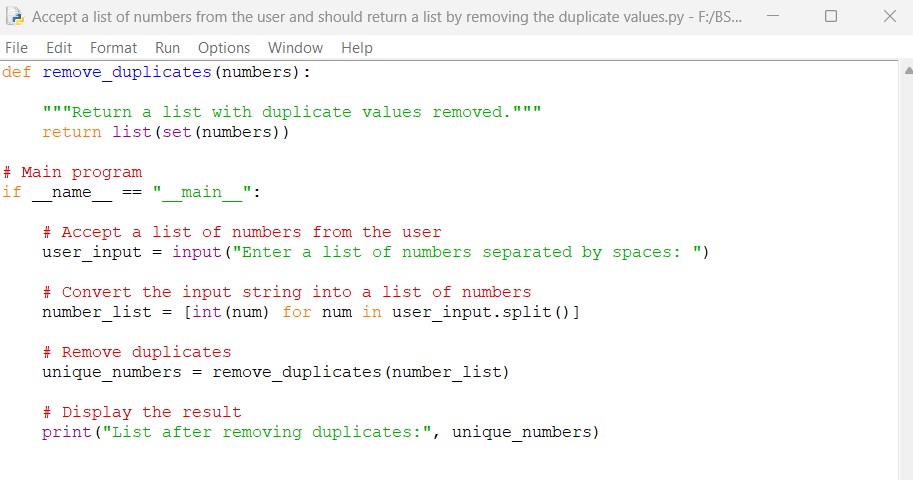
**Answer:**

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.

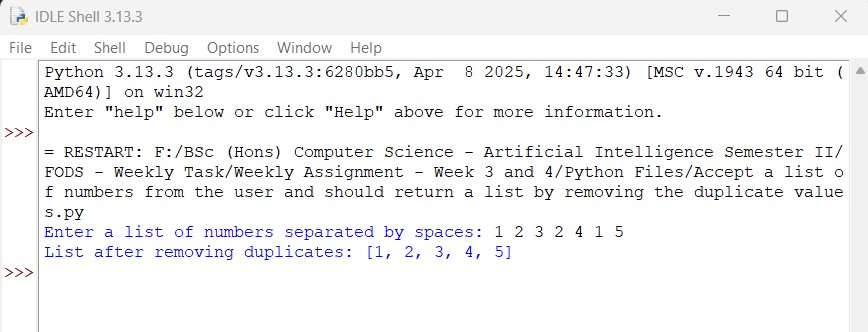
**38**

25024649

**Following code for input:**



**Output obtained in execution:**



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

**39**

25024649

**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 in 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’.

**40**

25024649

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.

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

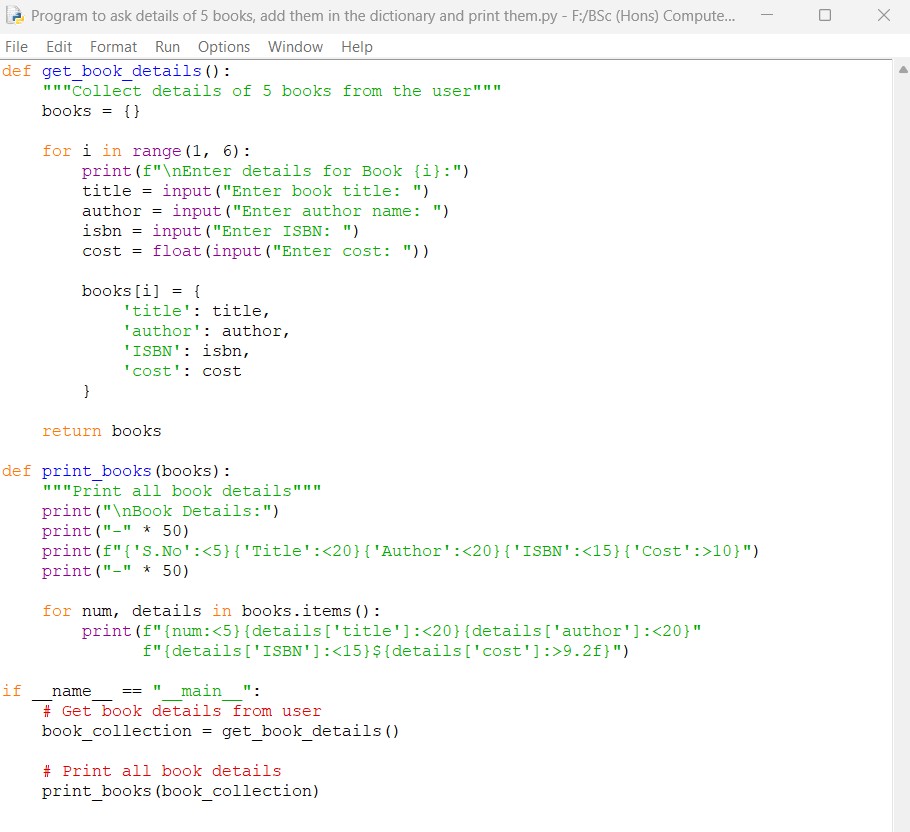
Answer:

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.

**41**

25024649

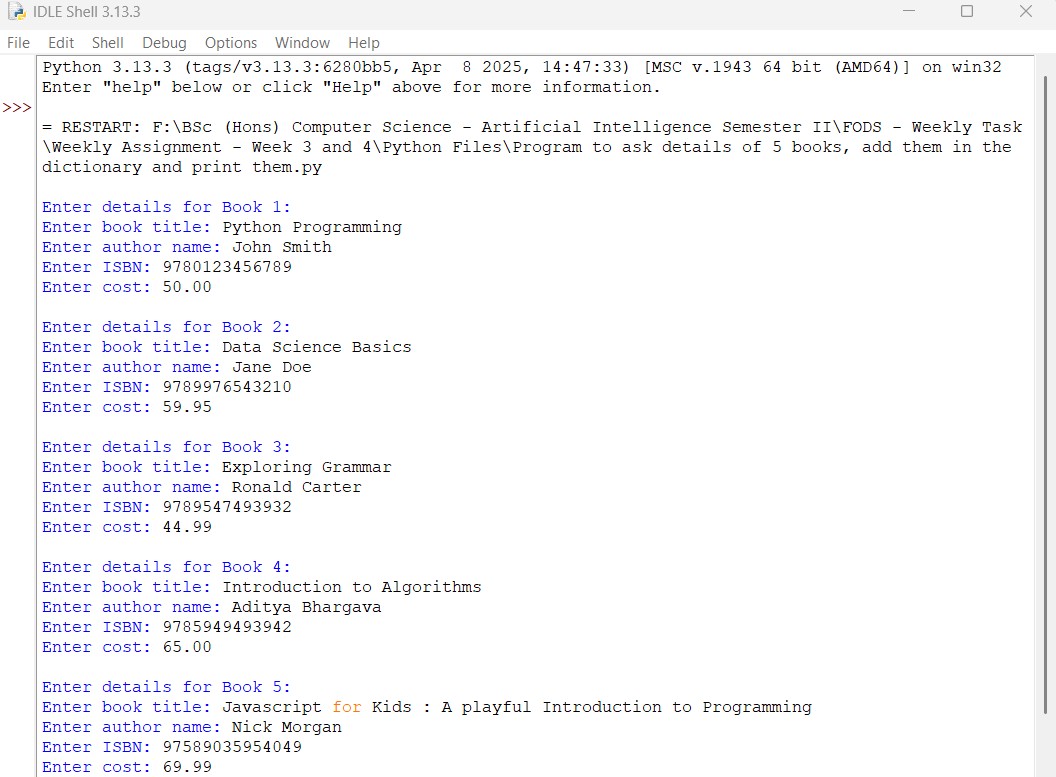
**Following code for input:**

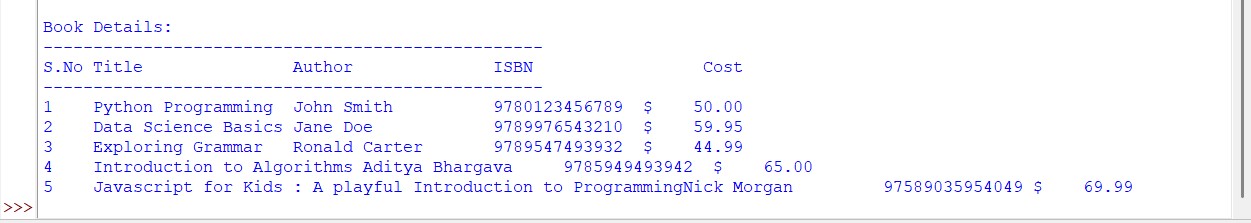


**42**

25024649

**Output obtained in execution:**

****

****

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

**43**

25024649

**Explanation of code:**

Function Definitions:

The program defines two functions: ‘get\_book\_detials()’ 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 tile, 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.

**44**

25024649

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 ley (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\_detials()’ 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 ad functions, loops, dictionaries, and formatted output.

**45**

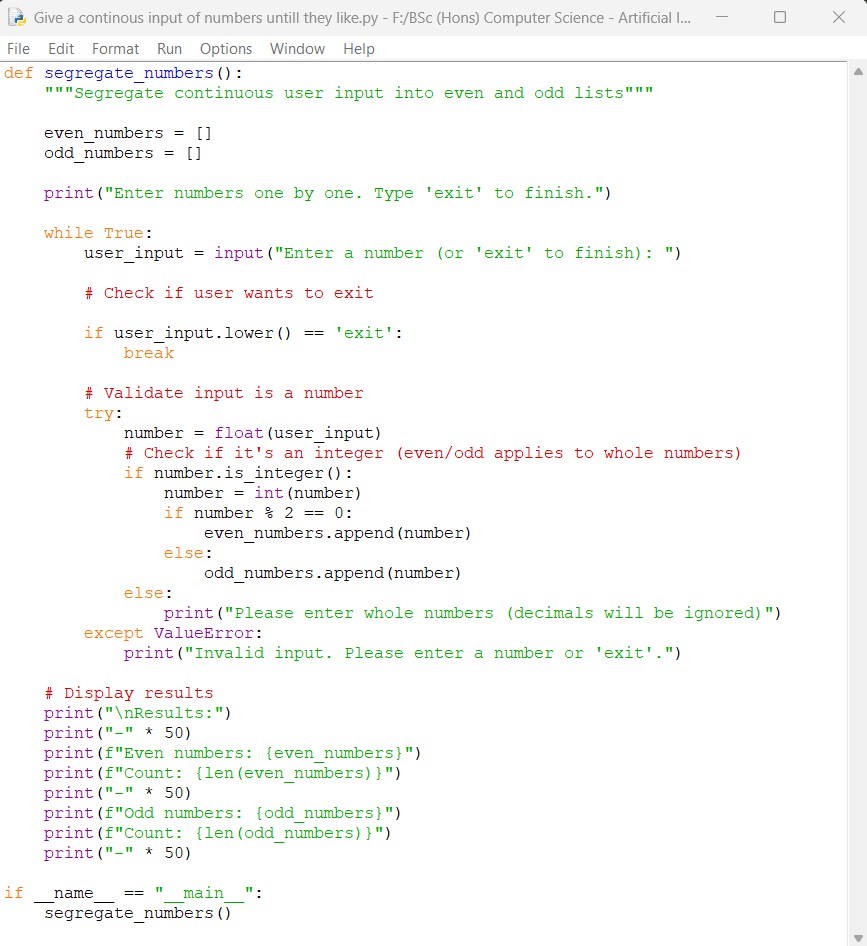
25024649

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.

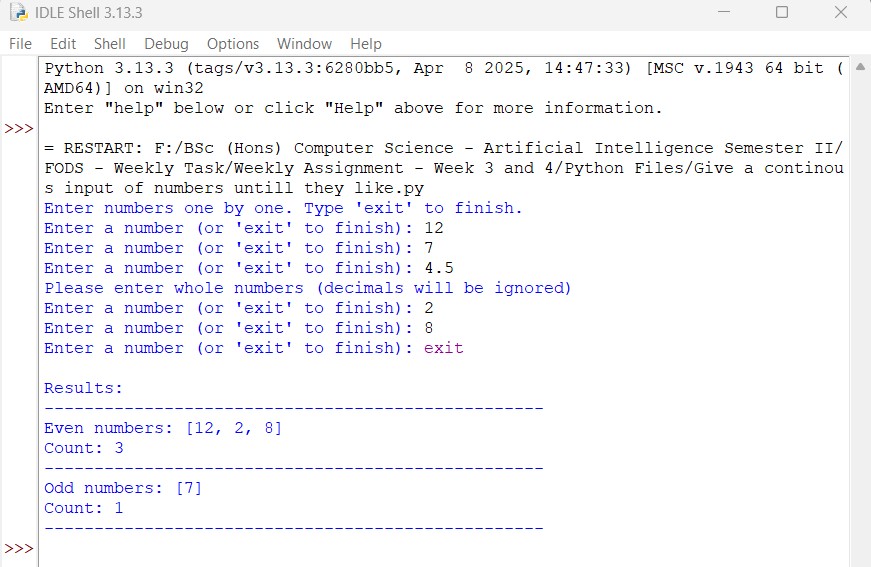
**Following code for input:**



**46**

25024649

**Output obtained in execution:**

****

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.

**47**

25024649

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 allows 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\_interger()’. 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 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.

**48**

25024649

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.

1. The program should have a list of card values like 2, 3, 4,…, Jack, Queen, King, Ace
2. The program should have a list of card suits like heart, diamond, club, spades.
3. The program should randomly pick up a number and a suit and keep as an answer in a separate list.
4. The program should ask the player to guess the card value and the suit.
5. 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.

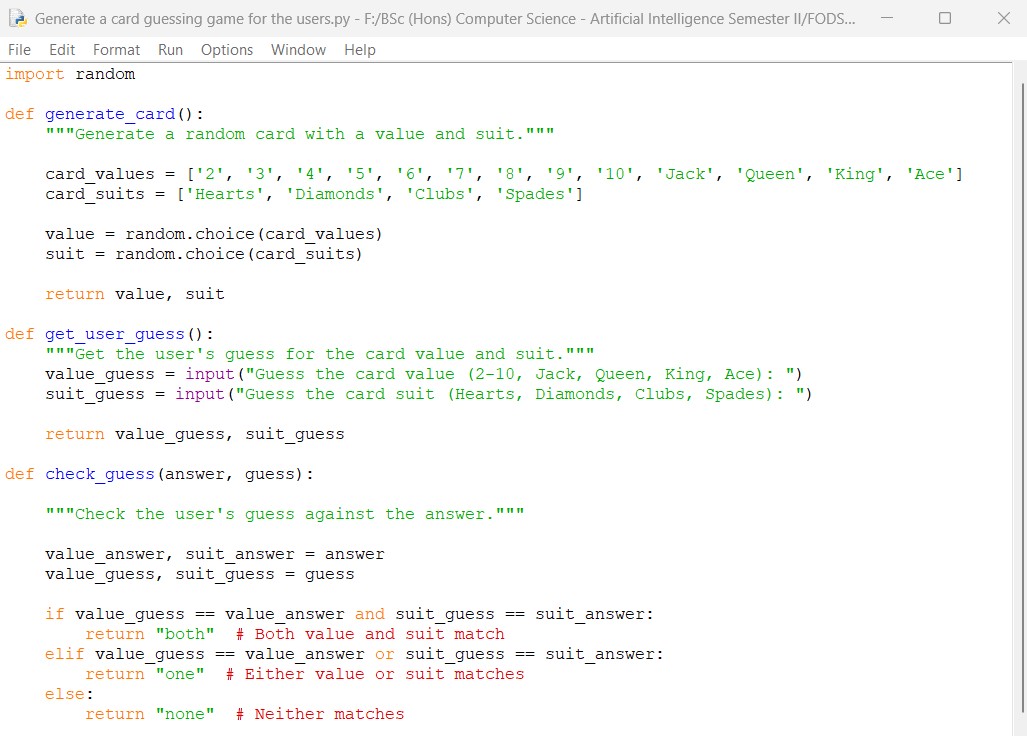
**49**

25024649

**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.

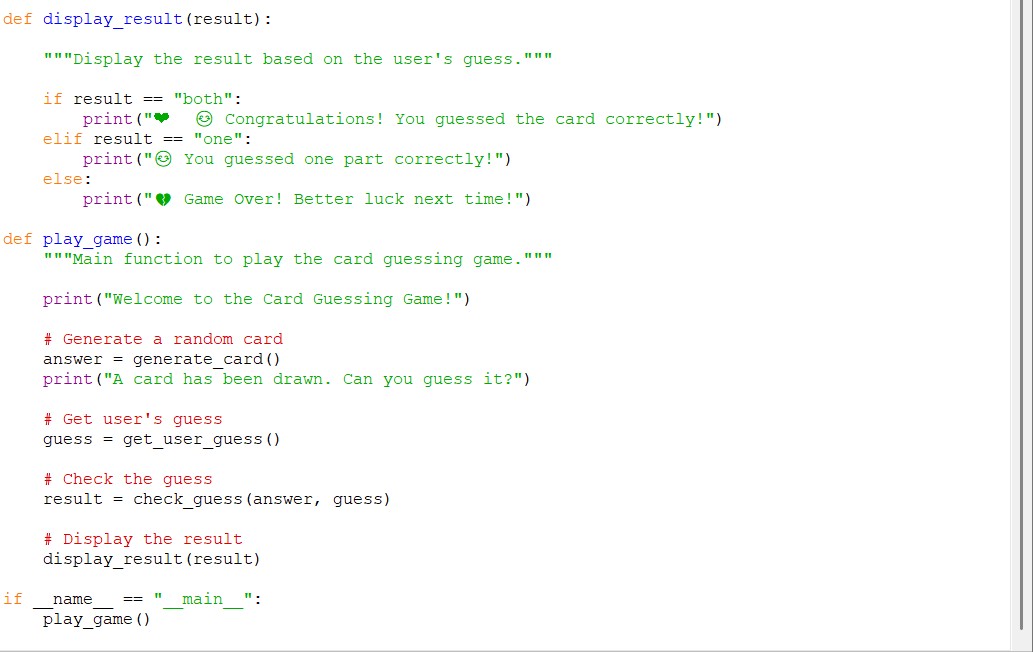
**Following code for input:**



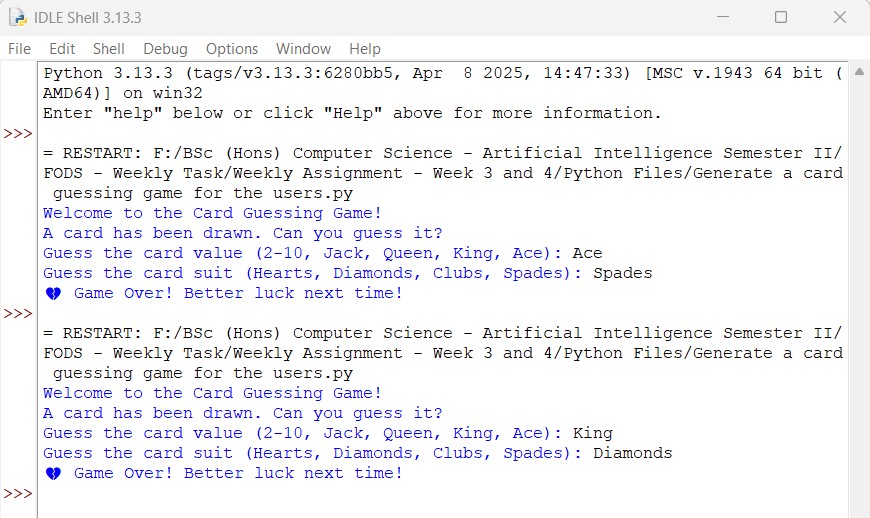
**50**

25024649

Continue:



**Explanation of code:**



**51**

25024649

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.

**52**

25024649

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.

**53**