**Name of Student: Raheel Kotwal**

**Roll Number: 45**

**Experiment Number: 3.1**

**Title: Appending List**

**Theory:**  A list is created to store BRICS nations: Brazil, Russia, India, China, and South Africa.

**Code:**

original\_list = [1, 2, 3]

print(f'Original list without any append: {original\_list}')

list\_by\_plus\_operator = original\_list + [4, 5, 6]

print(f'List after first append: {list\_by\_plus\_operator}')

appendss = original\_list.copy()

appendss.append(4)

appendss.append(5)

appendss.append(6)

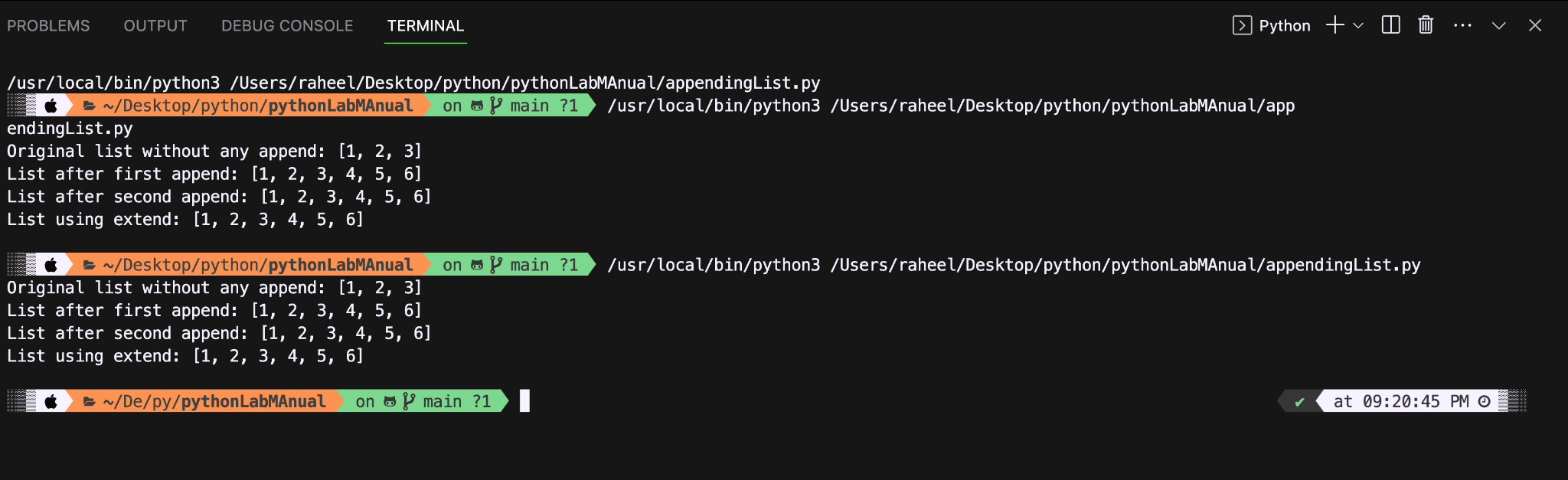
print(f'List after second append: {appendss}')

extendedd = original\_list.copy()

extendedd.extend([4, 5, 6])

print(f'List using extend: {extendedd}')

**Output:(screenshot):**



**Conclusion:**

**Three lists are created which successfully shows the different ways to add values to a list.**

**Name of Student: Raheel Kotwal**

**Roll Number: 45**

**Experiment Number: 3.2**

**Title: Addition of Matrices**

**Theory:** The following code takes the values of number of elements in a two dimensional matrix and adds them up. The code checks if both matrices are compactible or not, in order to add them.

**Code:**

mat1Rows = int(input("Enter the number of rows for the first matrix: "))

mat1Cols = int(input("Enter the number of columns for the first matrix: "))

mat2Rows = int(input("Enter the number of rows for the second matrix: "))

mat2Cols = int(input("Enter the number of columns for the second matrix: "))

if mat1Rows != mat2Rows or mat1Cols != mat2Cols:

print("Matrix addition is not possible. Dimensions are not compatible.")

exit()

# Input for the first matrix

matrix1 = []

print("Enter the elements of the first matrix:")

for i in range(mat1Rows):

row = []

for j in range(mat1Cols):

element = int(input(f"Enter {i + 1}th row & {j + 1}th column element of the first matrix: "))

row.append(element)

matrix1.append(row)

# Input for the second matrix

matrix2 = []

print("\nEnter the elements of the second matrix:")

for i in range(mat2Rows):

row = []

for j in range(mat2Cols):

element = int(input(f"Enter {i + 1}th row & {j + 1}th column element of the second matrix: "))

row.append(element)

matrix2.append(row)

# Matrix addition

result = [[0 for \_ in range(mat1Cols)] for \_ in range(mat1Rows)]

for i in range(mat1Rows):

for j in range(mat1Cols):

result[i][j] = matrix1[i][j] + matrix2[i][j]

print("\nResult of matrix addition:")

for i in range(mat1Rows):

print("|", end=" ")

for j in range(mat1Cols):

if j < mat1Cols - 1:

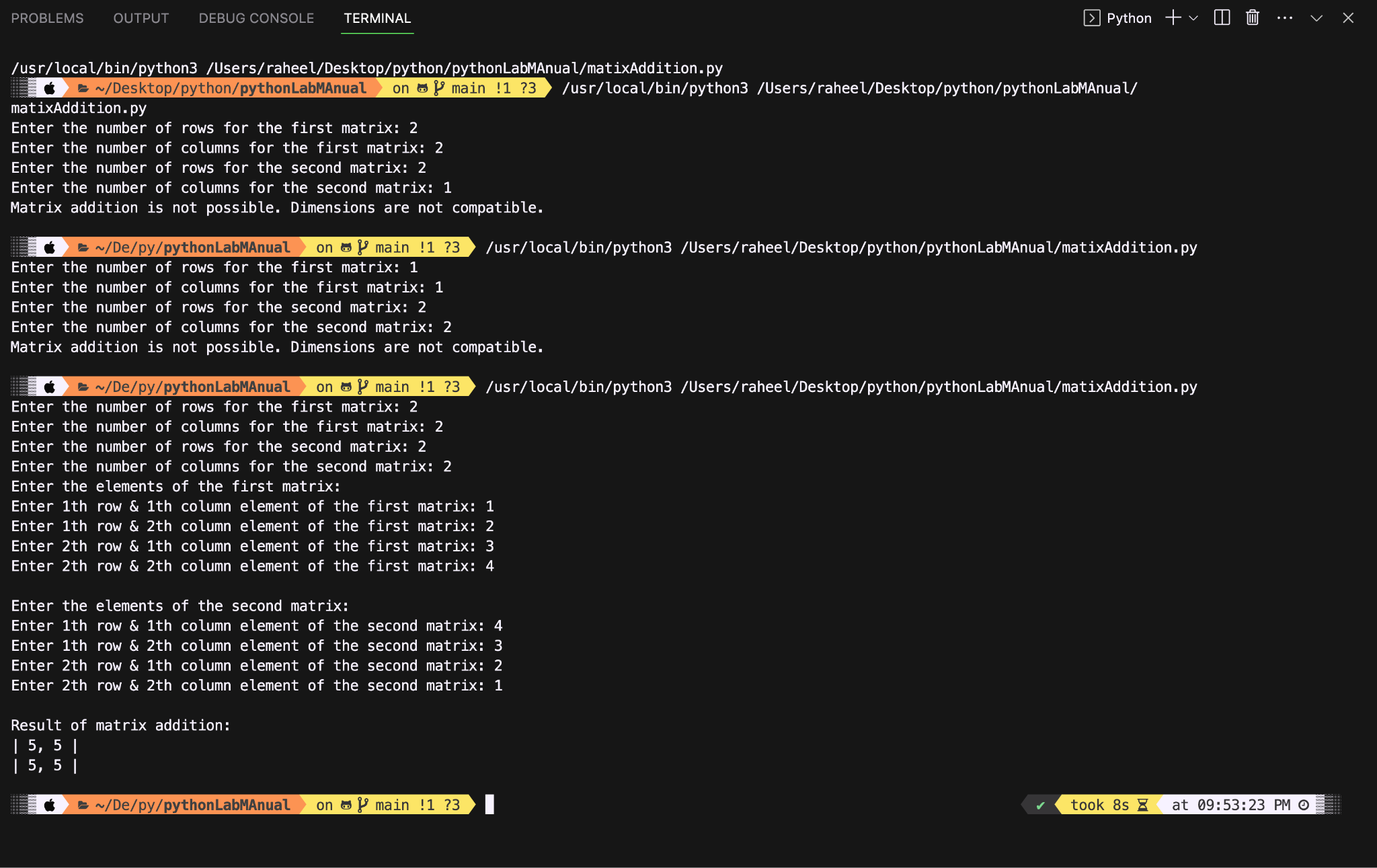
print(result[i][j], end=", ")

else:

print(result[i][j], end=" | ")

print()

**Output:(screenshot):**



**Conclusion:**

**The code properly checks the dimensions of both matrices and further more it also calculates the sum of both matrices precisely.**

**Name of Student: Raheel Kotwal**

**Roll Number: 45**

**Experiment Number: 3.2**

**Title: Addition of Matrices**

**Theory:** The following code takes the values of number of elements in a two dimensional matrix and adds them up. The code checks if both matrices are compactible or not, in order to add them.

**Code:**

mat1Rows = int(input("Enter the number of rows for the first matrix: "))

mat1Cols = int(input("Enter the number of columns for the first matrix: "))

mat2Rows = int(input("Enter the number of rows for the second matrix: "))

mat2Cols = int(input("Enter the number of columns for the second matrix: "))

if mat1Rows != mat2Rows or mat1Cols != mat2Cols:

print("Matrix addition is not possible. Dimensions are not compatible.")

exit()

# Input for the first matrix

matrix1 = []

print("Enter the elements of the first matrix:")

for i in range(mat1Rows):

row = []

for j in range(mat1Cols):

element = int(input(f"Enter {i + 1}th row & {j + 1}th column element of the first matrix: "))

row.append(element)

matrix1.append(row)

# Input for the second matrix

matrix2 = []

print("\nEnter the elements of the second matrix:")

for i in range(mat2Rows):

row = []

for j in range(mat2Cols):

element = int(input(f"Enter {i + 1}th row & {j + 1}th column element of the second matrix: "))

row.append(element)

matrix2.append(row)

# Matrix addition

result = [[0 for \_ in range(mat1Cols)] for \_ in range(mat1Rows)]

for i in range(mat1Rows):

for j in range(mat1Cols):

result[i][j] = matrix1[i][j] + matrix2[i][j]

print("\nResult of matrix addition:")

for i in range(mat1Rows):

print("|", end=" ")

for j in range(mat1Cols):

if j < mat1Cols - 1:

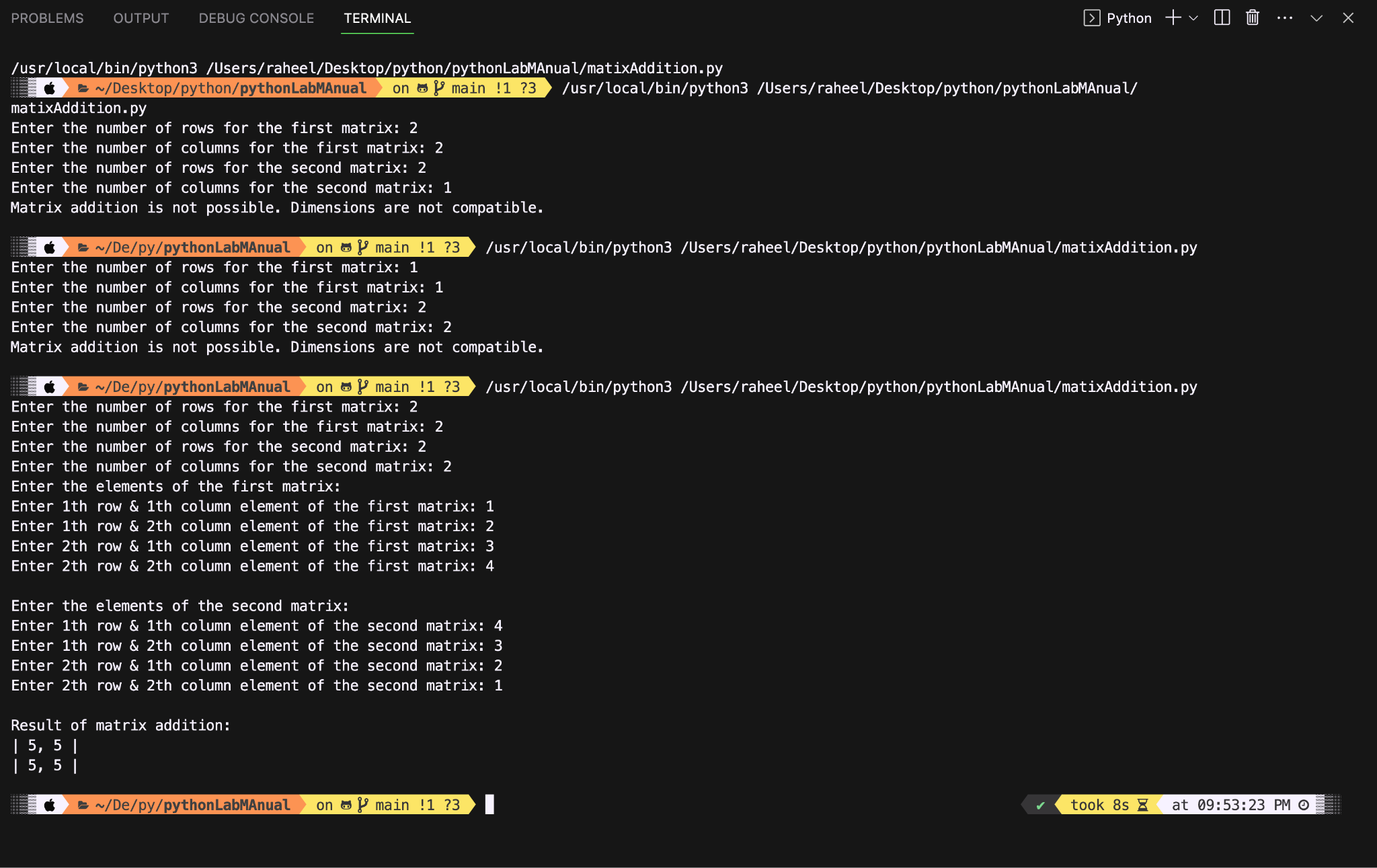
print(result[i][j], end=", ")

else:

print(result[i][j], end=" | ")

print()

**Output:(screenshot):**



**Conclusion:**

**The code properly checks the dimensions of both matrices and further more it also calculates the sum of both matrices precisely.**

**Name of Student: Raheel Kotwal**

**Roll Number: 45**

**Experiment Number: 3.3**

**Title: Distinction of list**

**Theory:** The following code takes a list from user and then creates a new list where no elements are same.

**Code:**

original\_list = []

num\_elements = int(input("Enter the number of elements in the list: "))

for i in range(num\_elements):

element = int(input(f"Enter element {i + 1}: "))

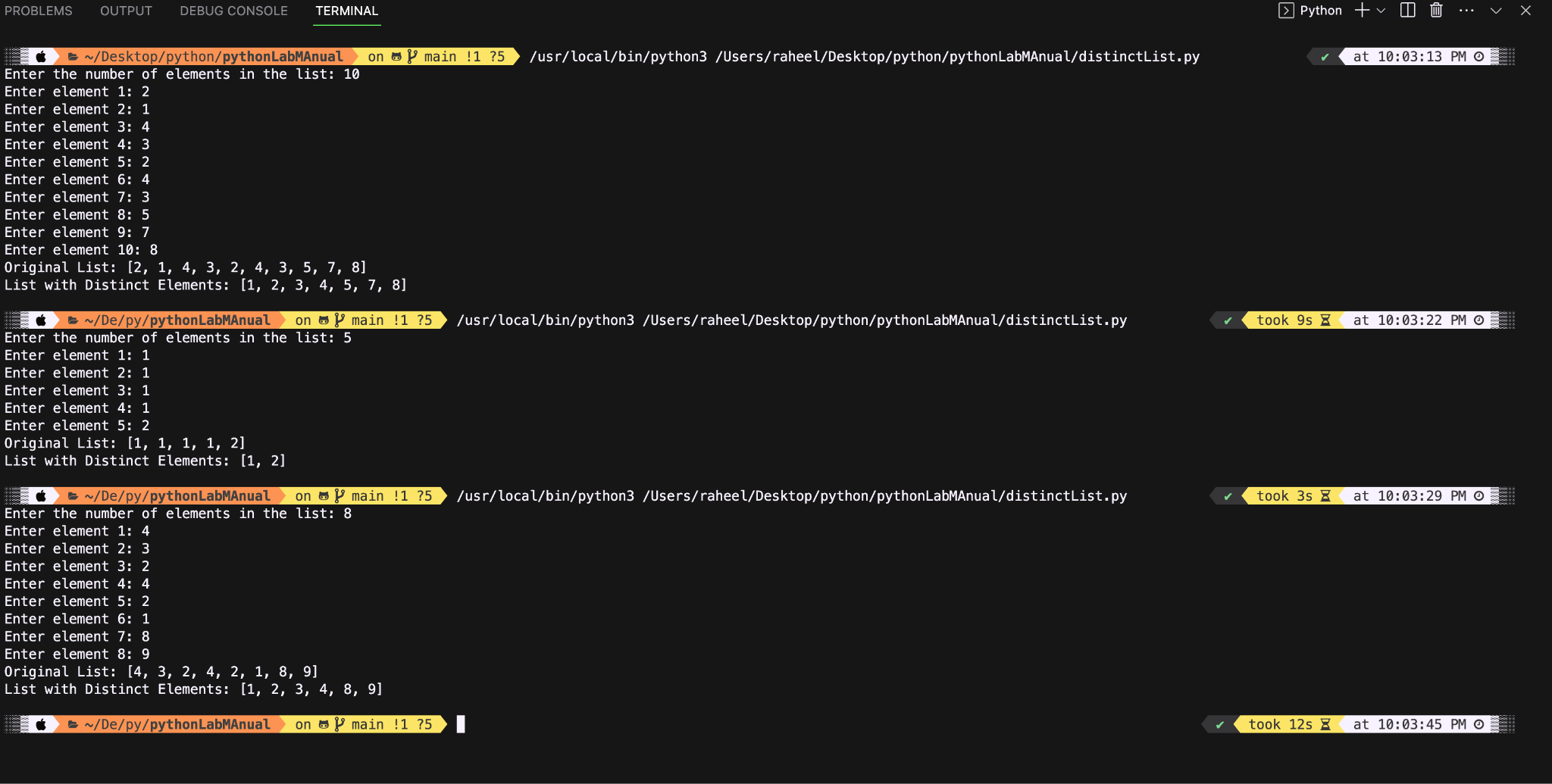
original\_list.append(element)

unique\_list = list(set(original\_list)) #set a built in data type that has unordered collection of unique elements.

print("Original List:", original\_list)

print("List with Distinct Elements:", unique\_list)

**Output:(screenshot):**



**Conclusion:**

**The code correctly works and uses the set() method for distinction of elements.**

**The set() method is used to return a unique list of elements from a previous list (as a parameter) in an unordered format.**

**Name of Student: Raheel Kotwal**

**Roll Number: 45**

**Experiment Number: 3.4**

**Title: Perfect Number**

**Theory:** A perfect number is one whose sum of proper divisors (excluding itself) equals the number.

**Code:**

number = int(input("Enter a number: "))

if number <= 0:

print(f"{number} is not a perfect number.")

else:

divisor\_sum = sum([i for i in range(1, number)if number % i == 0])

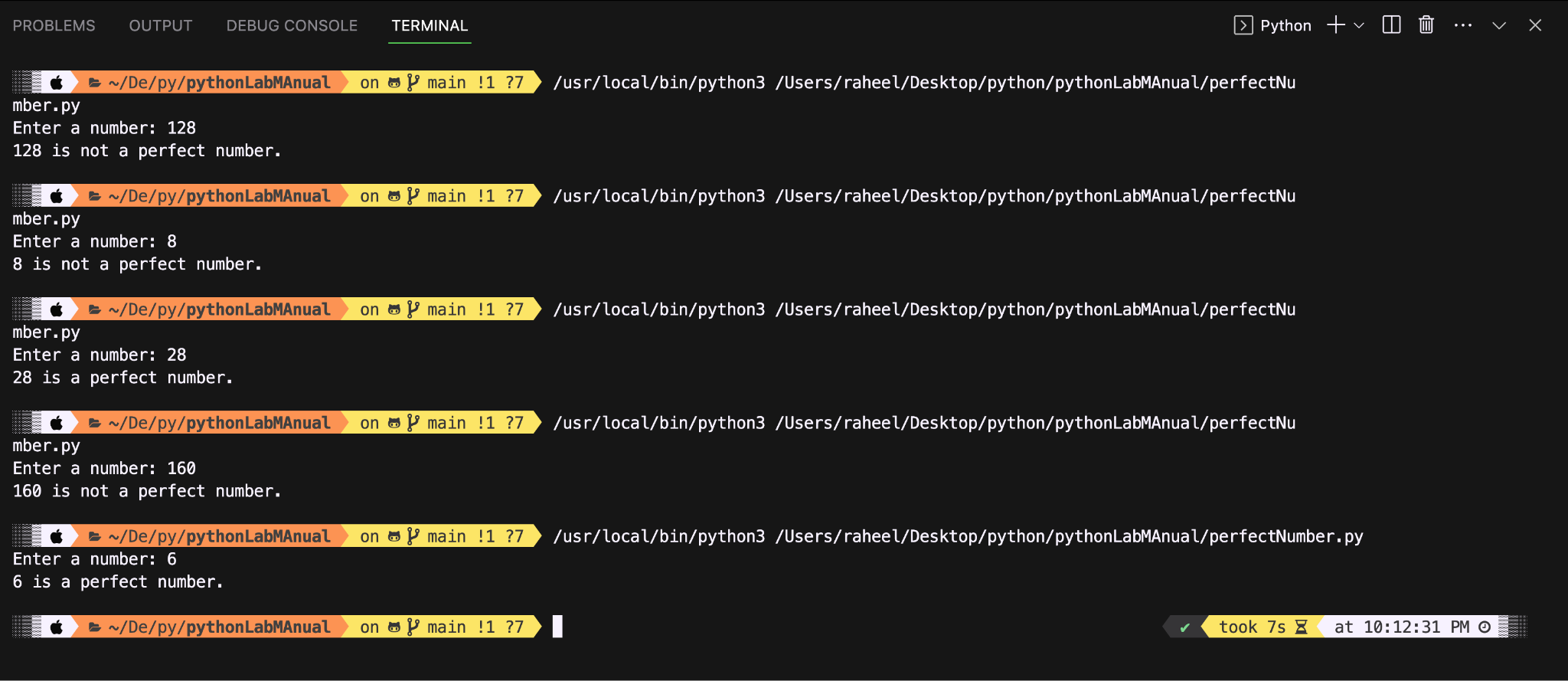
if divisor\_sum == number:

print(f"{number} is a perfect number.")

else:

print(f"{number} is not a perfect number.")

**Output:(screenshot):**



**Conclusion:**

**The code properly iterates through divisors, calculates their sum, and compares it to the original number.**

**Name of Student: Raheel Kotwal**

**Roll Number: 45**

**Experiment Number: 3.5**

**Title: Character count**

**Theory:** The following code takes a string from user and uses two predefined functions: isupper() and islower() to find if a character in that string is capitalised or not. Thus create count accordingly for both characters separately.

**Code:**

def count\_upper\_lower(string):

upper\_count = 0

lower\_count = 0

for char in string:

if char.isupper():

upper\_count += 1

elif char.islower():

lower\_count += 1

return upper\_count, lower\_count

stringToTest = str(input("Enter a long string:\n"))

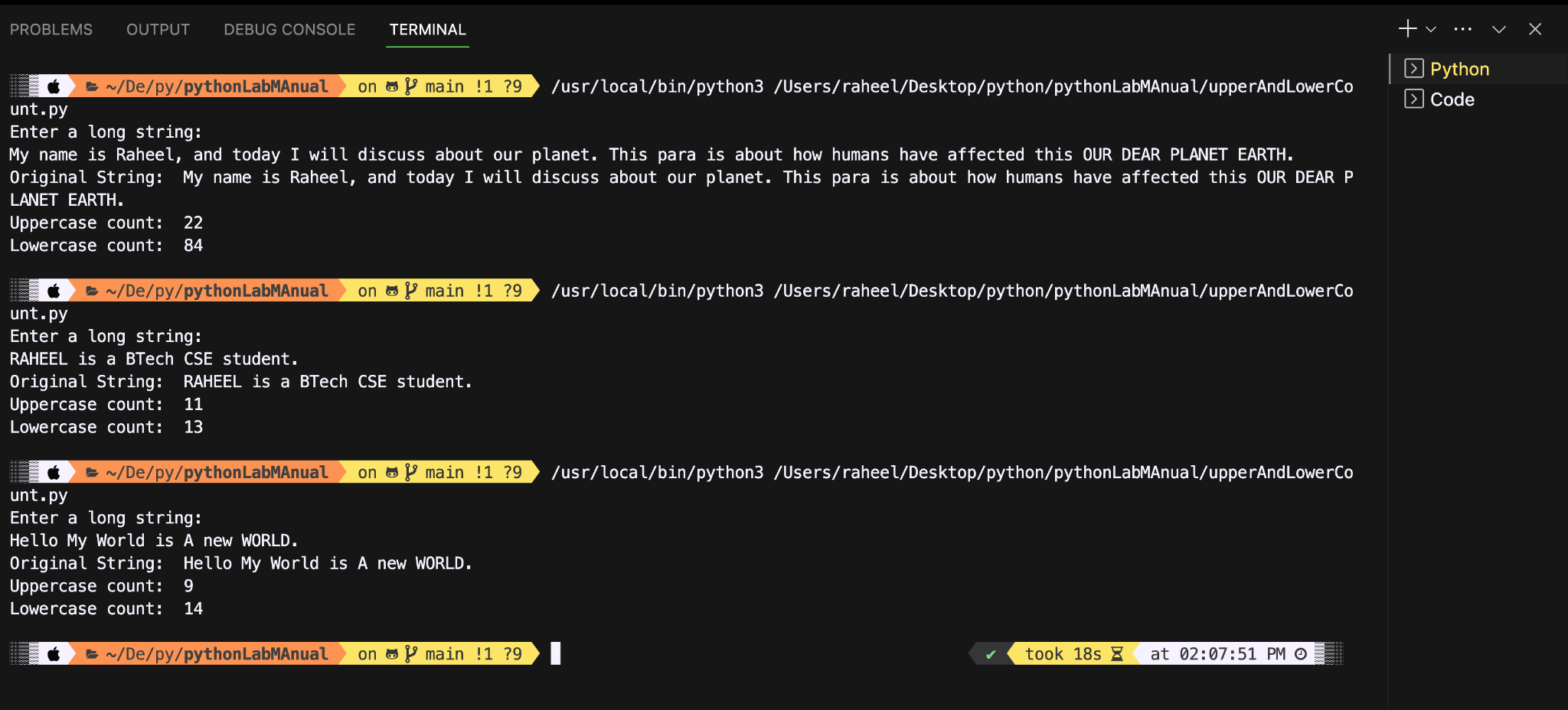
upper, lower = count\_upper\_lower(stringToTest)

print("Original String: ", stringToTest)

print("Uppercase count: ", upper)

print("Lowercase count: ", lower)

**Output:(screenshot):**



**Conclusion:**

**The code uses simple character case checks to achieve the desired outcome correctly.**