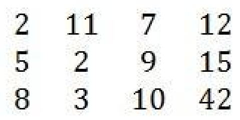
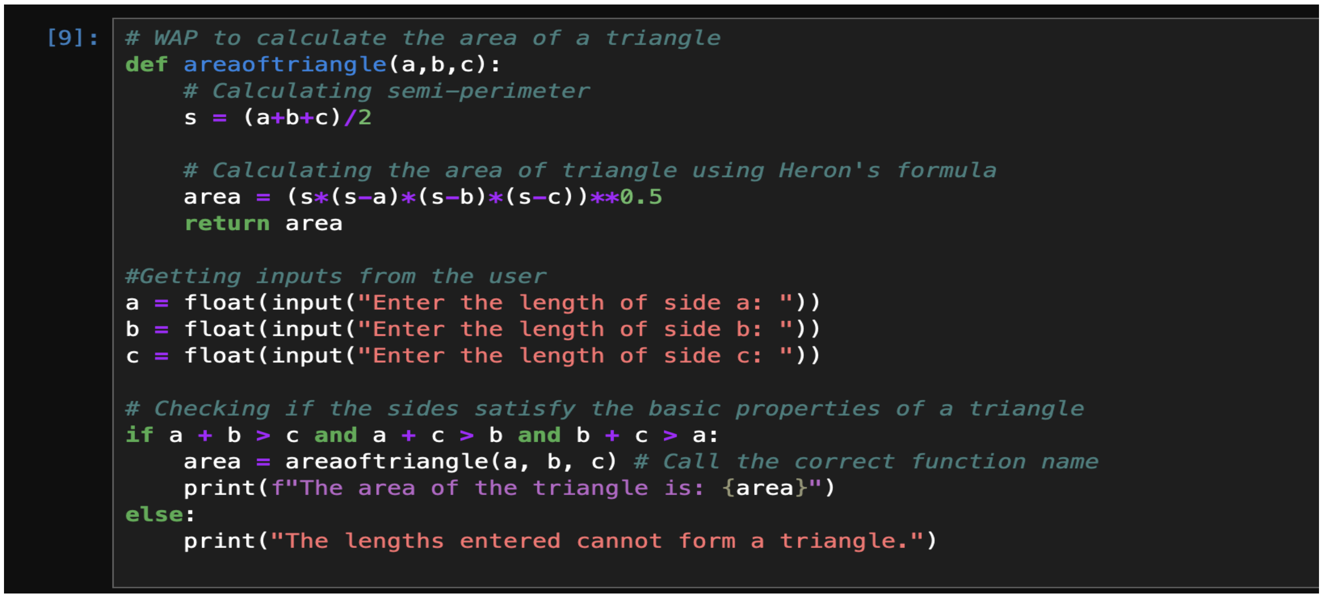
INDEX

6.	a.) Write a program that accepts a list from the user. Your program should reverse the content of the list and display it. Do not use the reverse () method. b) Find and display the largest number of a list without using built-in function max (). Your program should ask the user to input values in the list from the keyboard.	**16**	
7.	Find the sum of each row of matrix of size m x n. For example, for the following matrix output will be like this:  2 11 7 12 5 2 9 15 8 3 10 42 Sum of row 1 = 32 Sum of row 2 = 31 Sum of row 3 = 63	**18**	
8.	a) Write a program that reads a string from keyboard and display: * The number of uppercase letters in the string. * The number of lowercase letters in the string. * The number of digits in the string. * The number of whitespace characters in the string. b) Python Program to Find Common Characters in Two Strings. c) Python Program to Count the Number of Vowels in a String.	**20**	
9.	a) Write a Python program to check if a specified element presents in a tuple of tuples. Original list: ((‘Red’ ,’White’ , ‘Blue’),(‘Green’, ’Pink’ , ‘Purple’), (‘Orange’, ‘Yellow’, ‘Lime’)) Check if White present in said tuple of tuples! True Check if Olive present in said tuple of tuples! False b) Write a Python program to remove an empty tuple(s) from a list of tuples. Sample data: [(), (), ('',), ('a', 'b'), ('a', 'b', 'c'), ('d')] Expected output: [('',), ('a', 'b'), ('a', 'b', 'c'), 'd']	**24**	
10.	a) Write a Program in Python to Find the Differences Between Two Lists Using Sets.	**26**	

11.	a) Write a Python program Remove duplicate values across Dictionary Values. Input : test_dict = {‘Manjeet’: [1], ‘Akash’: [1, 8, 9]} Output : {‘Manjeet’: [], ‘Akash’: [8, 9]} Input : test_dict = {‘Manjeet’: [1, 1, 1], ‘Akash’: [1, 1, 1]} Output : {‘Manjeet’: [], ‘Akash’: []} b) Write a Python program to Count the frequencies in a list using dictionary in Python. Input : [1, 1, 1, 5, 5, 3, 1, 3, 3, 1,4, 4, 4, 2, 2, 2, 2] Output : 1 : 5 2 : 4 3 : 3 4 : 3 5 : 2 Explanation : Here 1 occurs 5 times, 2 occurs 4 times and so on...	**27**	
12.	a) Write a Python Program to Capitalize First Letter of Each Word in a File. b.) Write a Python Program to Print the Contents of File in Reverse Order.	**29**	
13.	WAP to catch an exception and handle it using try and except cod e blocks.	**31**	
14.	Write a Python Program to Append, Delete and Display Elements of a List using Classes.	**32**	
15.	Write a Python Program to Find the Area and Perimeter of the Circle using Class.	**34**	
16.	Create an interactive application using Python's Tkinter library for graphics programming.	**35**	

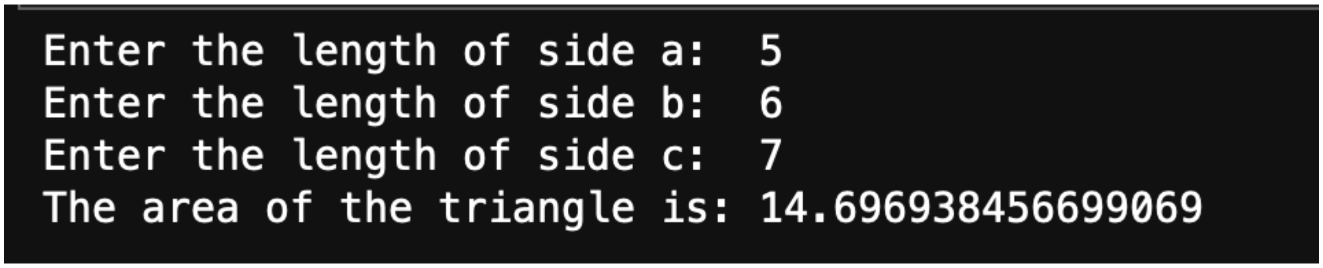
Program 1 (a): Write a Python Program to Calculate the Area of a Triangle

Solution:



```
[9]: # WAP to calculate the area of a triangle
def areaoftriangle(a,b,c):
    # Calculating semi-perimeter
    s = (a+b+c)/2

    # Calculating the area of triangle using Heron's formula
    area = (s*(s-a)*(s-b)*(s-c))**0.5
    return area

#Getting inputs from the user
a = float(input("Enter the length of side a: "))
b = float(input("Enter the length of side b: "))
c = float(input("Enter the length of side c: "))

# Checking if the sides satisfy the basic properties of a triangle
if a + b > c and a + c > b and b + c > a:
    area = areaoftriangle(a, b, c) # Call the correct function name
    print(f"The area of the triangle is: {area}")
else:
    print("The lengths entered cannot form a triangle.")
```

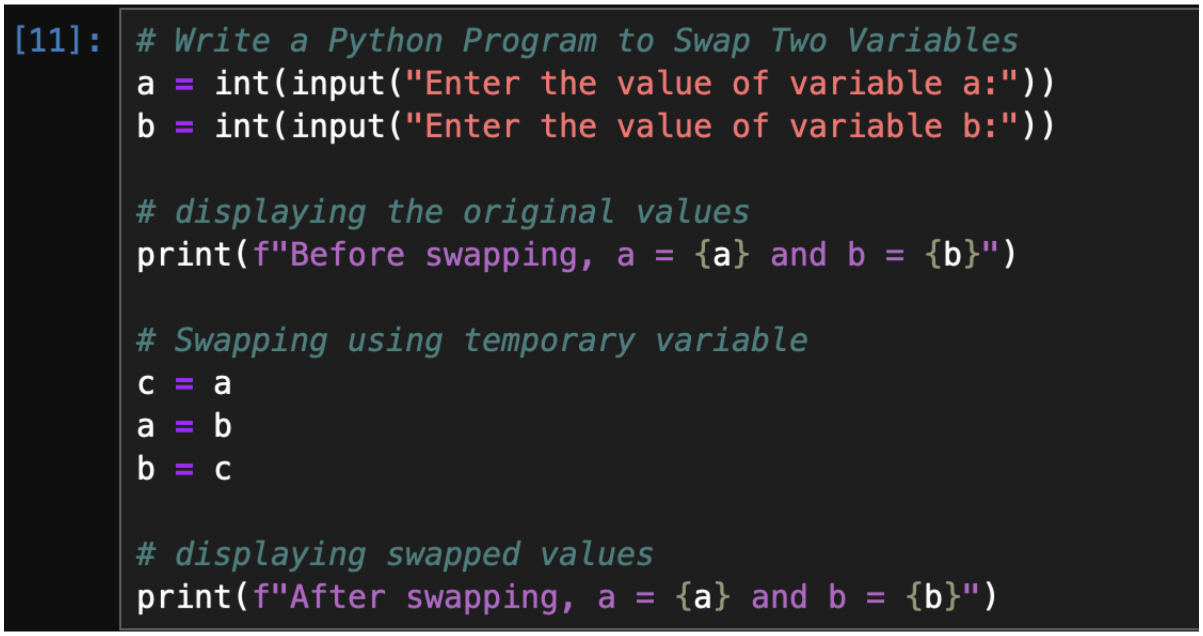
Output:



```
Enter the length of side a:  5
Enter the length of side b:  6
Enter the length of side c:  7
The area of the triangle is: 14.696938456699069
```

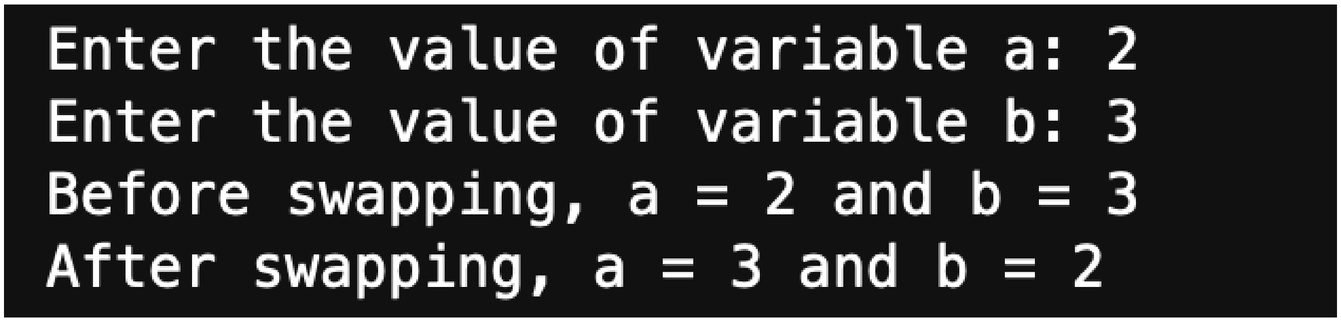
Program 1 (b): Write a Python Program to Swap Two Variables

Solution:



```
[11]: # Write a Python Program to Swap Two Variables
      a = int(input("Enter the value of variable a:"))
      b = int(input("Enter the value of variable b:"))

      # displaying the original values
      print(f"Before swapping, a = {a} and b = {b}")

      # Swapping using temporary variable
      c = a
      a = b
      b = c

      # displaying swapped values
      print(f"After swapping, a = {a} and b = {b}")
```

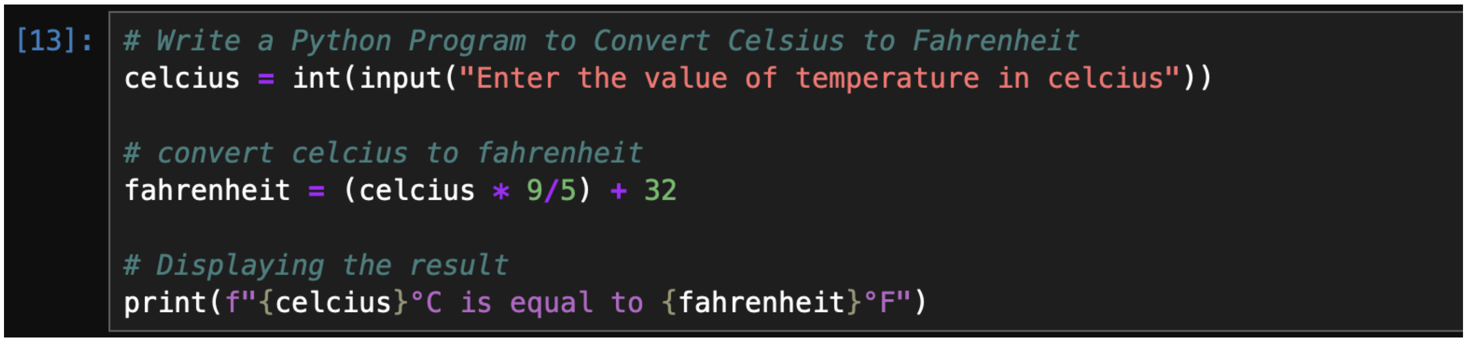
Output:



```
Enter the value of variable a: 2
Enter the value of variable b: 3
Before swapping, a = 2 and b = 3
After swapping, a = 3 and b = 2
```

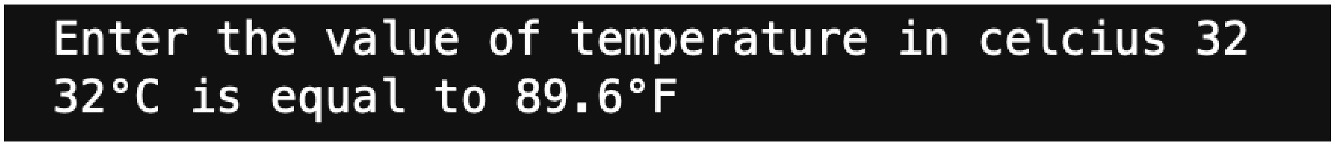
Program 1 (c): Write a Python Program to Convert Celsius to Fahrenheit

Solution:



```
[13]: # Write a Python Program to Convert Celsius to Fahrenheit
      celcius = int(input("Enter the value of temperature in celcius"))

      # convert celcius to fahrenheit
      fahrenheit = (celcius * 9/5) + 32

      # Displaying the result
      print(f"{celcius}°C is equal to {fahrenheit}°F")
```

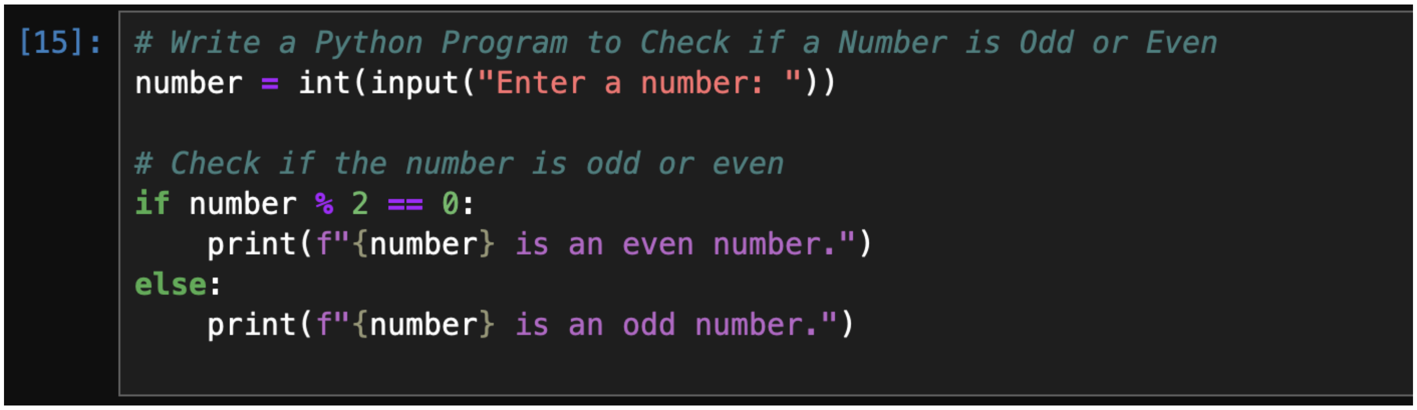
Output:



```
Enter the value of temperature in celcius 32
32°C is equal to 89.6°F
```

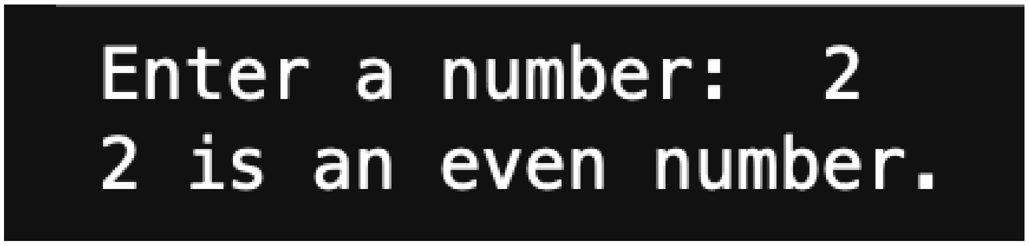
Program 2(a): Write a Python Program to Check if a Number is Odd or Even

Solution:

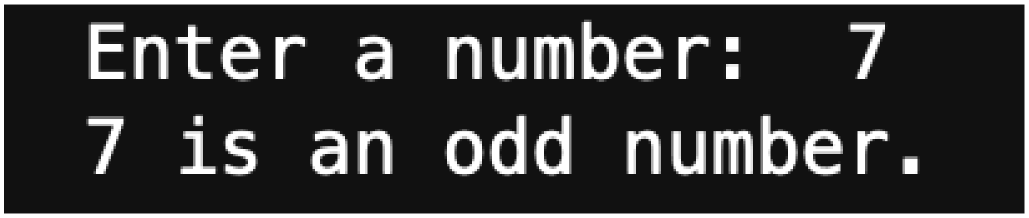


```
[15]: # Write a Python Program to Check if a Number is Odd or Even
      number = int(input("Enter a number: "))

      # Check if the number is odd or even
      if number % 2 == 0:
          print(f"{number} is an even number.")
      else:
          print(f"{number} is an odd number.")
```

Output:



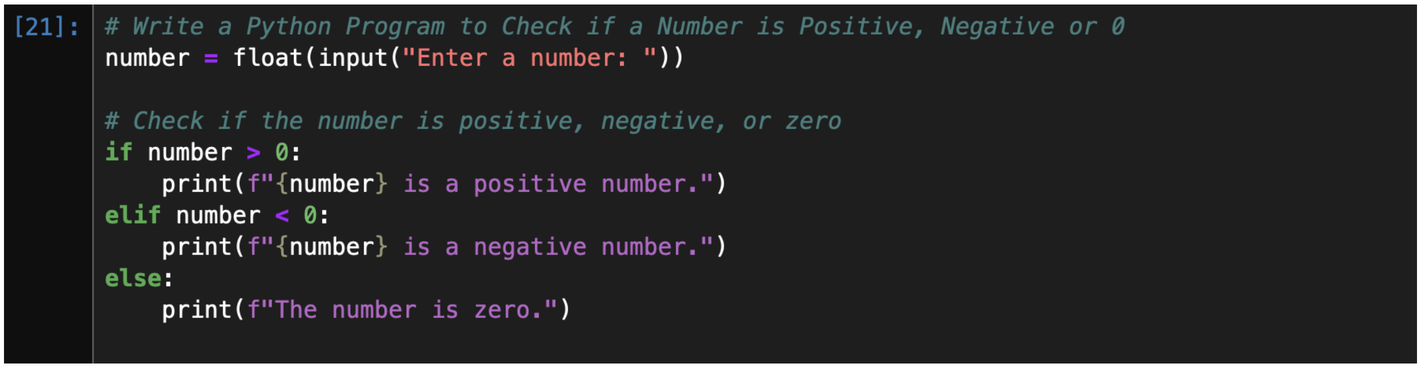
```
Enter a number:  2
2 is an even number.
```



```
Enter a number:  7
7 is an odd number.
```

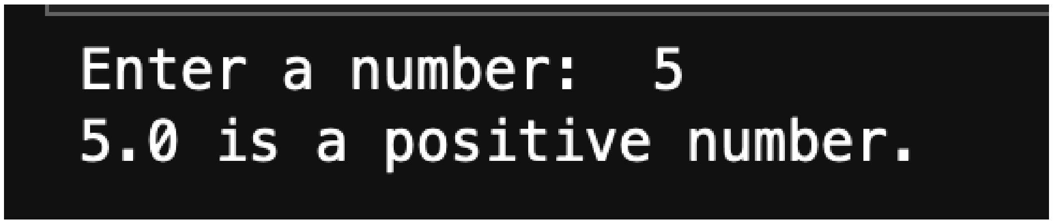
Program 2(b): Write a Python Program to Check if a Number is Positive, Negative or 0

Solution:

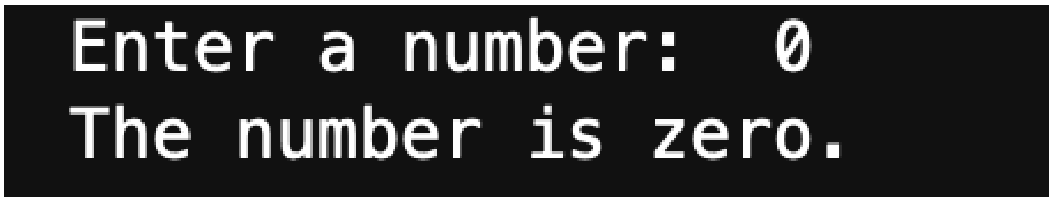


```
[21]: # Write a Python Program to Check if a Number is Positive, Negative or 0
number = float(input("Enter a number: "))

# Check if the number is positive, negative, or zero
if number > 0:
    print(f"{number} is a positive number.")
elif number < 0:
    print(f"{number} is a negative number.")
else:
    print(f"The number is zero.")
```

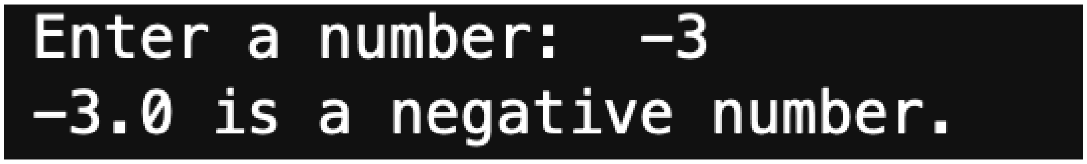
Output:



```
Enter a number:  5
5.0 is a positive number.
```



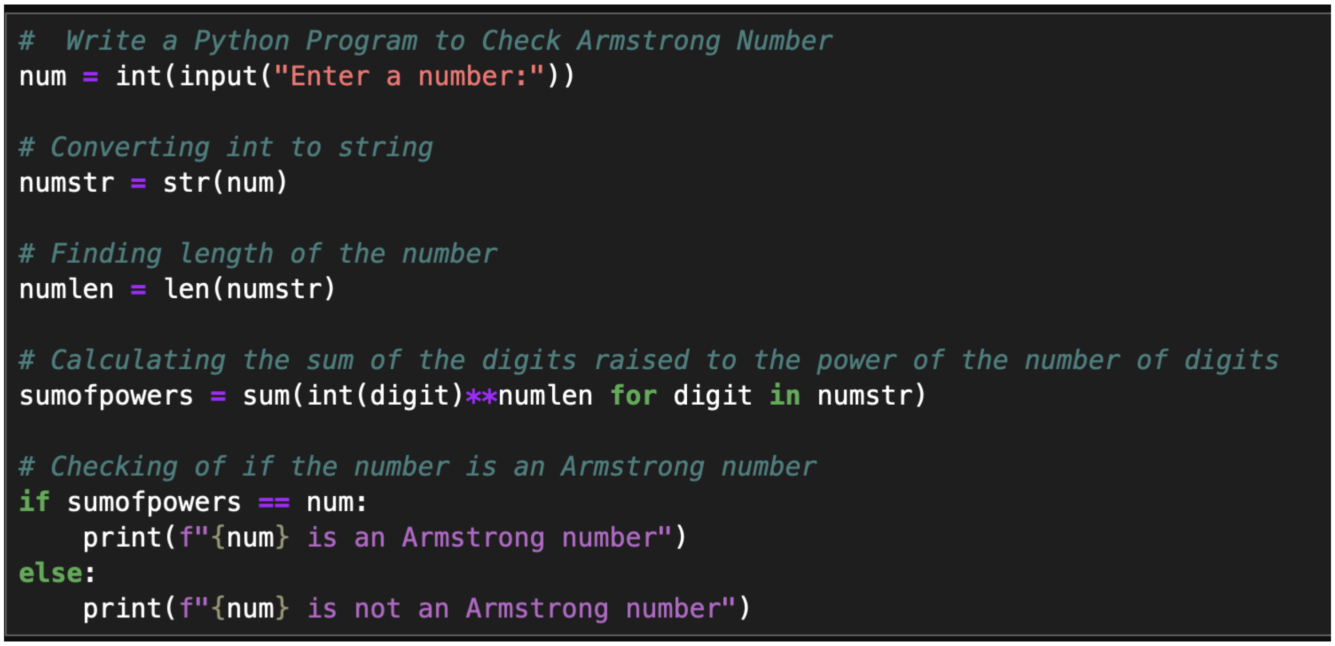
```
Enter a number:  0
The number is zero.
```



```
Enter a number:  -3
-3.0 is a negative number.
```

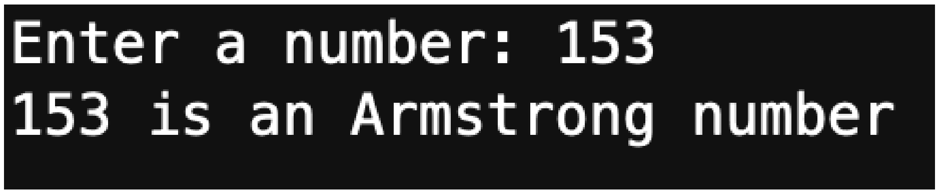
Program 2(c): Write a Python Program to Check Armstrong Number

Solution:

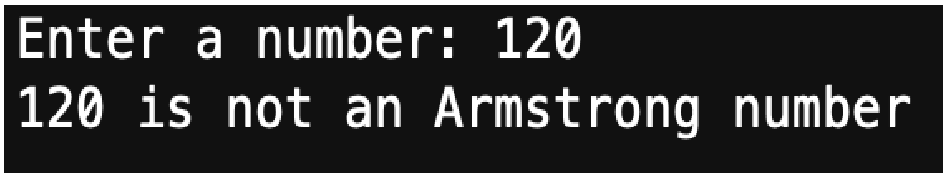


```python
#  Write a Python Program to Check Armstrong Number
num = int(input("Enter a number:"))

# Converting int to string
numstr = str(num)

# Finding length of the number
numlen = len(numstr)

# Calculating the sum of the digits raised to the power of the number of digits
sumofpowers = sum(int(digit)**numlen for digit in numstr)

# Checking of if the number is an Armstrong number
if sumofpowers == num:
    print(f"{num} is an Armstrong number")
else:
    print(f"{num} is not an Armstrong number")
```

Output:



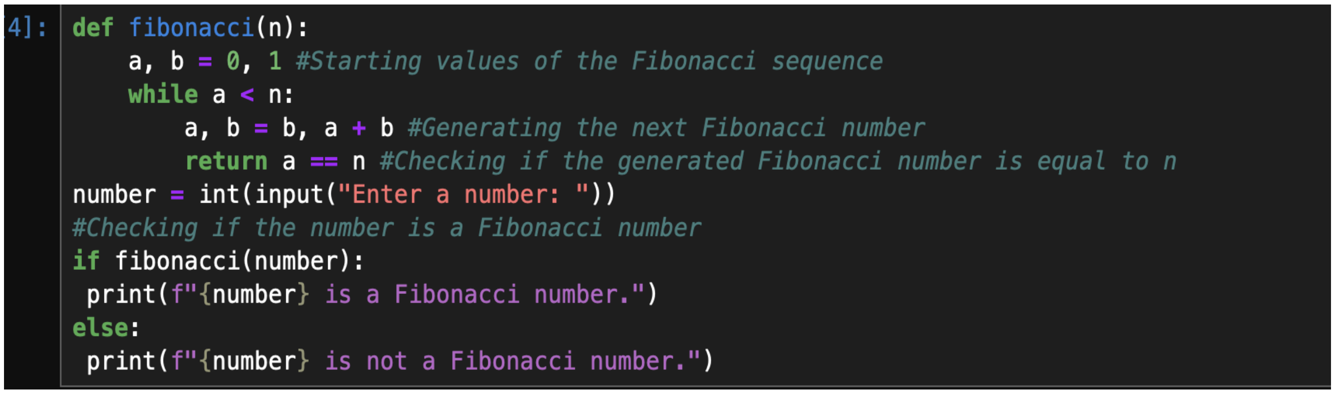
```
Enter a number: 153
153 is an Armstrong number
```



```
Enter a number: 120
120 is not an Armstrong number
```

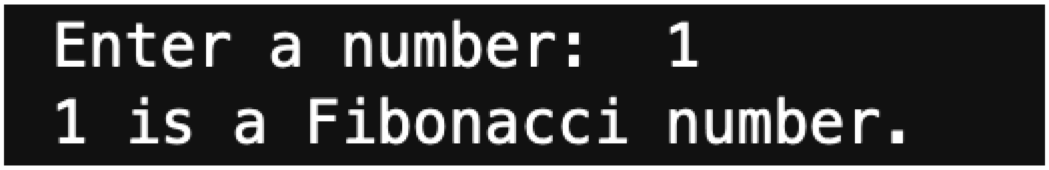
Program 3(a): Write a Python program to check if a given number is Fibonacci number

Solution:

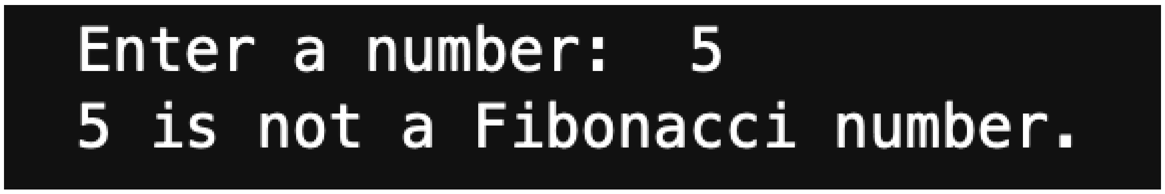


```
4]: def fibonacci(n):
        a, b = 0, 1 #Starting values of the Fibonacci sequence
        while a < n:
            a, b = b, a + b #Generating the next Fibonacci number
            return a == n #Checking if the generated Fibonacci number is equal to n
    number = int(input("Enter a number: "))
    #Checking if the number is a Fibonacci number
    if fibonacci(number):
     print(f"{number} is a Fibonacci number.")
    else:
     print(f"{number} is not a Fibonacci number.")
```

Output:



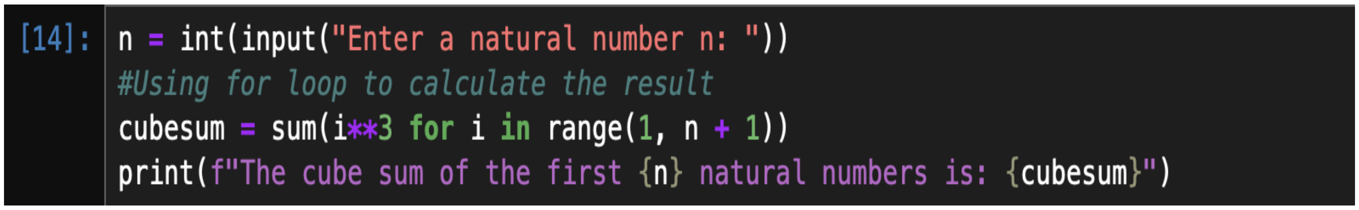
```
Enter a number:  1
1 is a Fibonacci number.
```



```
Enter a number:  5
5 is not a Fibonacci number.
```

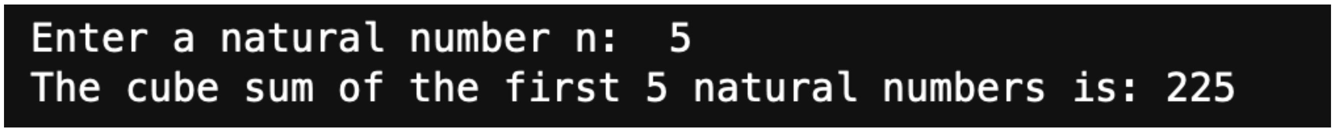
Program 3(b): Write a Python program to print cube sum of first n natural numbers

Solution:



```
[14]: n = int(input("Enter a natural number n: "))
      #Using for loop to calculate the result
      cubesum = sum(i**3 for i in range(1, n + 1))
      print(f"The cube sum of the first {n} natural numbers is: {cubesum}")
```

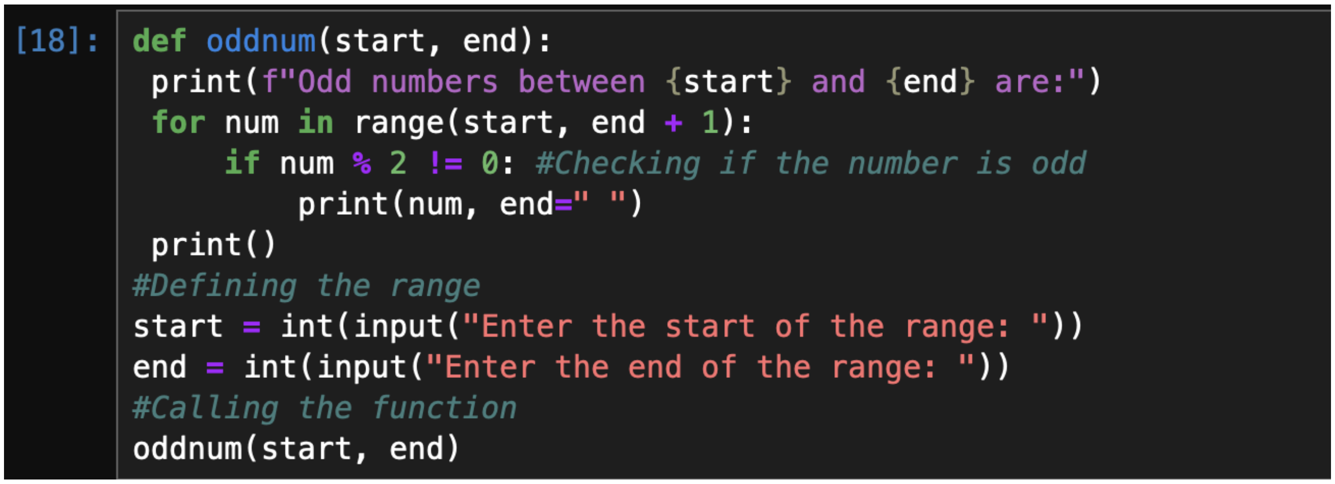
Output:



```
Enter a natural number n:  5
The cube sum of the first 5 natural numbers is: 225
```

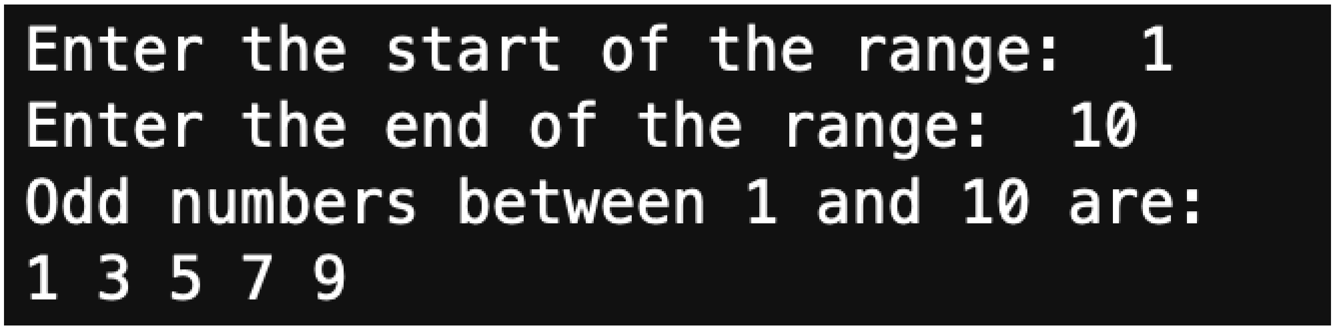
Program 3(c): Write a Python program to print all odd numbers in a range

Solution:



```
[18]: def oddnum(start, end):
       print(f"Odd numbers between {start} and {end} are:")
       for num in range(start, end + 1):
           if num % 2 != 0: #Checking if the number is odd
               print(num, end=" ")
       print()
      #Defining the range
      start = int(input("Enter the start of the range: "))
      end = int(input("Enter the end of the range: "))
      #Calling the function
      oddnum(start, end)
```

Output:

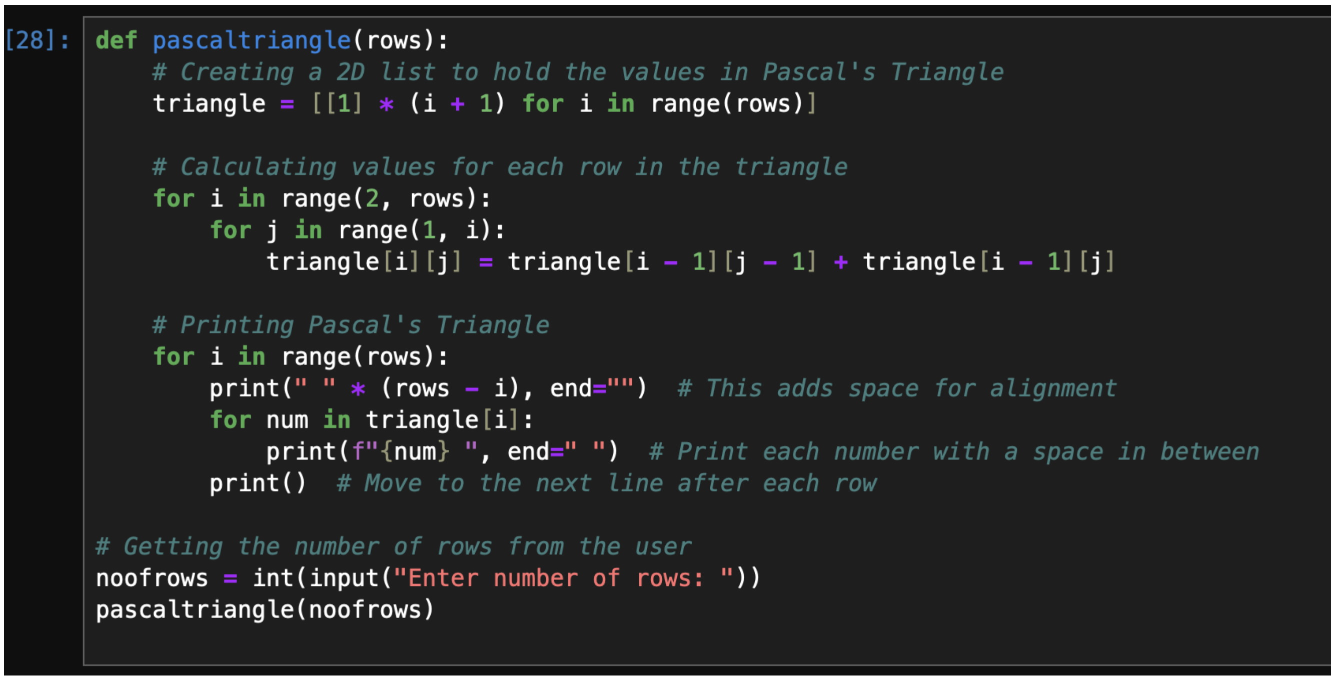


```
Enter the start of the range:  1
Enter the end of the range:  10
Odd numbers between 1 and 10 are:
1 3 5 7 9
```

Program 4(a): Write a Python Program to Print Pascal Triangle (Hint: Enter number of rows: 4)

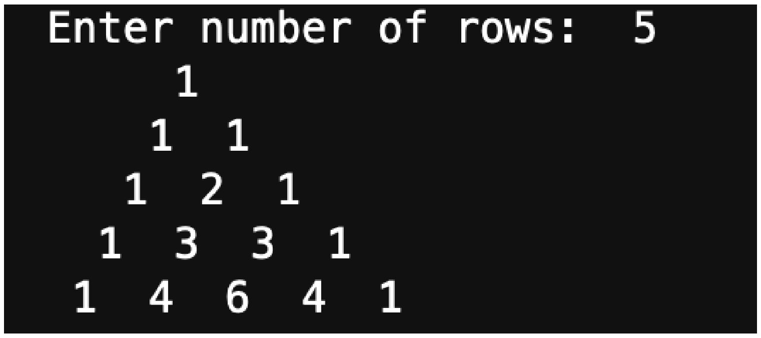
```
      1
    1   1
  1   2   1
1   3   3   1
```

Solution:



```
[28]: def pascaltriangle(rows):
          # Creating a 2D list to hold the values in Pascal's Triangle
          triangle = [[1] * (i + 1) for i in range(rows)]

          # Calculating values for each row in the triangle
          for i in range(2, rows):
              for j in range(1, i):
                  triangle[i][j] = triangle[i - 1][j - 1] + triangle[i - 1][j]

          # Printing Pascal's Triangle
          for i in range(rows):
              print(" " * (rows - i), end="")  # This adds space for alignment
              for num in triangle[i]:
                  print(f"{num} ", end=" ")  # Print each number with a space in between
              print()  # Move to the next line after each row

      # Getting the number of rows from the user
      noofrows = int(input("Enter number of rows: "))
      pascaltriangle(noofrows)
```

Output:

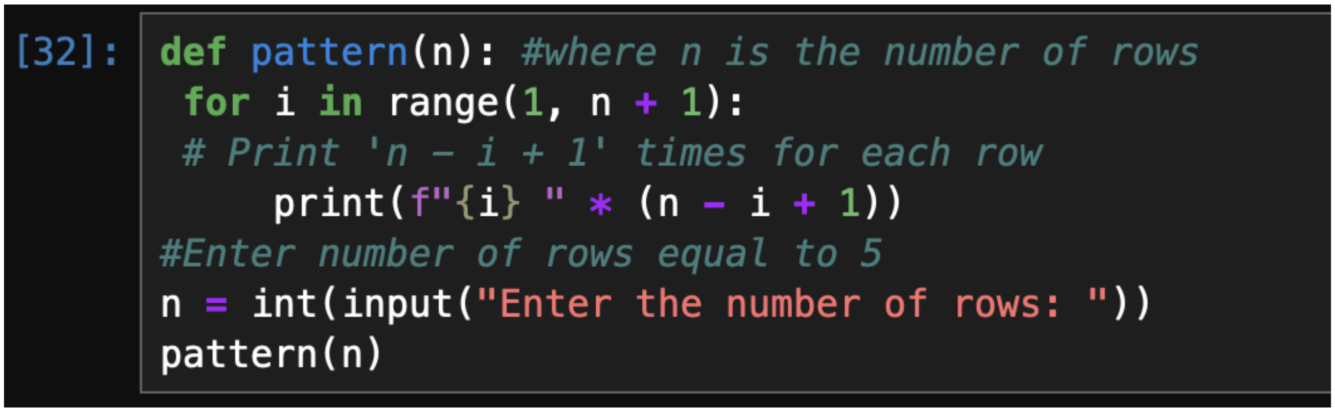


```
Enter number of rows:  5
     1
    1  1
   1  2  1
  1  3  3  1
 1  4  6  4  1
```

Program 4(b): Write a Python Program to Draw the following Pattern for n number:

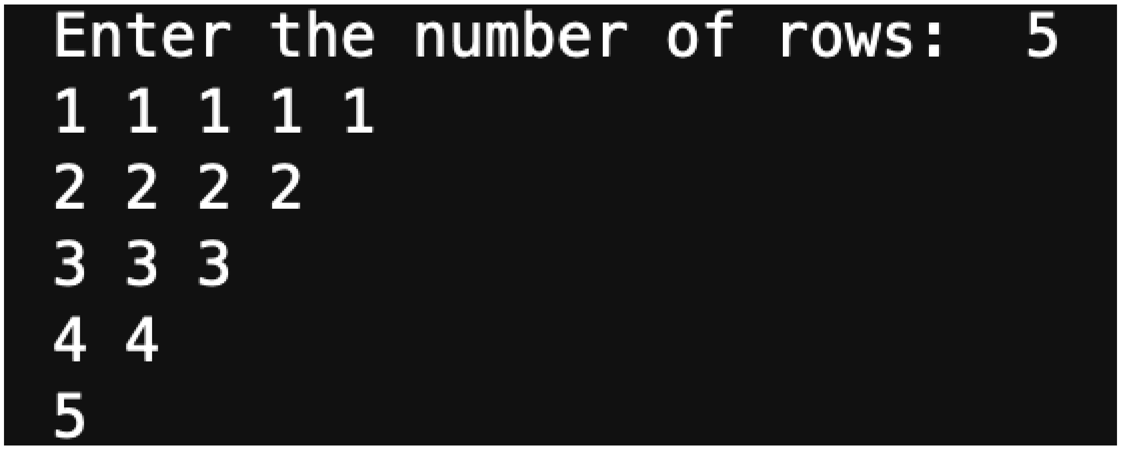
1 1 1 1 1
2 2 2 2
3 3 3
4 4
5

Solution:



```
[32]: def pattern(n): #where n is the number of rows
       for i in range(1, n + 1):
       # Print 'n − i + 1' times for each row
           print(f"{i} " * (n - i + 1))
      #Enter number of rows equal to 5
      n = int(input("Enter the number of rows: "))
      pattern(n)
```

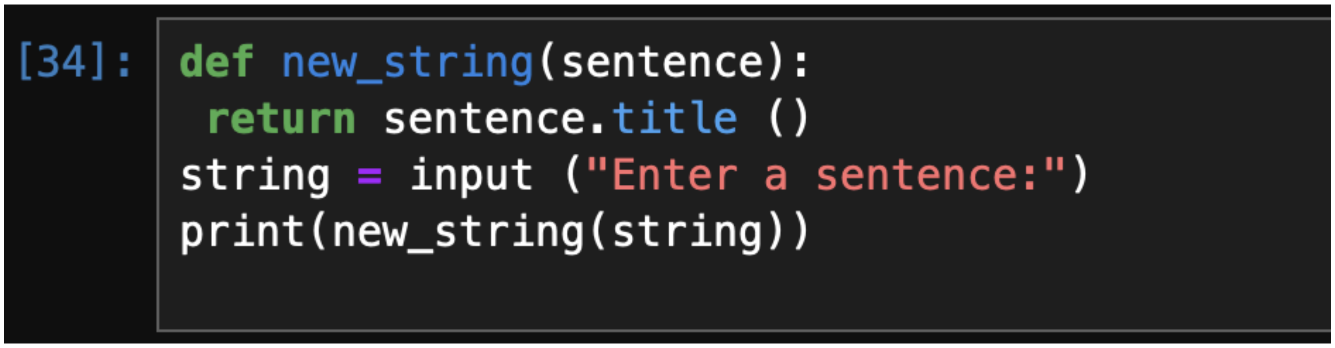
Output:



```
Enter the number of rows:  5
1 1 1 1 1
2 2 2 2
3 3 3
4 4
5
```

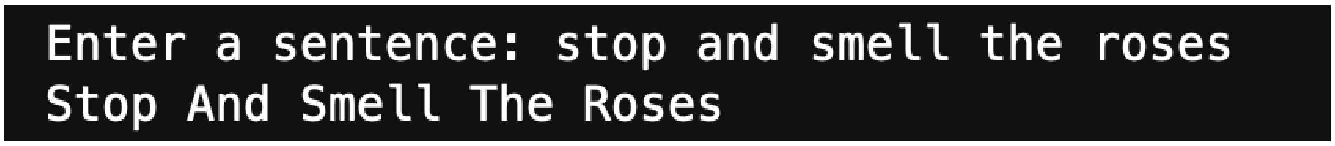
Program 5: Write a program with a function that accepts a string from the keyboard and creates a new string after converting the character of each word capitalized. For instance, if the sentence is "stop and smell the roses" the output should be "Stop And Smell The Roses"

Solution:



```
[34]: def new_string(sentence):
       return sentence.title ()
      string = input ("Enter a sentence:")
      print(new_string(string))
```

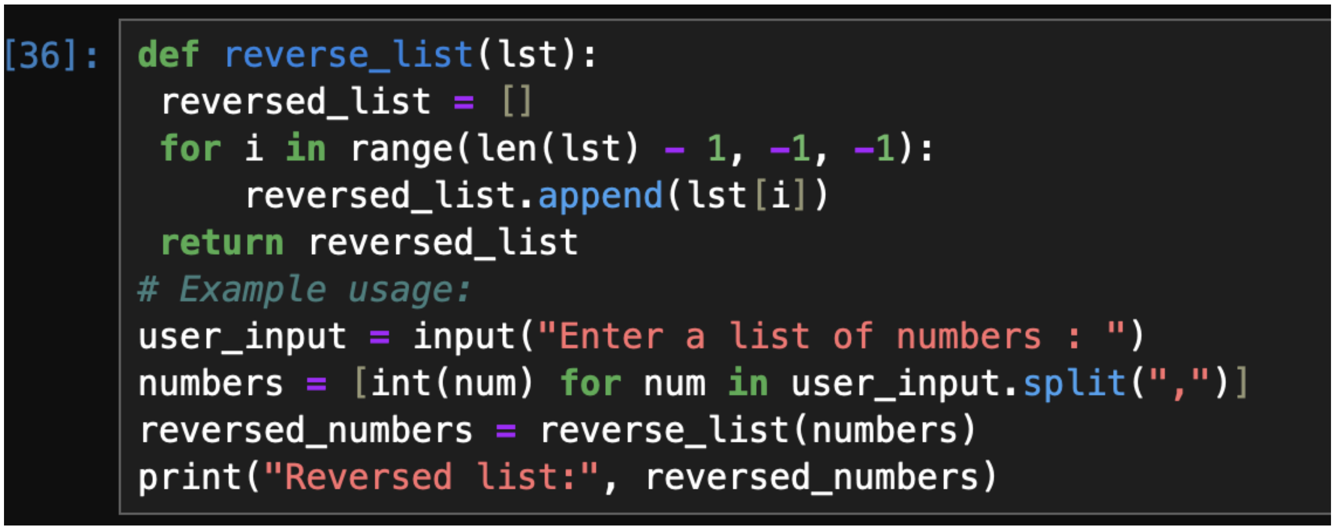
Output:



```
Enter a sentence: stop and smell the roses
Stop And Smell The Roses
```

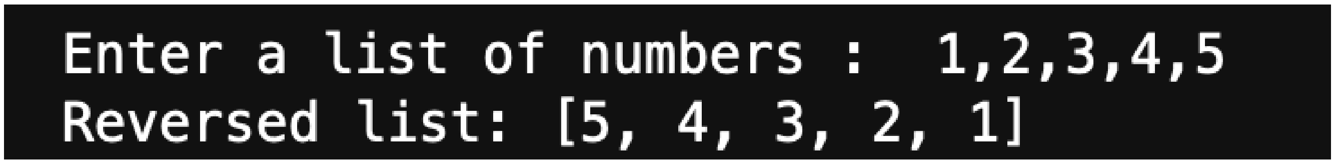
Program 6(a): Write a program that accepts a list from the user. Your program should reverse the content of the list and display it. Do not use the reverse () method.

Solution:



```
[36]: def reverse_list(lst):
       reversed_list = []
       for i in range(len(lst) - 1, -1, -1):
           reversed_list.append(lst[i])
       return reversed_list
      # Example usage:
      user_input = input("Enter a list of numbers : ")
      numbers = [int(num) for num in user_input.split(",")]
      reversed_numbers = reverse_list(numbers)
      print("Reversed list:", reversed_numbers)
```

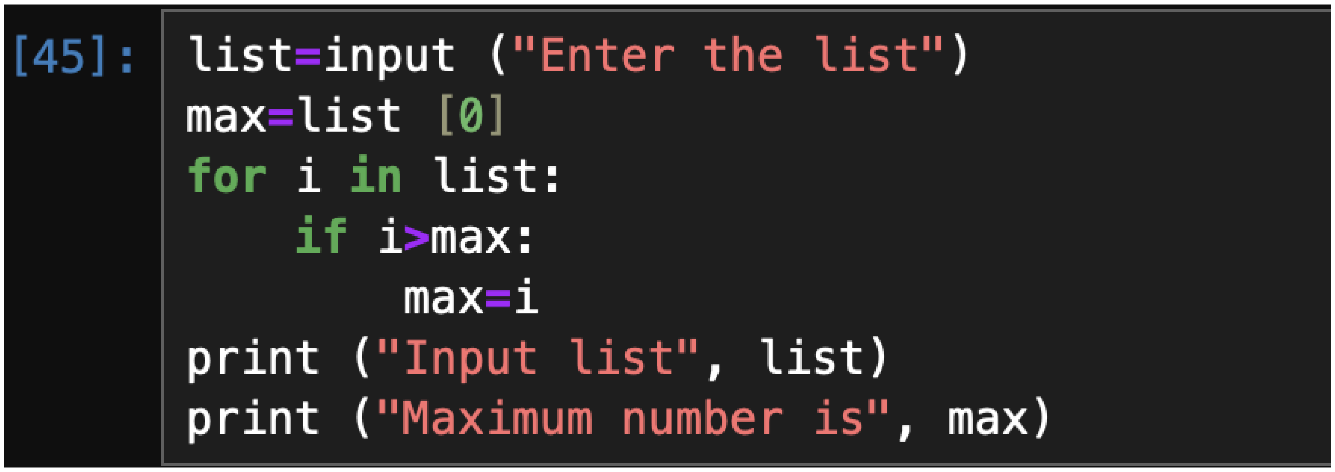
Output:



```
Enter a list of numbers :  1,2,3,4,5
Reversed list: [5, 4, 3, 2, 1]
```

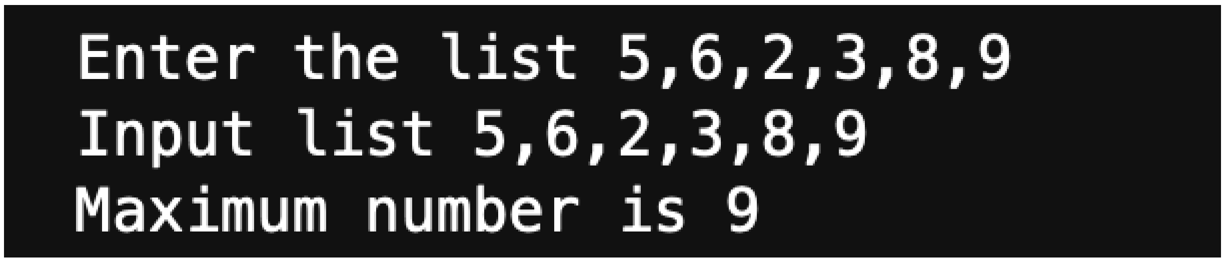
Program 6(b): Find and display the largest number of a list without using built-in function max ().

Solution:



```
[45]: list=input ("Enter the list")
      max=list [0]
      for i in list:
          if i>max:
              max=i
      print ("Input list", list)
      print ("Maximum number is", max)
```

Output:



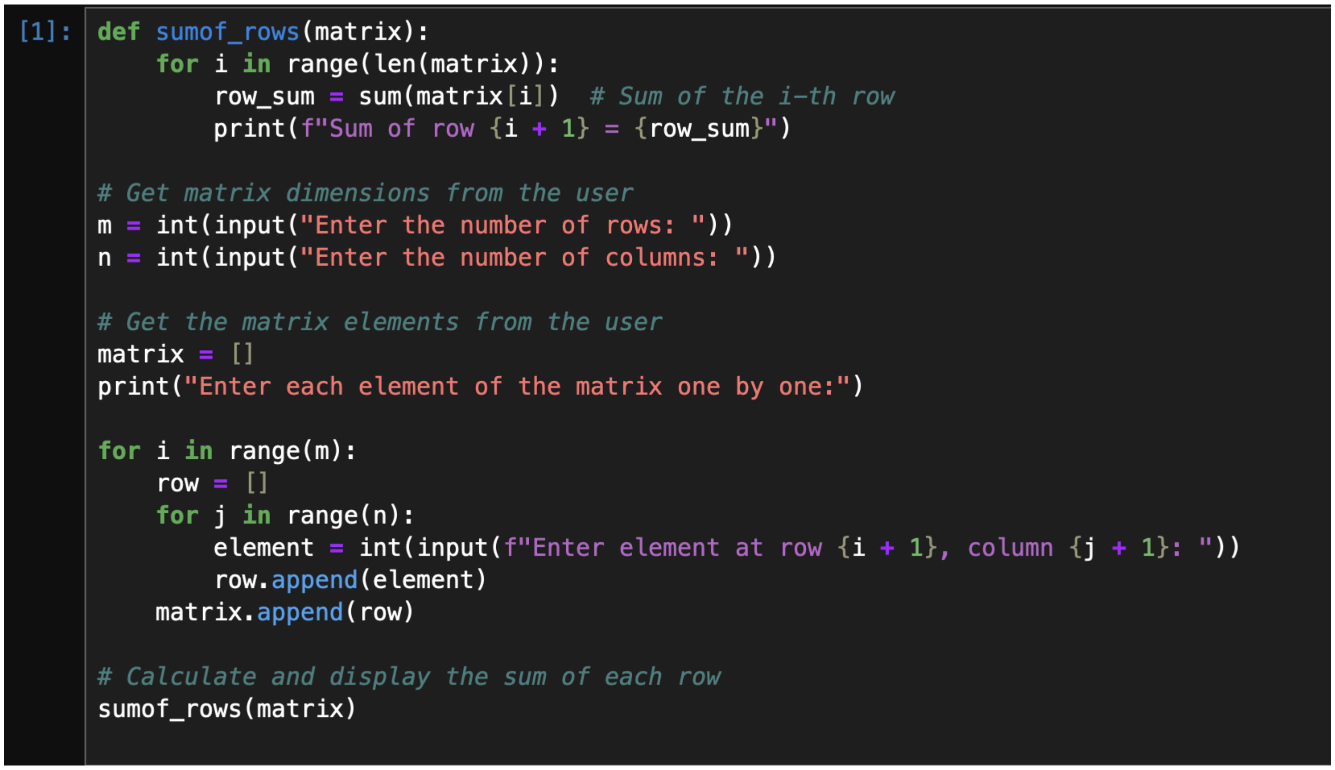
```
Enter the list 5,6,2,3,8,9
Input list 5,6,2,3,8,9
Maximum number is 9
```

Program 7: Find the sum of each row of matrix of size m x n. For example, for the following matrix output will be like this:

2	11	7	12
5	2	9	15
8	3	10	42

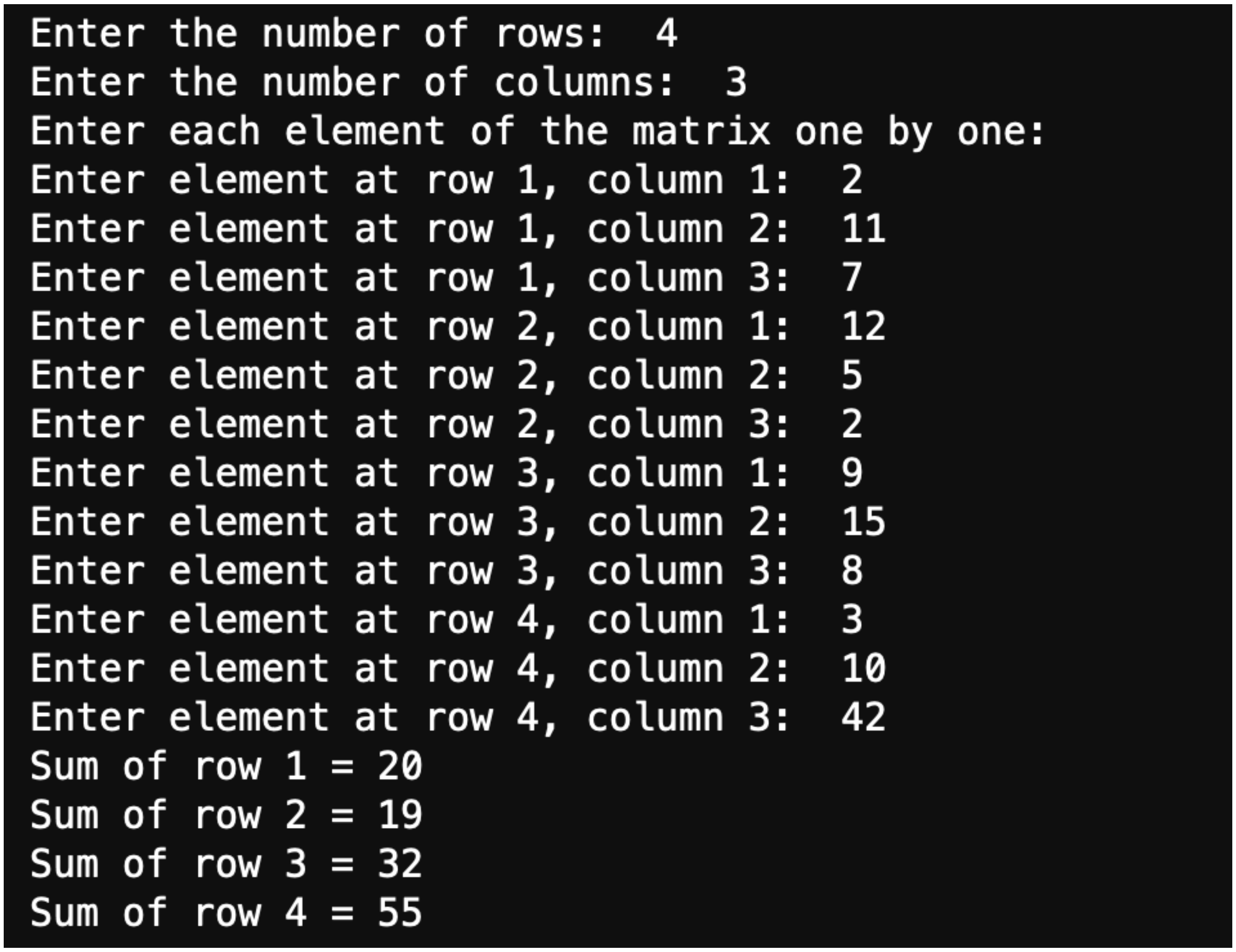
Sum of row 1 = 32 Sum of row 2 = 31 Sum of row 3 = 63

<u>Solution:</u>



```
[1]: def sumof_rows(matrix):
         for i in range(len(matrix)):
             row_sum = sum(matrix[i])  # Sum of the i-th row
             print(f"Sum of row {i + 1} = {row_sum}")

     # Get matrix dimensions from the user
     m = int(input("Enter the number of rows: "))
     n = int(input("Enter the number of columns: "))

     # Get the matrix elements from the user
     matrix = []
     print("Enter each element of the matrix one by one:")

     for i in range(m):
         row = []
         for j in range(n):
             element = int(input(f"Enter element at row {i + 1}, column {j + 1}: "))
             row.append(element)
         matrix.append(row)

     # Calculate and display the sum of each row
     sumof_rows(matrix)
```

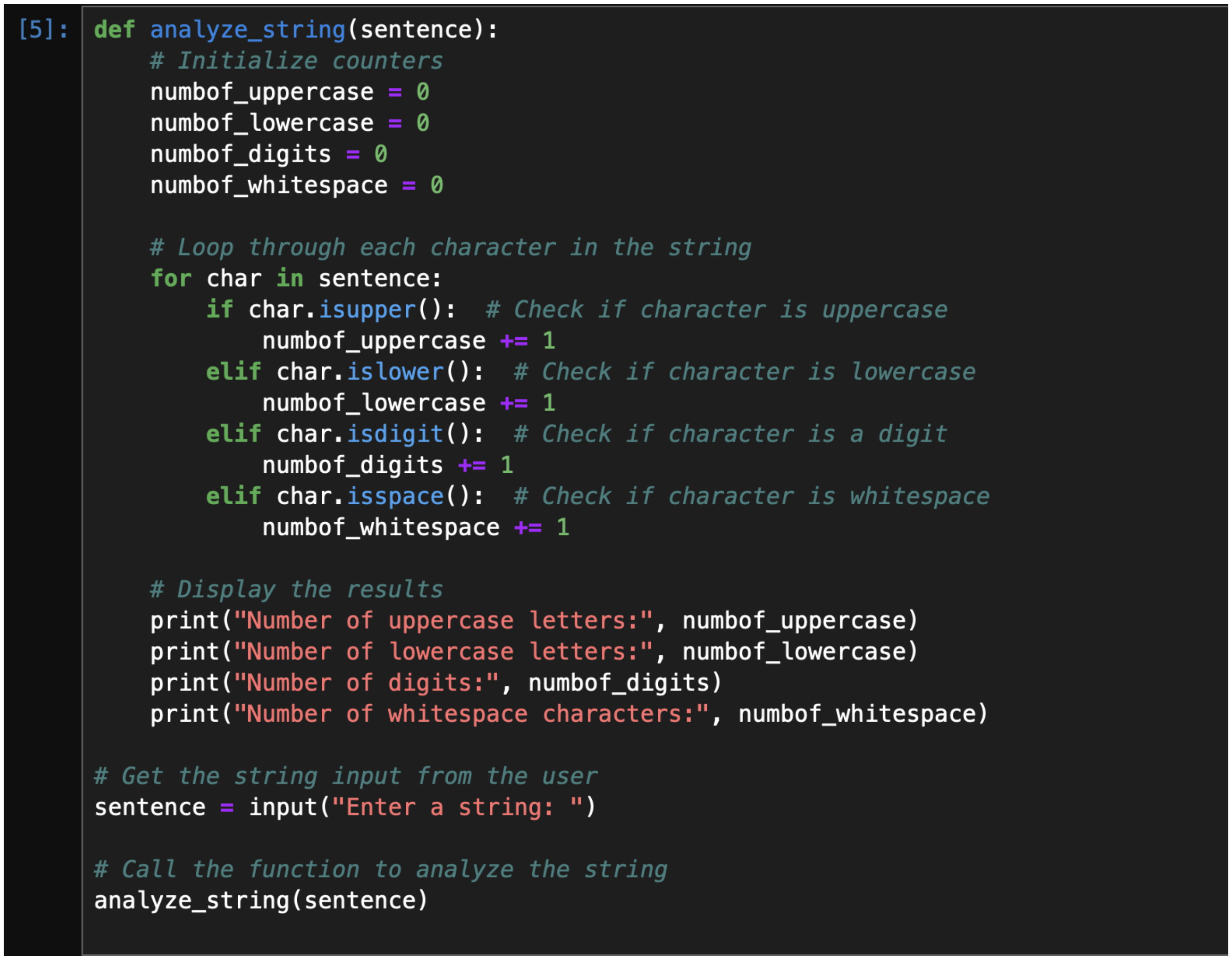
Output:



```
Enter the number of rows:  4
Enter the number of columns:  3
Enter each element of the matrix one by one:
Enter element at row 1, column 1:  2
Enter element at row 1, column 2:  11
Enter element at row 1, column 3:  7
Enter element at row 2, column 1:  12
Enter element at row 2, column 2:  5
Enter element at row 2, column 3:  2
Enter element at row 3, column 1:  9
Enter element at row 3, column 2:  15
Enter element at row 3, column 3:  8
Enter element at row 4, column 1:  3
Enter element at row 4, column 2:  10
Enter element at row 4, column 3:  42
Sum of row 1 = 20
Sum of row 2 = 19
Sum of row 3 = 32
Sum of row 4 = 55
```

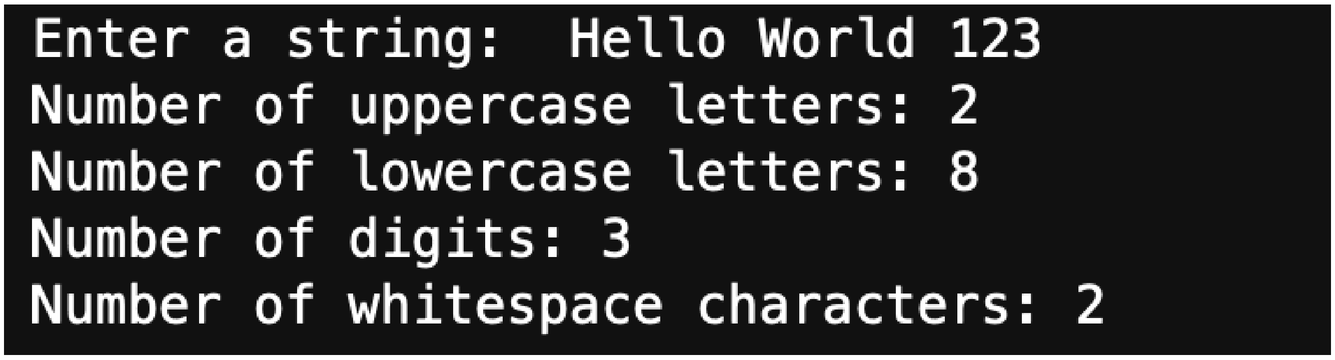
Program 8(a): Write a program that reads a string from keyboard and display:
* The number of uppercase letters in the string.
* The number of lowercase letters in the string.
* The number of digits in the string.
* The number of whitespace characters in the string.

<u>Solution:</u>



```
[5]: def analyze_string(sentence):
         # Initialize counters
         numbof_uppercase = 0
         numbof_lowercase = 0
         numbof_digits = 0
         numbof_whitespace = 0

         # Loop through each character in the string
         for char in sentence:
             if char.isupper():  # Check if character is uppercase
                 numbof_uppercase += 1
             elif char.islower():  # Check if character is lowercase
                 numbof_lowercase += 1
             elif char.isdigit():  # Check if character is a digit
                 numbof_digits += 1
             elif char.isspace():  # Check if character is whitespace
                 numbof_whitespace += 1

         # Display the results
         print("Number of uppercase letters:", numbof_uppercase)
         print("Number of lowercase letters:", numbof_lowercase)
         print("Number of digits:", numbof_digits)
         print("Number of whitespace characters:", numbof_whitespace)

     # Get the string input from the user
     sentence = input("Enter a string: ")

     # Call the function to analyze the string
     analyze_string(sentence)
```

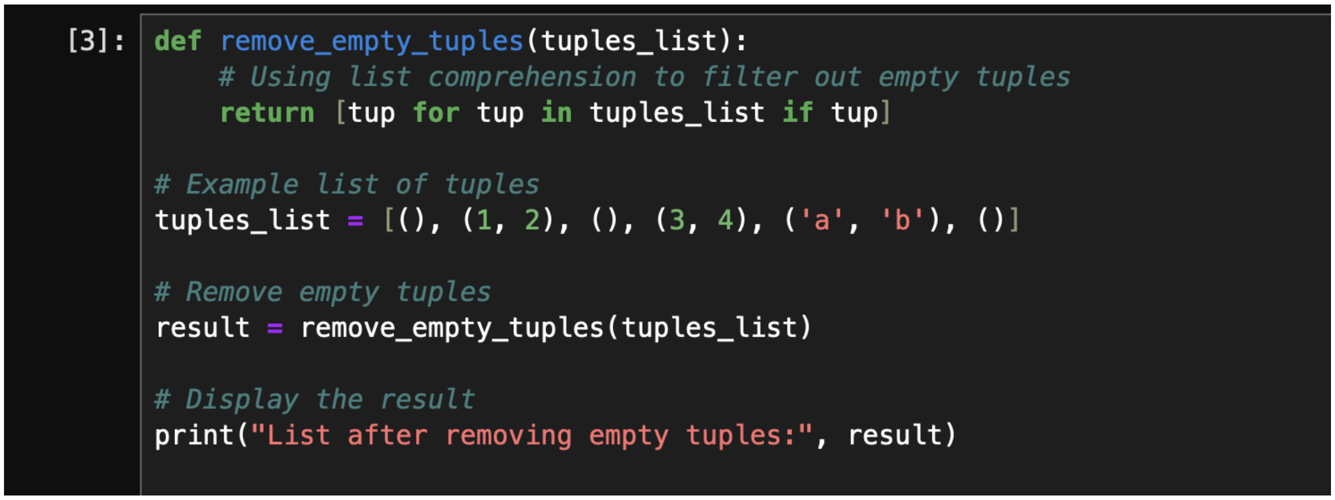
Output:



```
Enter a string:  Hello World 123
Number of uppercase letters: 2
Number of lowercase letters: 8
Number of digits: 3
Number of whitespace characters: 2
```

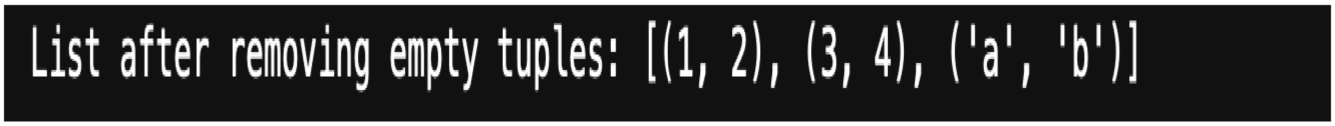
Program 8(b): Write a Python program to remove an empty tuple(s) from a list of tuples

Solution:



```
[3]: def remove_empty_tuples(tuples_list):
         # Using list comprehension to filter out empty tuples
         return [tup for tup in tuples_list if tup]

     # Example list of tuples
     tuples_list = [(), (1, 2), (), (3, 4), ('a', 'b'), ()]

     # Remove empty tuples
     result = remove_empty_tuples(tuples_list)

     # Display the result
     print("List after removing empty tuples:", result)
```

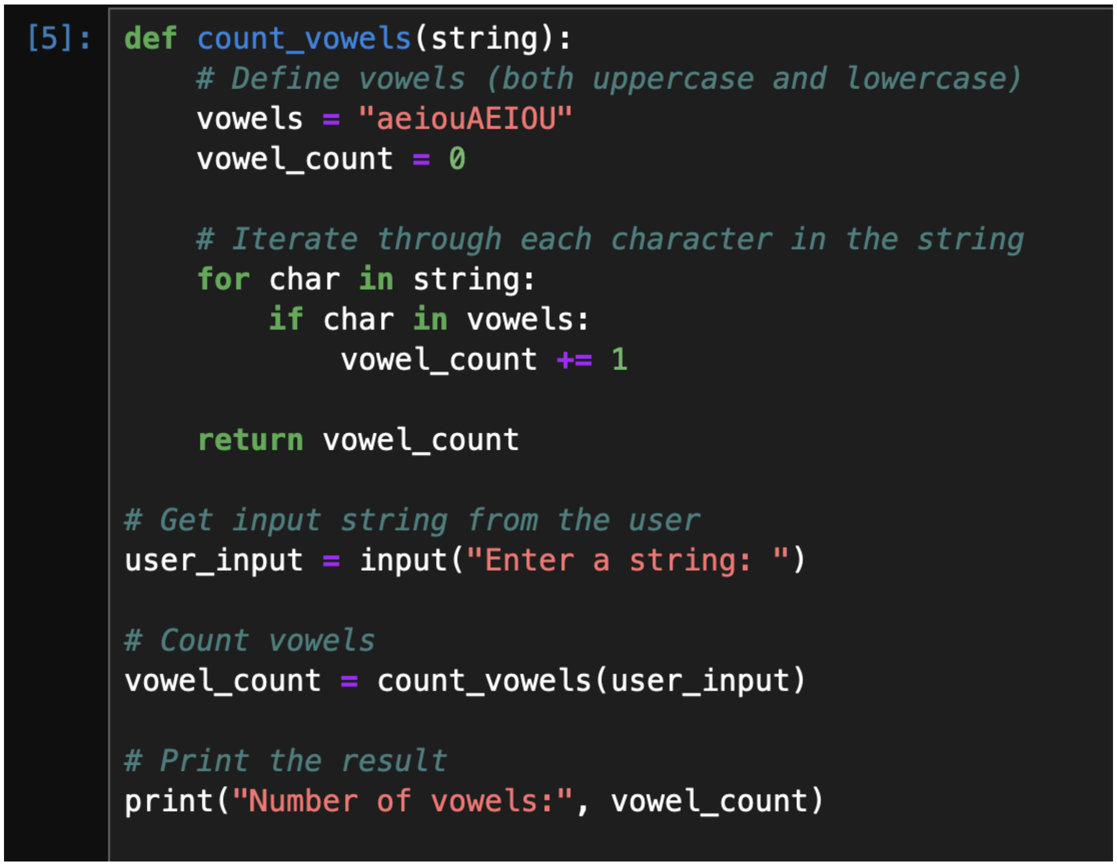
Output:



```
List after removing empty tuples: [(1, 2), (3, 4), ('a', 'b')]
```

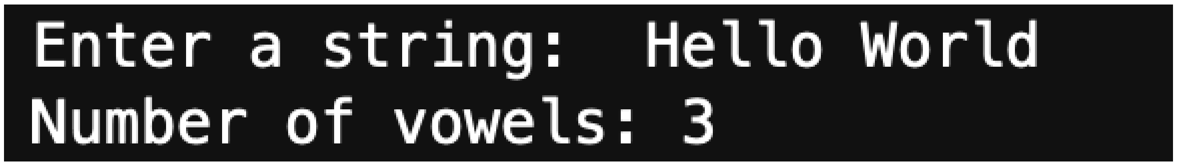
Program 8(c):Python Program to Count the Number of Vowels in a String.

Solution:



```
[5]: def count_vowels(string):
         # Define vowels (both uppercase and lowercase)
         vowels = "aeiouAEIOU"
         vowel_count = 0

         # Iterate through each character in the string
         for char in string:
             if char in vowels:
                 vowel_count += 1

         return vowel_count

     # Get input string from the user
     user_input = input("Enter a string: ")

     # Count vowels
     vowel_count = count_vowels(user_input)

     # Print the result
     print("Number of vowels:", vowel_count)
```

Output:

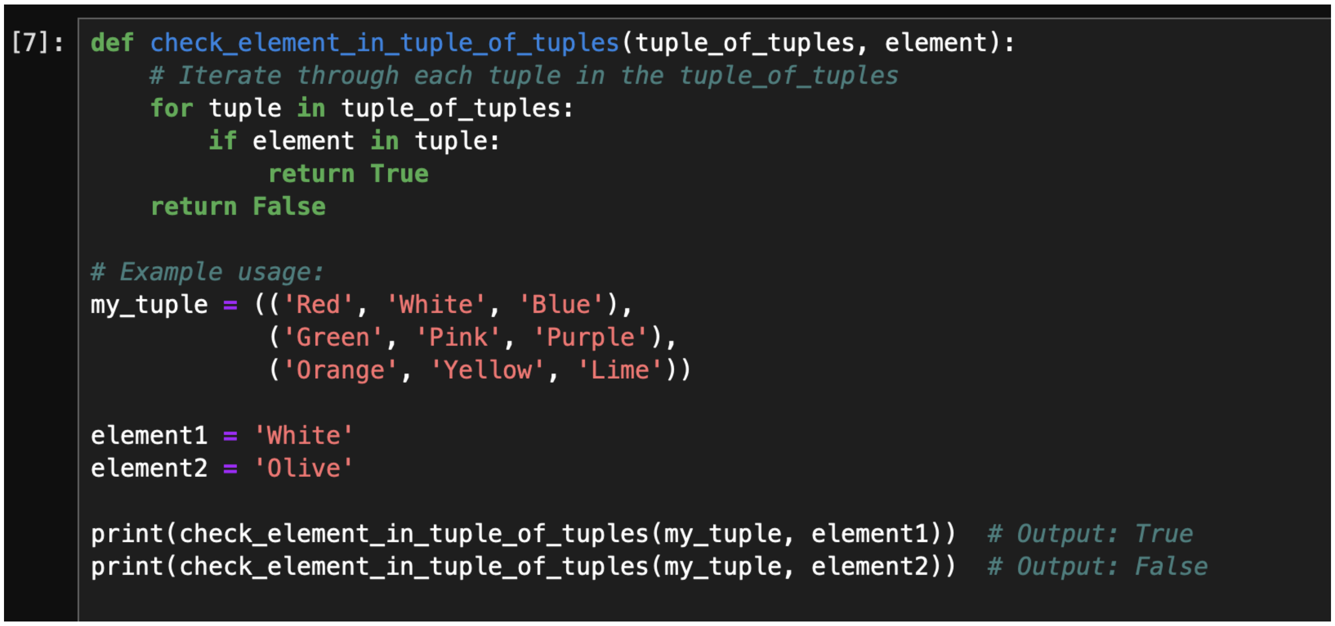


```
Enter a string:  Hello World
Number of vowels: 3
```

Program 9(a): Write a Python program to check if a specified element presents in a tuple of tuples.

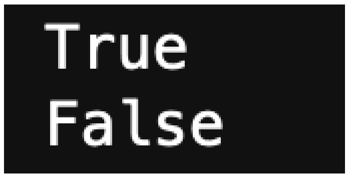
Original list:
((‘Red’ ,’White’ , ‘Blue’),(‘Green’, ’Pink’ , ‘Purple’), (‘Orange’, ‘Yellow’, ‘Lime’))
Check if White present in said tuple of tuples! True
Check if Olive is present in said tuple of tuples! False

Solution:



```
[7]: def check_element_in_tuple_of_tuples(tuple_of_tuples, element):
         # Iterate through each tuple in the tuple_of_tuples
         for tuple in tuple_of_tuples:
             if element in tuple:
                 return True
         return False

     # Example usage:
     my_tuple = (('Red', 'White', 'Blue'),
                 ('Green', 'Pink', 'Purple'),
                 ('Orange', 'Yellow', 'Lime'))

     element1 = 'White'
     element2 = 'Olive'

     print(check_element_in_tuple_of_tuples(my_tuple, element1))  # Output: True
     print(check_element_in_tuple_of_tuples(my_tuple, element2))  # Output: False
```

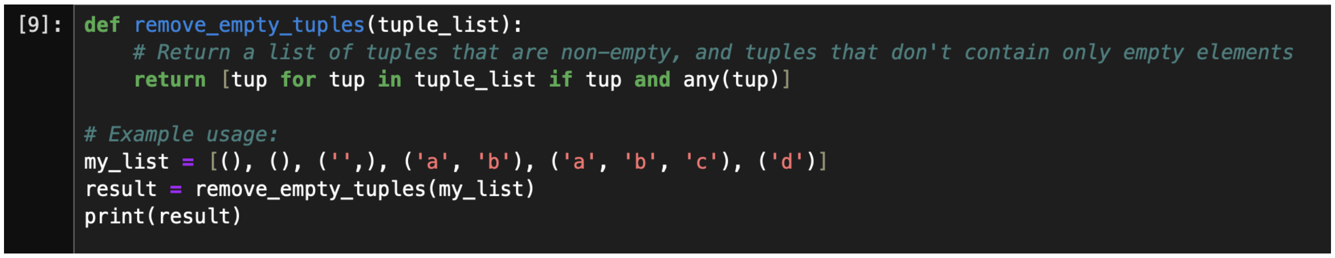
Output:



```
True
False
```

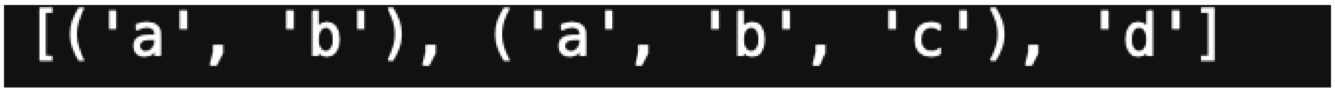
Program 9(b): Write a Python program to remove an empty tuple(s) from a list of tuples.
Sample data: [(), (), ('',), ('a', 'b'), ('a', 'b', 'c'), ('d')]
Expected output: [('',), ('a', 'b'), ('a', 'b', 'c'), 'd']

Solution:



```
[9]: def remove_empty_tuples(tuple_list):
         # Return a list of tuples that are non-empty, and tuples that don't contain only empty elements
         return [tup for tup in tuple_list if tup and any(tup)]

     # Example usage:
     my_list = [(), (), ('',), ('a', 'b'), ('a', 'b', 'c'), ('d')]
     result = remove_empty_tuples(my_list)
     print(result)
```

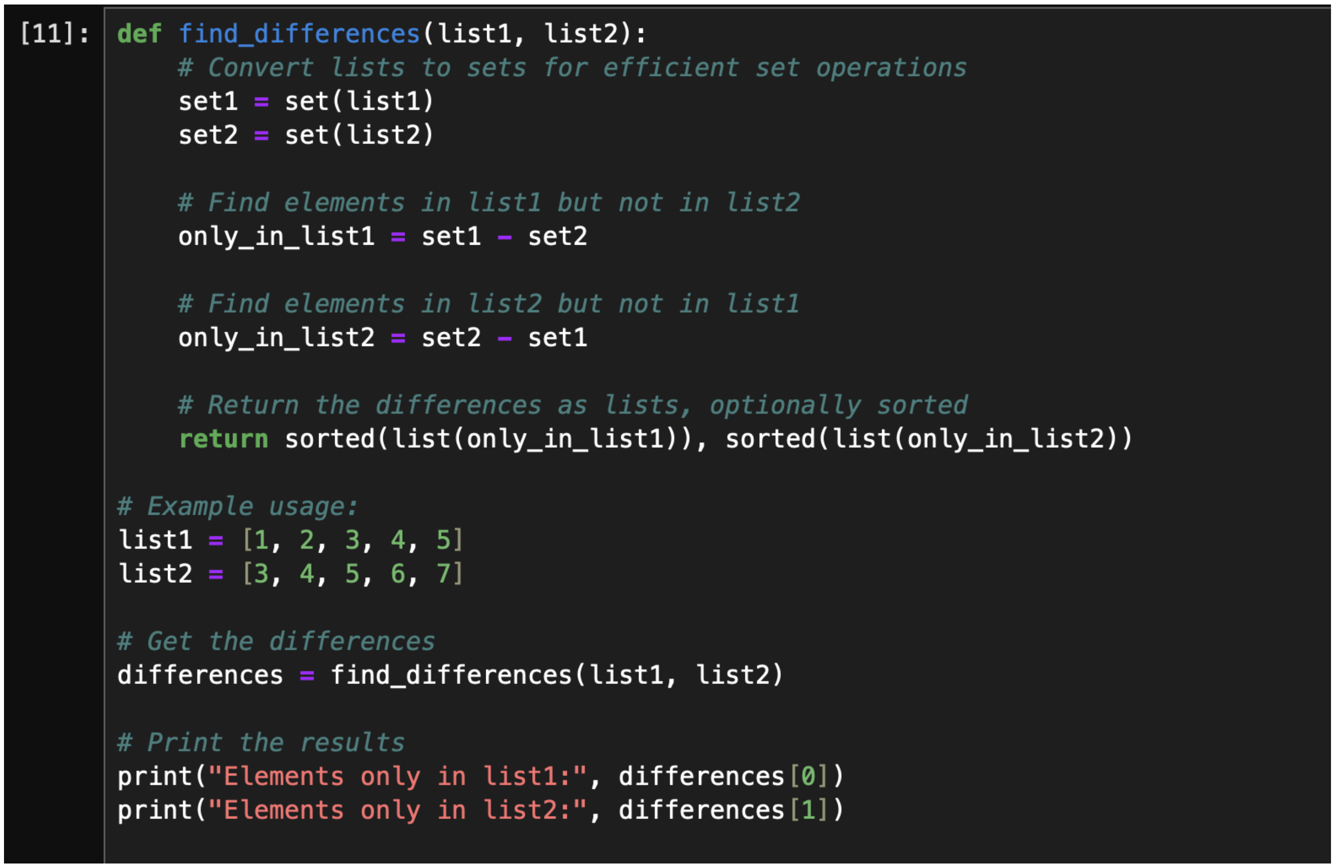
Output:



```
[('a', 'b'), ('a', 'b', 'c'), 'd']
```

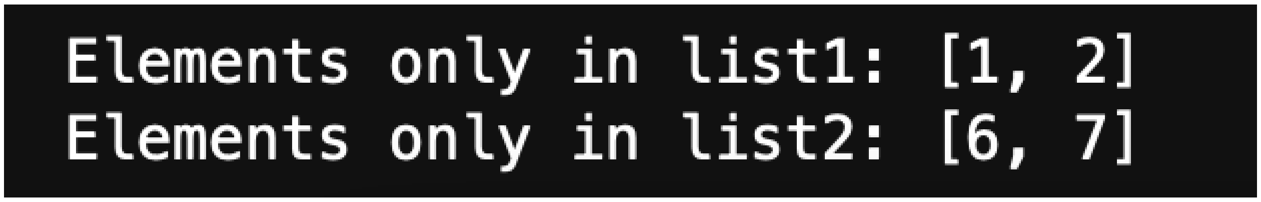
Program 10: Write a Program in Python to Find the Differences Between Two Lists Using Sets.

Solution:



```
[11]: def find_differences(list1, list2):
          # Convert lists to sets for efficient set operations
          set1 = set(list1)
          set2 = set(list2)

          # Find elements in list1 but not in list2
          only_in_list1 = set1 - set2

          # Find elements in list2 but not in list1
          only_in_list2 = set2 - set1

          # Return the differences as lists, optionally sorted
          return sorted(list(only_in_list1)), sorted(list(only_in_list2))

      # Example usage:
      list1 = [1, 2, 3, 4, 5]
      list2 = [3, 4, 5, 6, 7]

      # Get the differences
      differences = find_differences(list1, list2)

      # Print the results
      print("Elements only in list1:", differences[0])
      print("Elements only in list2:", differences[1])
```

Output:



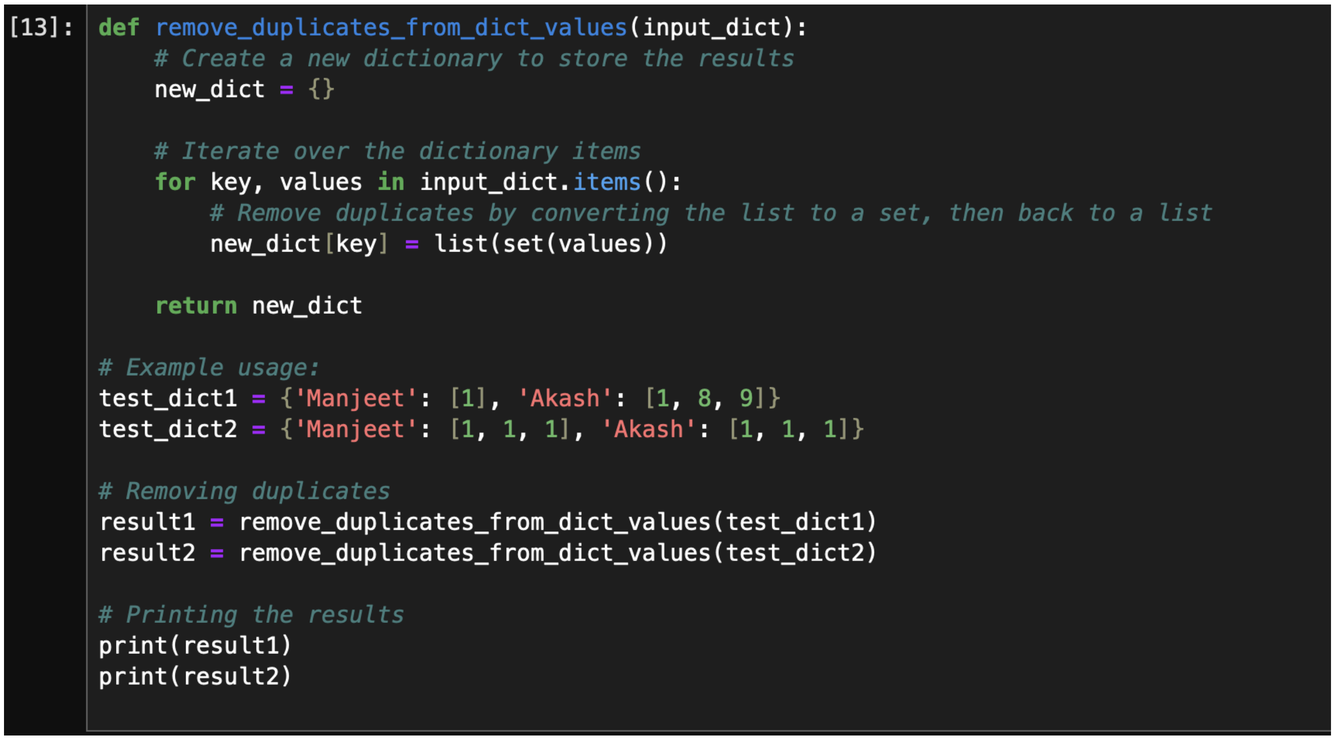
```
Elements only in list1: [1, 2]
Elements only in list2: [6, 7]
```

Program 11(a) :) Write a Python program Remove duplicate values across Dictionary Values.

Input : test_dict = {‘Manjeet’: [1], ‘Akash’: [1, 8, 9]}
Output : {‘Manjeet’: [], ‘Akash’: [8, 9]}

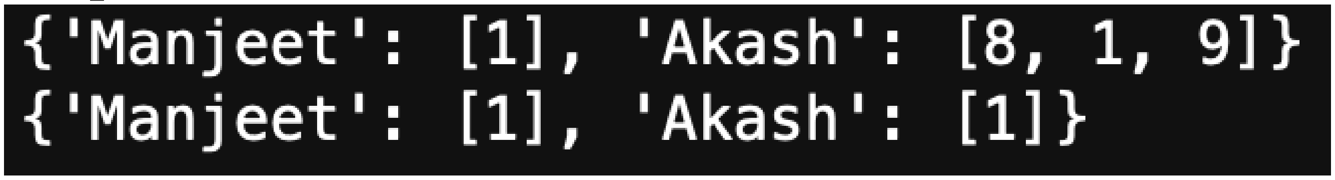
Input : test_dict = {‘Manjeet’: [1, 1, 1], ‘Akash’: [1, 1, 1]}
Output : {‘Manjeet’: [], ‘Akash’: []}

Solution:



```
[13]: def remove_duplicates_from_dict_values(input_dict):
          # Create a new dictionary to store the results
          new_dict = {}

          # Iterate over the dictionary items
          for key, values in input_dict.items():
              # Remove duplicates by converting the list to a set, then back to a list
              new_dict[key] = list(set(values))

          return new_dict

      # Example usage:
      test_dict1 = {'Manjeet': [1], 'Akash': [1, 8, 9]}
      test_dict2 = {'Manjeet': [1, 1, 1], 'Akash': [1, 1, 1]}

      # Removing duplicates
      result1 = remove_duplicates_from_dict_values(test_dict1)
      result2 = remove_duplicates_from_dict_values(test_dict2)

      # Printing the results
      print(result1)
      print(result2)
```

Output:



```
{'Manjeet': [1], 'Akash': [8, 1, 9]}
{'Manjeet': [1], 'Akash': [1]}
```

Program 11(b) : Write a Python program to Count the frequencies in a list using a dictionary in Python.

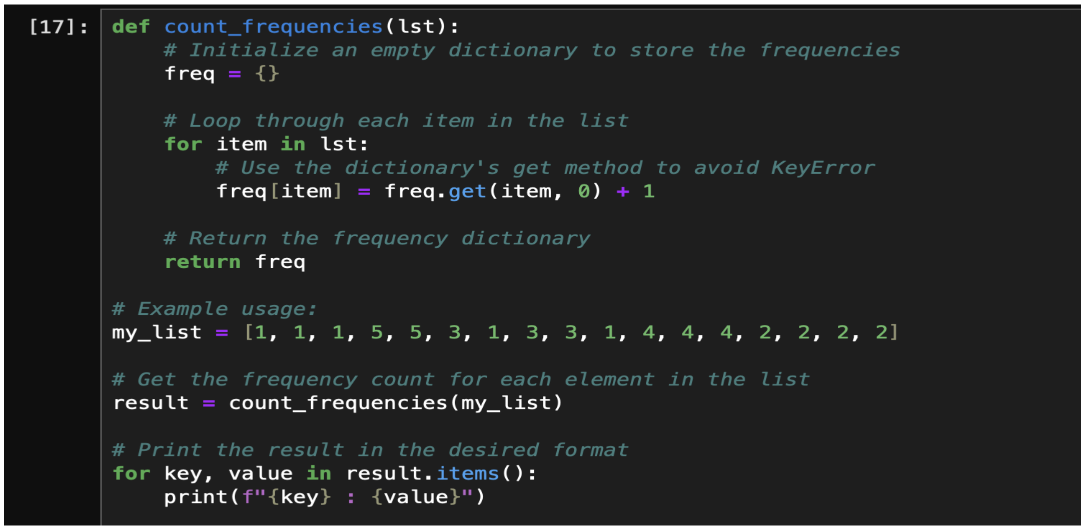
Input : [1, 1, 1, 5, 5, 3, 1, 3, 3, 1,4, 4, 4, 2, 2, 2, 2]

Output :

```
1 : 5
2 : 4
3 : 3
4 : 3
```

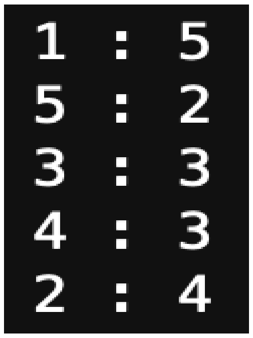
Explanation : Here 1 occurs 5 times, 2 occurs 4 times and so on…

Solution:



```
[17]: def count_frequencies(lst):
          # Initialize an empty dictionary to store the frequencies
          freq = {}

          # Loop through each item in the list
          for item in lst:
              # Use the dictionary's get method to avoid KeyError
              freq[item] = freq.get(item, 0) + 1

          # Return the frequency dictionary
          return freq

      # Example usage:
      my_list = [1, 1, 1, 5, 5, 3, 1, 3, 3, 1, 4, 4, 4, 2, 2, 2, 2]

      # Get the frequency count for each element in the list
      result = count_frequencies(my_list)

      # Print the result in the desired format
      for key, value in result.items():
          print(f"{key} : {value}")
```

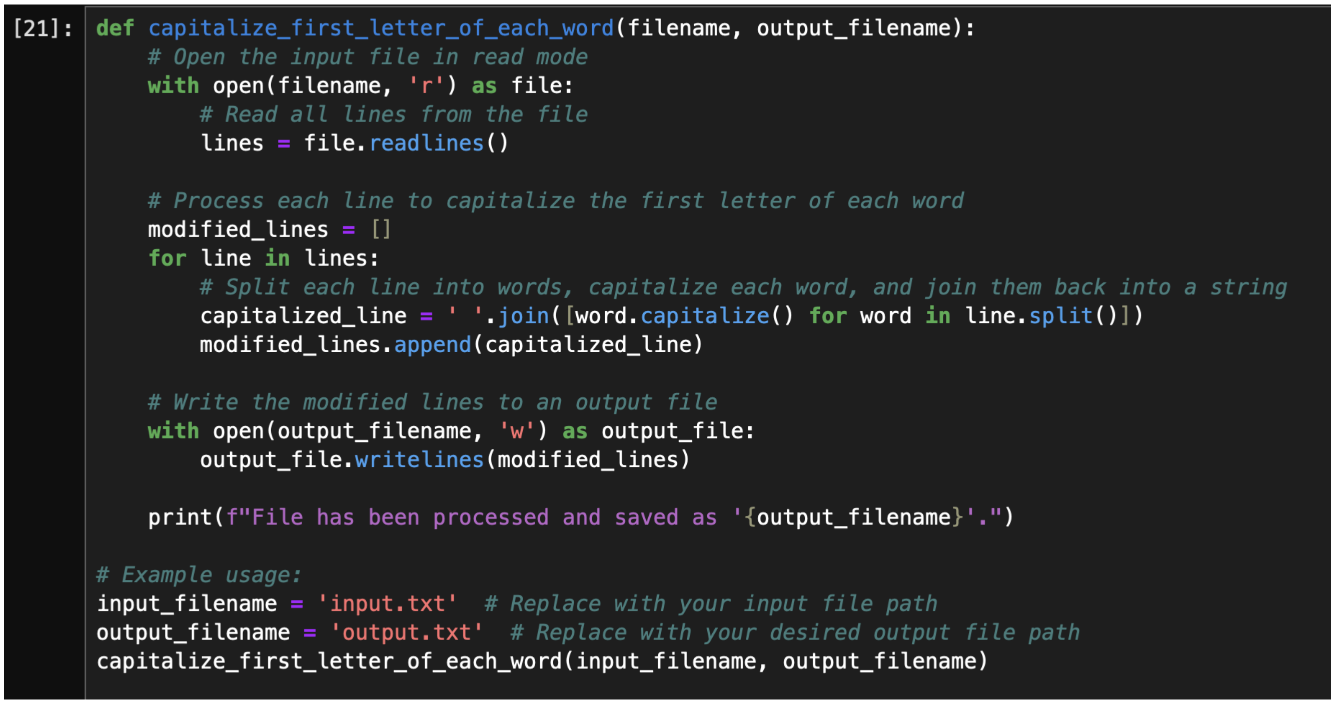
Output:



```
1 : 5
5 : 2
3 : 3
4 : 3
2 : 4
```

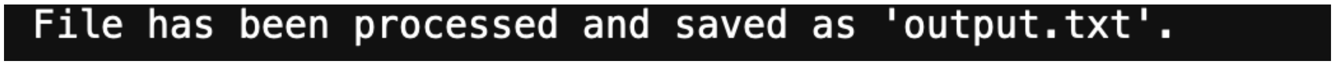
Program 12(a): Write a Python Program to Capitalize First Letter of Each Word in a File.

Solution:



```
[21]: def capitalize_first_letter_of_each_word(filename, output_filename):
          # Open the input file in read mode
          with open(filename, 'r') as file:
              # Read all lines from the file
              lines = file.readlines()

          # Process each line to capitalize the first letter of each word
          modified_lines = []
          for line in lines:
              # Split each line into words, capitalize each word, and join them back into a string
              capitalized_line = ' '.join([word.capitalize() for word in line.split()])
              modified_lines.append(capitalized_line)

          # Write the modified lines to an output file
          with open(output_filename, 'w') as output_file:
              output_file.writelines(modified_lines)

          print(f"File has been processed and saved as '{output_filename}'.")

      # Example usage:
      input_filename = 'input.txt'  # Replace with your input file path
      output_filename = 'output.txt'  # Replace with your desired output file path
      capitalize_first_letter_of_each_word(input_filename, output_filename)
```

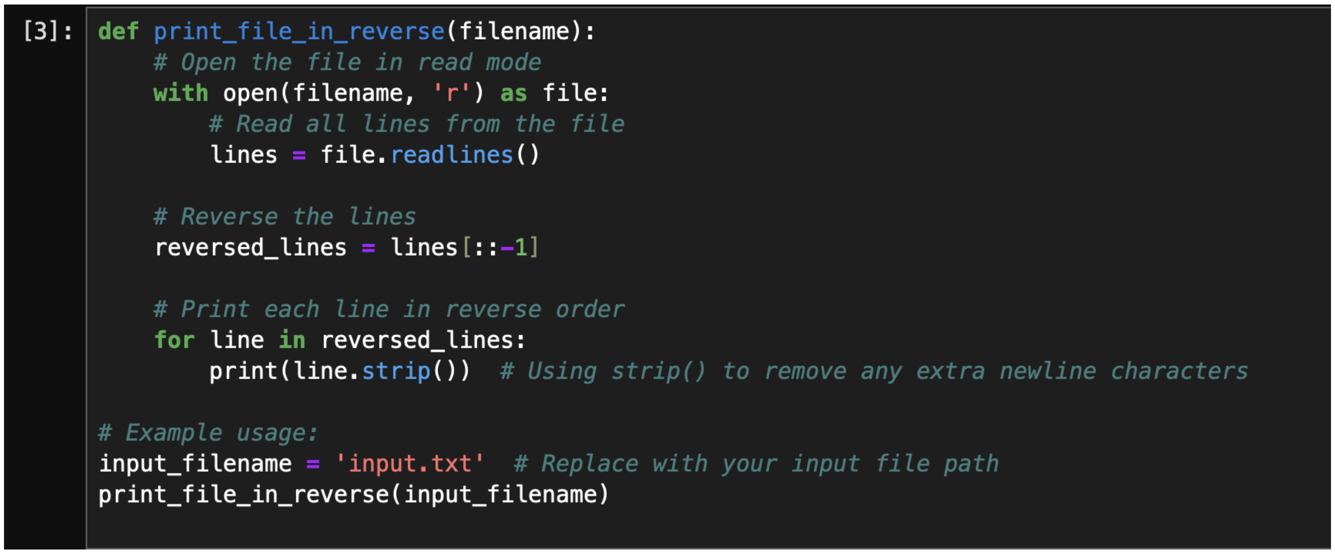
Output:



```
File has been processed and saved as 'output.txt'.
```

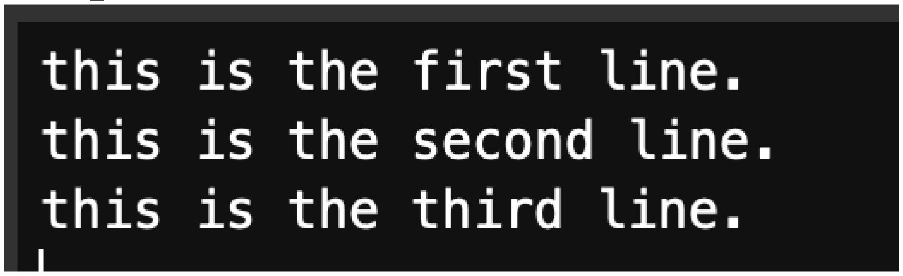
Program 12(b): Write a Python Program to Print the Contents of File in Reverse Order.

Solution:

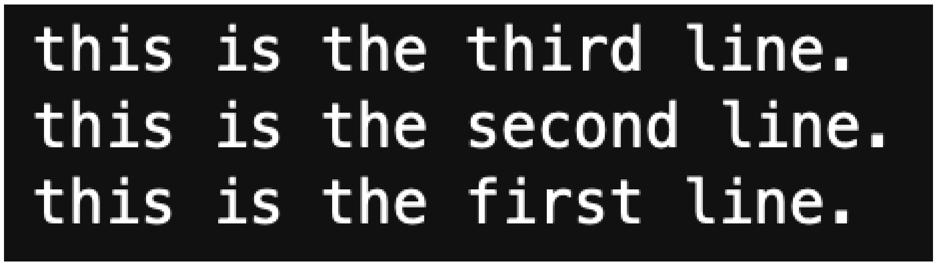


```
[3]: def print_file_in_reverse(filename):
         # Open the file in read mode
         with open(filename, 'r') as file:
             # Read all lines from the file
             lines = file.readlines()

         # Reverse the lines
         reversed_lines = lines[::-1]

         # Print each line in reverse order
         for line in reversed_lines:
             print(line.strip())  # Using strip() to remove any extra newline characters

     # Example usage:
     input_filename = 'input.txt'  # Replace with your input file path
     print_file_in_reverse(input_filename)
```

Output:



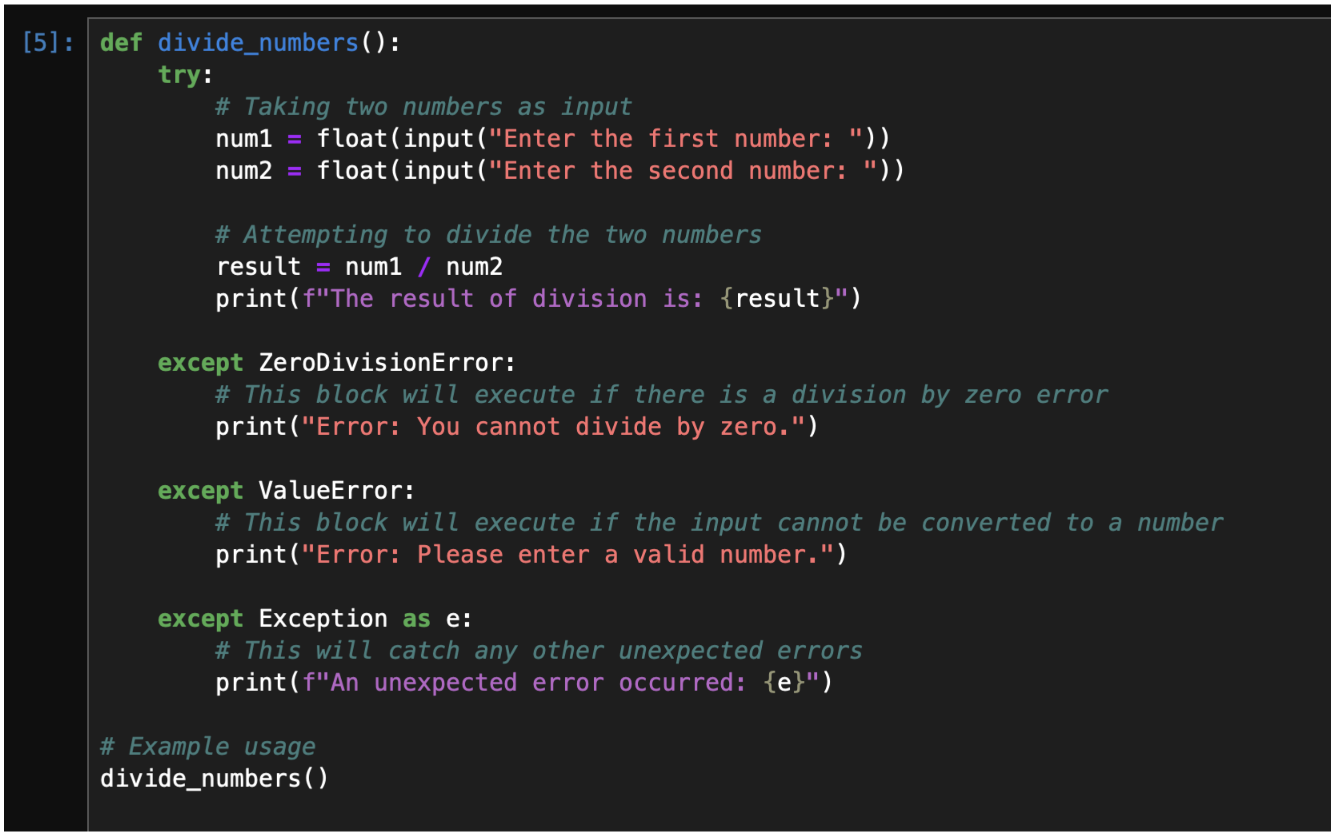
```
this is the first line.
this is the second line.
this is the third line.
```



```
this is the third line.
this is the second line.
this is the first line.
```

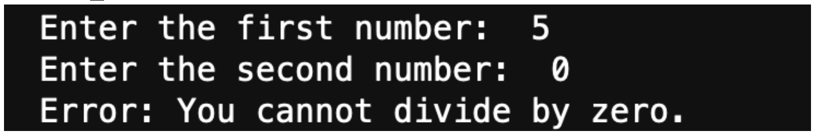
Program 13: WAP to catch an exception and handle it using try and except code blocks

Solution:

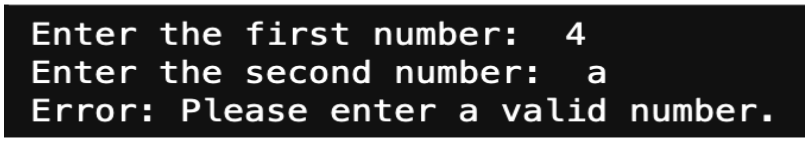


```
[5]: def divide_numbers():
         try:
             # Taking two numbers as input
             num1 = float(input("Enter the first number: "))
             num2 = float(input("Enter the second number: "))

             # Attempting to divide the two numbers
             result = num1 / num2
             print(f"The result of division is: {result}")

         except ZeroDivisionError:
             # This block will execute if there is a division by zero error
             print("Error: You cannot divide by zero.")

         except ValueError:
             # This block will execute if the input cannot be converted to a number
             print("Error: Please enter a valid number.")

         except Exception as e:
             # This will catch any other unexpected errors
             print(f"An unexpected error occurred: {e}")

     # Example usage
     divide_numbers()
```

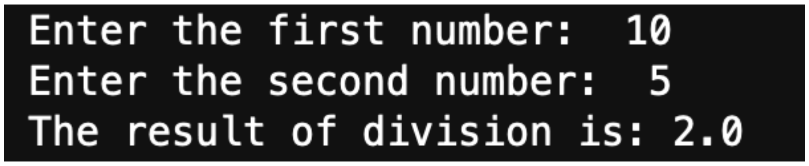
Output:



```
Enter the first number:  5
Enter the second number:  0
Error: You cannot divide by zero.
```



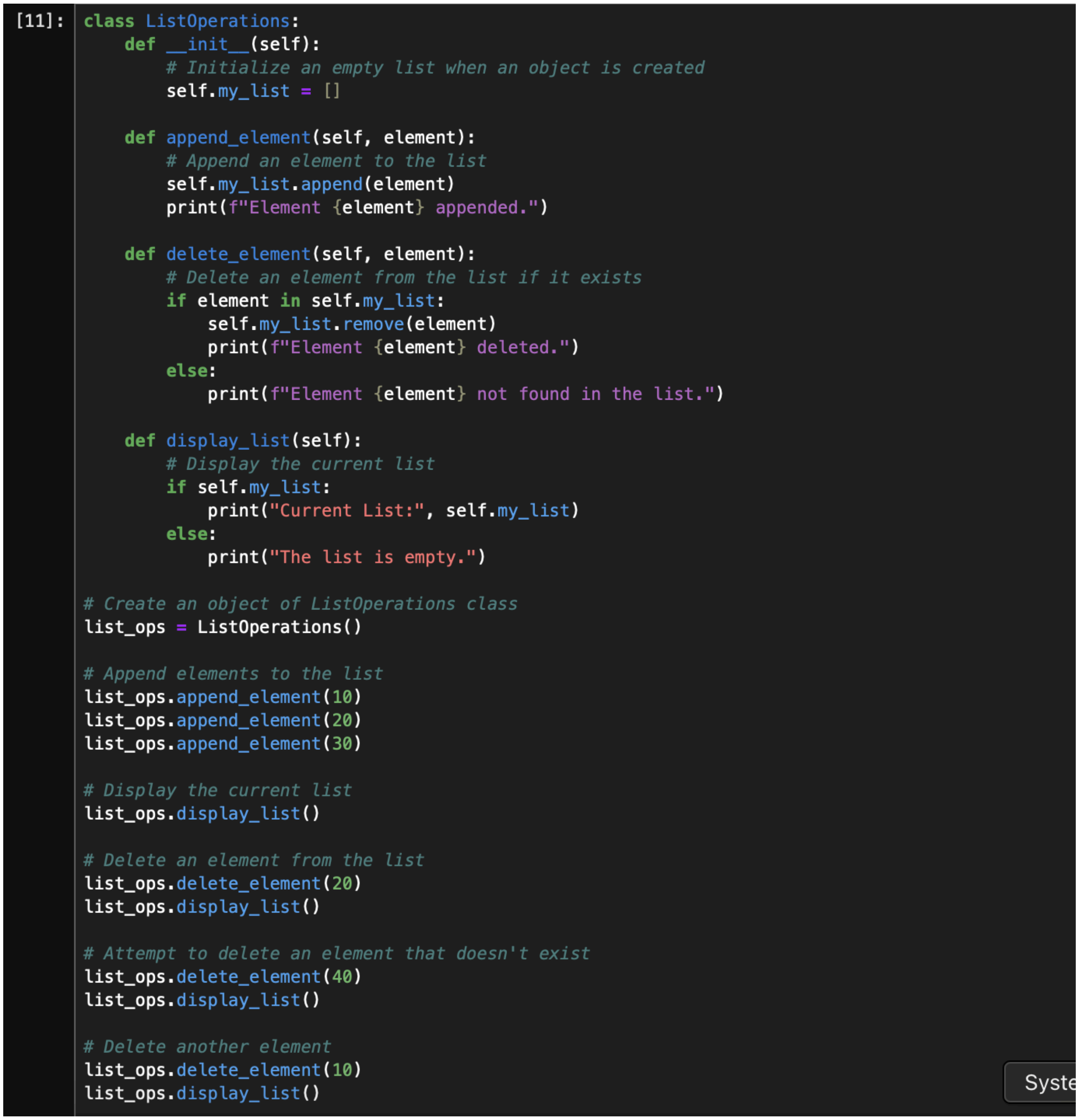
```
Enter the first number:  4
Enter the second number:  a
Error: Please enter a valid number.
```



```
Enter the first number:  10
Enter the second number:  5
The result of division is: 2.0
```

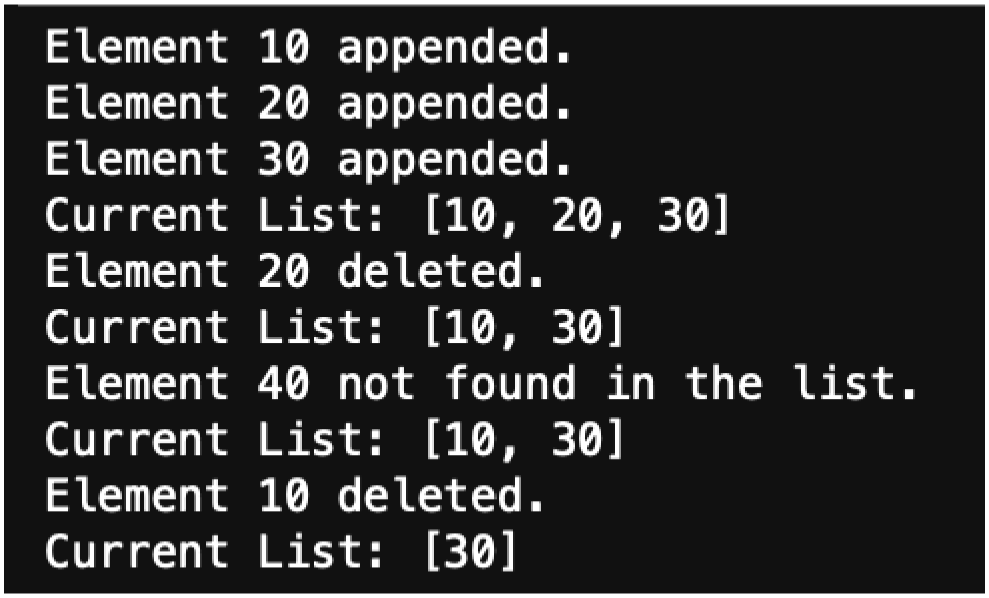
Program 14: Write a Python Program to Append, Delete and Display Elements of a List using Classes.

Solution:



```
[11]: class ListOperations:
          def __init__(self):
              # Initialize an empty list when an object is created
              self.my_list = []

          def append_element(self, element):
              # Append an element to the list
              self.my_list.append(element)
              print(f"Element {element} appended.")

          def delete_element(self, element):
              # Delete an element from the list if it exists
              if element in self.my_list:
                  self.my_list.remove(element)
                  print(f"Element {element} deleted.")
              else:
                  print(f"Element {element} not found in the list.")

          def display_list(self):
              # Display the current list
              if self.my_list:
                  print("Current List:", self.my_list)
              else:
                  print("The list is empty.")

      # Create an object of ListOperations class
      list_ops = ListOperations()

      # Append elements to the list
      list_ops.append_element(10)
      list_ops.append_element(20)
      list_ops.append_element(30)

      # Display the current list
      list_ops.display_list()

      # Delete an element from the list
      list_ops.delete_element(20)
      list_ops.display_list()

      # Attempt to delete an element that doesn't exist
      list_ops.delete_element(40)
      list_ops.display_list()

      # Delete another element
      list_ops.delete_element(10)
      list_ops.display_list()
```

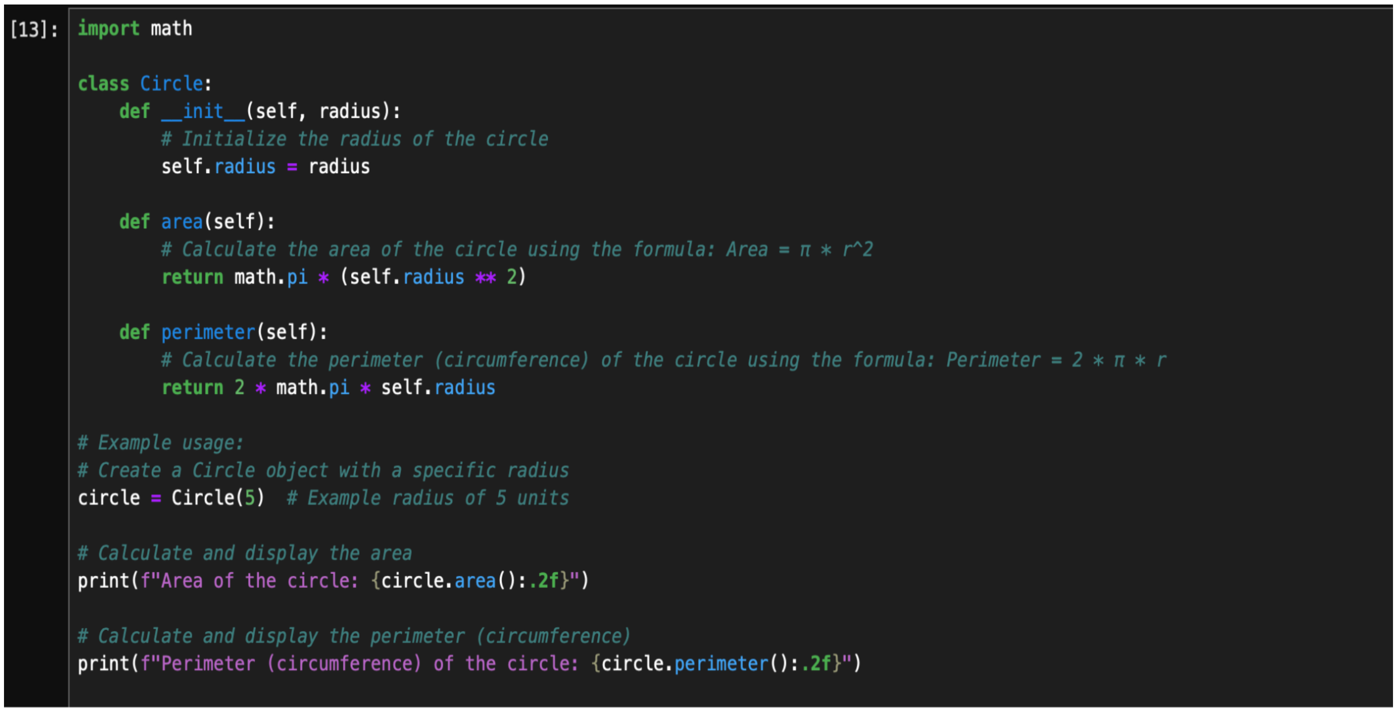
Output:



```
Element 10 appended.
Element 20 appended.
Element 30 appended.
Current List: [10, 20, 30]
Element 20 deleted.
Current List: [10, 30]
Element 40 not found in the list.
Current List: [10, 30]
Element 10 deleted.
Current List: [30]
```

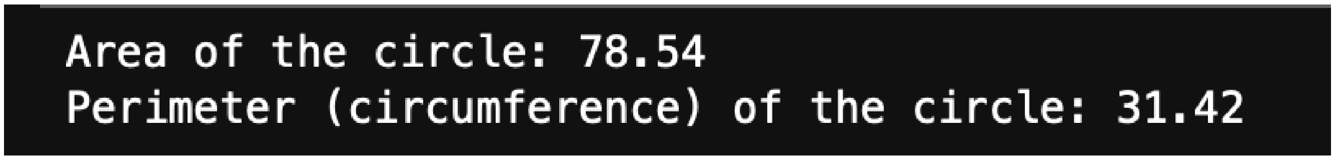
Program 15: Write a Python Program to Find the Area and Perimeter of the Circle using Class.

Solution:



```
[13]: import math

class Circle:
    def __init__(self, radius):
        # Initialize the radius of the circle
        self.radius = radius

    def area(self):
        # Calculate the area of the circle using the formula: Area = π * r^2
        return math.pi * (self.radius ** 2)

    def perimeter(self):
        # Calculate the perimeter (circumference) of the circle using the formula: Perimeter = 2 * π * r
        return 2 * math.pi * self.radius

# Example usage:
# Create a Circle object with a specific radius
circle = Circle(5)  # Example radius of 5 units

# Calculate and display the area
print(f"Area of the circle: {circle.area():.2f}")

# Calculate and display the perimeter (circumference)
print(f"Perimeter (circumference) of the circle: {circle.perimeter():.2f}")
```

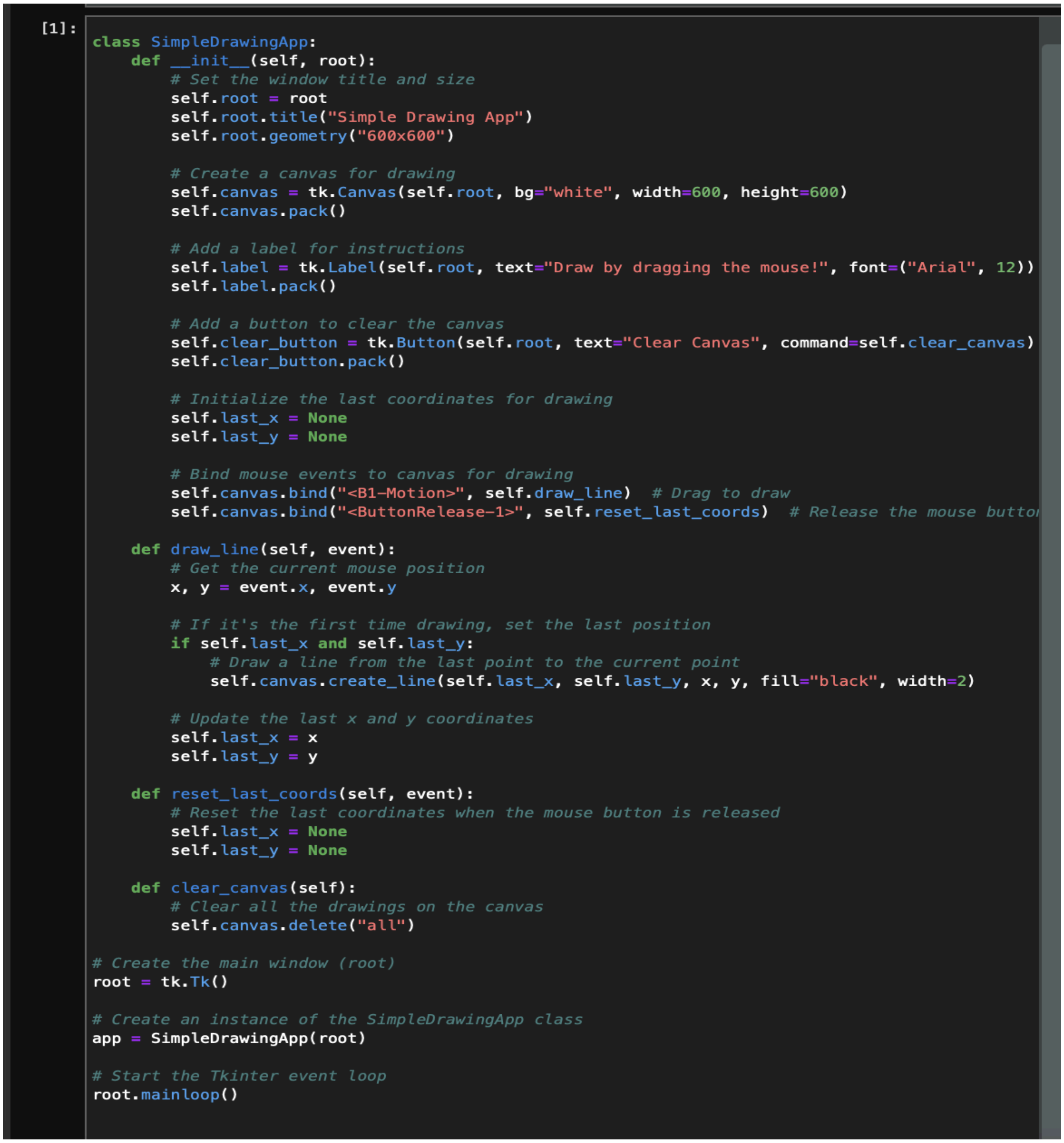
Output:



```
Area of the circle: 78.54
Perimeter (circumference) of the circle: 31.42
```

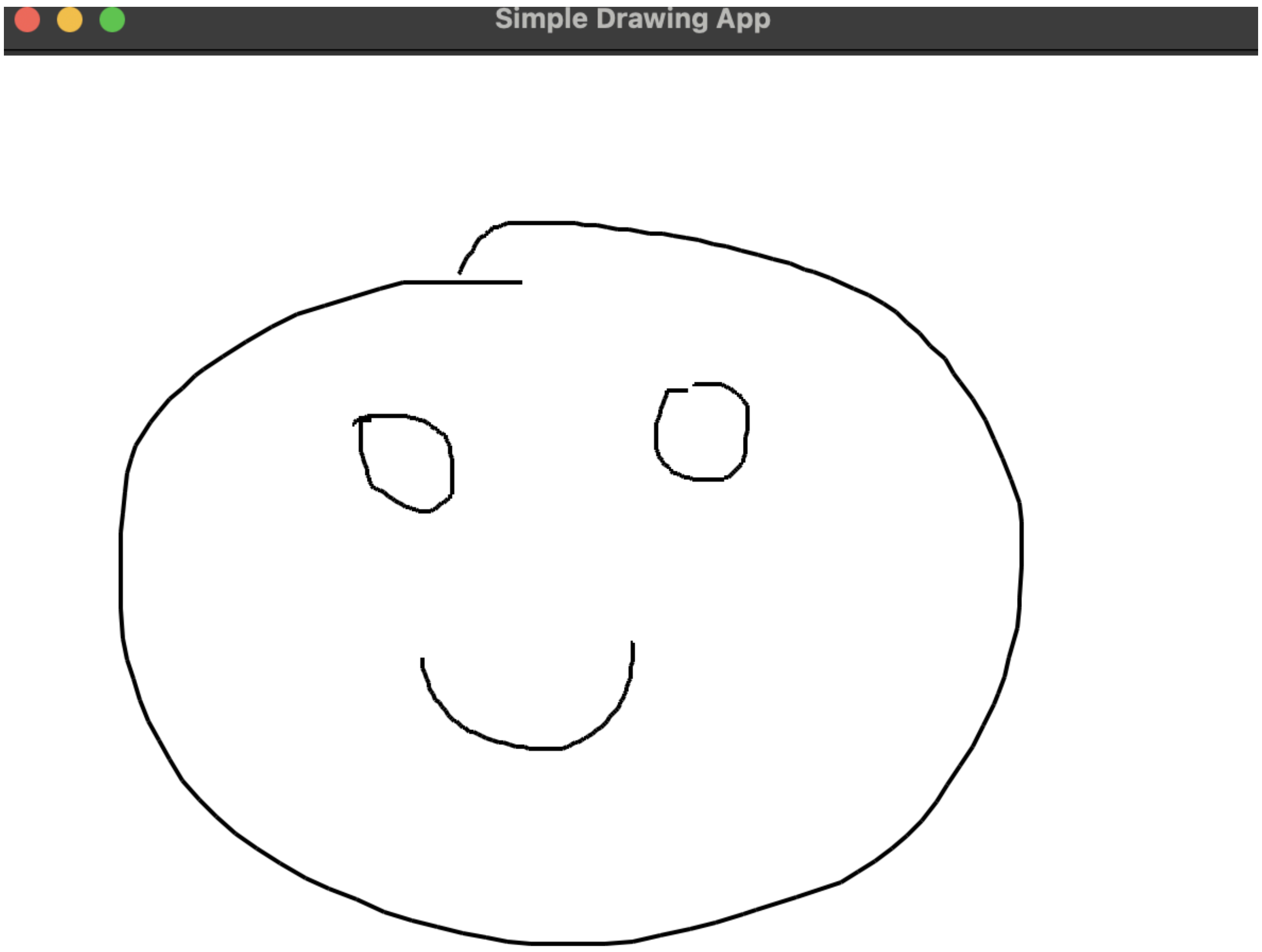
Program 16: Create an interactive application using Python's Tkinter library for graphics programming.

Solution:



```
[1]:
class SimpleDrawingApp:
    def __init__(self, root):
        # Set the window title and size
        self.root = root
        self.root.title("Simple Drawing App")
        self.root.geometry("600x600")

        # Create a canvas for drawing
        self.canvas = tk.Canvas(self.root, bg="white", width=600, height=600)
        self.canvas.pack()

        # Add a label for instructions
        self.label = tk.Label(self.root, text="Draw by dragging the mouse!", font=("Arial", 12))
        self.label.pack()

        # Add a button to clear the canvas
        self.clear_button = tk.Button(self.root, text="Clear Canvas", command=self.clear_canvas)
        self.clear_button.pack()

        # Initialize the last coordinates for drawing
        self.last_x = None
        self.last_y = None

        # Bind mouse events to canvas for drawing
        self.canvas.bind("<B1-Motion>", self.draw_line)  # Drag to draw
        self.canvas.bind("<ButtonRelease-1>", self.reset_last_coords)  # Release the mouse butto

    def draw_line(self, event):
        # Get the current mouse position
        x, y = event.x, event.y

        # If it's the first time drawing, set the last position
        if self.last_x and self.last_y:
            # Draw a line from the last point to the current point
            self.canvas.create_line(self.last_x, self.last_y, x, y, fill="black", width=2)

        # Update the last x and y coordinates
        self.last_x = x
        self.last_y = y

    def reset_last_coords(self, event):
        # Reset the last coordinates when the mouse button is released
        self.last_x = None
        self.last_y = None

    def clear_canvas(self):
        # Clear all the drawings on the canvas
        self.canvas.delete("all")

# Create the main window (root)
root = tk.Tk()

# Create an instance of the SimpleDrawingApp class
app = SimpleDrawingApp(root)

# Start the Tkinter event loop
root.mainloop()
```

Output:


Simple Drawing App