K.Mythri

BL.EN.U4AIE23113

AIE-E

Q1.

def count\_pairs\_with\_sum\_10(numbers):

count = 0 # Initialize the count of pairs

n = len(numbers) # Get the length of the list

# Iterate through the list to find pairs

for i in range(0, n):

for j in range(i + 1, n):

if numbers[i] + numbers[j] == 10:

count += 1

return count # Return the count of pairs

# Main program

if \_\_name\_\_ == "\_\_main\_\_":

numbers\_list = [2, 7, 4, 1, 3, 6]

pair\_count = count\_pairs\_with\_sum\_10(numbers\_list)

print("Number of pairs with sum 10:", pair\_count)

Q2.

def calculate\_range(numbers):

if len(numbers) < 3:

return "Error: Range determination is not possible" # Return error message for lists with fewer than 3 elements

else:

min\_value = min(numbers) # Find the minimum value in the list

max\_value = max(numbers) # Find the maximum value in the list

range\_value = max\_value - min\_value # Calculate the range

return range\_value # Return the range

# Main program

if \_\_name\_\_ == "\_\_main\_\_":

numbers\_list = [5, 3, 8, 1, 0, 4] # Sample list

range\_result = calculate\_range(numbers\_list) # Calculate the range

print("Range of the list:", range\_result) # Print the result

Q3.

import numpy as np

def matrix\_power(A, m):

if m <= 0:

return "Error: Exponent must be a positive integer."

else:

result = np.linalg.matrix\_power(A, m) # Calculate matrix power using NumPy

return result # Return the result

# Main program

if \_\_name\_\_ == "\_\_main\_\_":

A = np.array([[1, 2], [3, 4]]) # Example square matrix

m = 3 # Example positive integer exponent

result\_matrix = matrix\_power(A, m) # Calculate A^m

if isinstance(result\_matrix, str): # Check if result is an error message

print(result\_matrix)

else:

print("Matrix A raised to the power of", m, "is:") # Print the result

print(result\_matrix)

Q4.

import collections

def highest\_occurring\_alphabet(text):

# Filter out non-alphabetic characters and convert to lowercase

filtered\_text = ''.join(ch for ch in text if ch.isalpha()).lower()

# Create a Counter object to count alphabet frequencies

alphabet\_counts = collections.Counter(filtered\_text)

# Find the alphabet with the highest frequency

highest\_alphabet = max(alphabet\_counts, key=alphabet\_counts.get)

occurrence\_count = alphabet\_counts[highest\_alphabet]

return highest\_alphabet, occurrence\_count

# Main program

if \_\_name\_\_ == "\_\_main\_\_":

input\_string = "hippopotamus"

highest\_alphabet, occurrence\_count = highest\_occurring\_alphabet(input\_string)

print("Highest occurring alphabet:", highest\_alphabet)

print("Occurrence count:", occurrence\_count)