def count\_pairs\_with\_sum(numbers, target\_sum):

count = 0

n = len(numbers)

for i in range(n):

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

if numbers[i] + numbers[j] == target\_sum:

count += 1

return count

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

target\_sum = 10

result = count\_pairs\_with\_sum(numbers, target\_sum)

print(f"Number of pairs with sum {target\_sum}: {result}")

def calculate\_range(numbers):

if len(numbers) < 3:

return "Range determination not possible"

max\_value = max(numbers)

min\_value = min(numbers)

range\_value = max\_value - min\_value

return range\_value

input\_list = [5, 3, 8, 1, 0, 4]

result = calculate\_range(input\_list)

print(f"The range of the list {input\_list} is: {result}")

def count\_highest\_occurring\_char(input\_string):

input\_string = input\_string.lower()

char\_count = {}

for char in input\_string:

if char.isalpha():

char\_count[char] = char\_count.get(char, 0) + 1

max\_count = max(char\_count.values())

highest\_char = next(char for char, count in char\_count.items() if count == max\_count)

return highest\_char, max\_count

input\_string = "hippopotamus"

highest\_char, occurrence\_count = count\_highest\_occurring\_char(input\_string)

print(f"The highest occurring character in '{input\_string}' is '{highest\_char}' with an occurrence count of {occurrence\_count}.")

import numpy as np

def matrix\_power(A, m):

if A.shape[0] != A.shape[1]:

return "Error: Input must be a square matrix."

if not isinstance(m, int) or m <= 0:

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

result = np.eye(A.shape[0])

for \_ in range(m):

result = np.dot(result, A)

return result

A = np.array([[1, 2], [3, 4]])

m = 3

result = matrix\_power(A, m)

print(result)