Day 4 Python Programming

1)Perfect number

def is\_perfect\_number(num):

if num <= 1:

return False

sum\_divisors = 0

for i in range(1, num // 2 + 1):

if num % i == 0:

sum\_divisors += i

return sum\_divisors == num

number = 28

if is\_perfect\_number(number):

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

else:

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

2)Transpose of a Matrix

def transpose\_matrix(matrix):

# Use list comprehension to transpose the matrix

return [list(row) for row in zip(\*matrix)]

matrix = [

[1, 2, 3],

[4, 5, 6],

[7, 8, 9]

]

transposed = transpose\_matrix(matrix)

print("Original Matrix:")

for row in matrix:

print(row)

print("\nTransposed Matrix:")

for row in transposed:

print(row)

3)Sum of row,column and diagonal of the given matrix

def sum\_rows(matrix):

return [sum(row) for row in matrix]

def sum\_columns(matrix):

return [sum(matrix[row][col] for row in range(len(matrix))) for col in range(len(matrix[0]))]

def sum\_diagonals(matrix):

primary\_diagonal = sum(matrix[i][i] for i in range(len(matrix)))

secondary\_diagonal = sum(matrix[i][len(matrix) - 1 - i] for i in range(len(matrix)))

return primary\_diagonal, secondary\_diagonal

matrix = [

[1, 2, 3],

[4, 5, 6],

[7, 8, 9]

]

rows\_sum = sum\_rows(matrix)

columns\_sum = sum\_columns(matrix)

primary\_diagonal\_sum, secondary\_diagonal\_sum = sum\_diagonals(matrix)

print("Matrix:")

for row in matrix:

print(row)

print("\nSum of each row:")

print(rows\_sum)

print("\nSum of each column:")

print(columns\_sum)

print("\nSum of primary diagonal:")

print(primary\_diagonal\_sum)

print("\nSum of secondary diagonal:")

print(secondary\_diagonal\_sum)

4)Sum of Boundary elements of a matrix

def sum\_boundary\_elements(matrix):

if not matrix or not matrix[0]:

return 0

rows = len(matrix)

cols = len(matrix[0])

top\_row\_sum = sum(matrix[0])

bottom\_row\_sum = sum(matrix[-1])

left\_column\_sum = sum(matrix[i][0] for i in range(1, rows - 1))

right\_column\_sum = sum(matrix[i][-1] for i in range(1, rows - 1))

total\_sum = top\_row\_sum + bottom\_row\_sum + left\_column\_sum + right\_column\_sum

return total\_sum

matrix = [

[1, 2, 3, 4],

[5, 6, 7, 8],

[9, 10, 11, 12],

[13, 14, 15, 16]

]

boundary\_sum = sum\_boundary\_elements(matrix)

print("Matrix:")

for row in matrix:

print(row)

print("\nSum of boundary elements:")

print(boundary\_sum)

5)Matrix in Spiral order

def spiral\_order(matrix):

if not matrix:

return []

result = []

top, bottom, left, right = 0, len(matrix) - 1, 0, len(matrix[0]) - 1

while top <= bottom and left <= right:

for col in range(left, right + 1):

result.append(matrix[top][col])

top += 1

for row in range(top, bottom + 1):

result.append(matrix[row][right])

right -= 1

if top <= bottom:

# Traverse from right to left along the bottom row

for col in range(right, left - 1, -1):

result.append(matrix[bottom][col])

bottom -= 1

if left <= right:

# Traverse from bottom to top along the left column

for row in range(bottom, top - 1, -1):

result.append(matrix[row][left])

left += 1

return result

matrix = [

[1, 2, 3, 4],

[5, 6, 7, 8],

[9, 10, 11, 12],

[13, 14, 15, 16]

]

spiral = spiral\_order(matrix)

print("Matrix:")

for row in matrix:

print(row)

print("\nSpiral order:")

print(spiral)

6) Sum of n numbers

def sum\_of\_n\_numbers\_formula(n):

return n \* (n + 1) // 2

n = 10

result = sum\_of\_n\_numbers\_formula(n)

print(f"Sum of the first {n} numbers using the formula is {result}.")

7) Sum of square of n numbers

def sum\_of\_squares\_loop(n):

total = 0

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

total += i \*\* 2

return total

n = 5

result = sum\_of\_squares\_loop(n)

print(f"Sum of squares of the first {n} numbers using a loop is {result}.")

8)Sum of n factorial

import math

def sum\_of\_factorials\_loop(n):

total = 0

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

total += math.factorial(i)

return total

n = 5

result = sum\_of\_factorials\_loop(n)

print(f"Sum of factorials of the first {n} numbers using a loop is {result}.")

9) Mean,median and mode of elements in the list

def calculate\_mean(data):

return sum(data) / len(data)

def calculate\_median(data):

sorted\_data = sorted(data)

n = len(sorted\_data)

mid = n // 2

if n % 2 == 0:

return (sorted\_data[mid - 1] + sorted\_data[mid]) / 2

else:

return sorted\_data[mid]

def calculate\_mode(data):

from collections import Counter

count = Counter(data)

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

modes = [key for key, value in count.items() if value == max\_count]

if len(modes) == len(data):

return "No unique mode" # All elements are equally frequent

return modes[0] # Return the first mode

data = [1, 2, 2, 3, 4, 5, 5, 5, 6]

mean\_value = calculate\_mean(data)

median\_value = calculate\_median(data)

mode\_value = calculate\_mode(data)

print(f"Data: {data}")

print(f"Mean: {mean\_value}")

print(f"Median: {median\_value}")

print(f"Mode: {mode\_value}")

10)Find the largest number in the list

def find\_largest\_number(numbers):

if not numbers:

return None # Return None if the list is empty

largest = numbers[0]

for num in numbers[1:]:

if num > largest:

largest = num

return largest

numbers = [3, 5, 7, 2, 8, 6, 4]

largest\_number = find\_largest\_number(numbers)

print(f"The largest number in the list is {largest\_number}.")