Name - Soumyadip Adhikari

Roll number - 2107021

Subject - AI ML Assignment 1.13

1)def count\_vowels(string):

vowels = 'aeiou'

vowel\_count = {}

# Initializing the vowel count dictionary

for vowel in vowels:

vowel\_count[vowel] = 0

# Counting the vowels in the string

for char in string.lower():

if char in vowels:

vowel\_count[char] += 1

return vowel\_count

input\_string = input("Enter a string: ")

vowel\_counts = count\_vowels(input\_string)

print("Number of each vowel in the string:")

for vowel, count in vowel\_counts.items():

print(vowel, ":", count)

2)def count\_statistics(string):

word\_count = len(string.split())

char\_count = len(string)

space\_count = string.count(' ')

return word\_count, char\_count, space\_count

input\_string = input("Enter a string: ")

word\_count, char\_count, space\_count = count\_statistics(input\_string)

print("Number of words:", word\_count)

print("Number of characters:", char\_count)

print("Number of spaces:", space\_count)

3)def remove\_punctuation(string):

punctuations = '''!"#$%&'()\*+,-./:;<=>?@[\]^\_`{|}~'''

new\_string = ""

for char in string:

if char not in punctuations:

new\_string += char

return new\_string

input\_string = input("Enter a string: ")

result = remove\_punctuation(input\_string)

print("String after removing punctuation:", result)

4)def create\_matrix(m, n):

matrix = []

for i in range(m):

row = []

for j in range(n):

element = int(input("Enter element at position ({}, {}): ".format(i, j)))

row.append(element)

matrix.append(row)

return matrix

# Get matrix dimensions from the user

m = int(input("Enter the number of rows (m): "))

n = int(input("Enter the number of columns (n): "))

# Create the matrix

matrix = create\_matrix(m, n)

# Print the matrix

print("Matrix:")

for row in matrix:

print(row)

5)def create\_matrix(m, n):

matrix = []

for i in range(m):

row = []

for j in range(n):

element = int(input("Enter element at position ({}, {}): ".format(i, j)))

row.append(element)

matrix.append(row)

return matrix

def add\_matrices(matrix1, matrix2):

result = []

for i in range(len(matrix1)):

row = []

for j in range(len(matrix1[0])):

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

row.append(element)

result.append(row)

return result

def multiply\_matrices(matrix1, matrix2):

result = []

for i in range(len(matrix1)):

row = []

for j in range(len(matrix2[0])):

element = 0

for k in range(len(matrix2)):

element += matrix1[i][k] \* matrix2[k][j]

row.append(element)

result.append(row)

return result

def transpose\_matrix(matrix):

result = []

for j in range(len(matrix[0])):

row = []

for i in range(len(matrix)):

element = matrix[i][j]

row.append(element)

result.append(row)

return result

# Get matrix dimensions from the user

m = int(input("Enter the number of rows: "))

n = int(input("Enter the number of columns: "))

print("Enter elements of Matrix 1:")

matrix1 = create\_matrix(m, n)

print("Enter elements of Matrix 2:")

matrix2 = create\_matrix(m, n)

# Addition of matrices

addition\_result = add\_matrices(matrix1, matrix2)

print("Addition of the two matrices:")

for row in addition\_result:

print(row)

# Multiplication of matrices

if len(matrix1[0]) == len(matrix2):

multiplication\_result = multiply\_matrices(matrix1, matrix2)

print("Multiplication of the two matrices:")

for row in multiplication\_result:

print(row)

else:

print("Cannot multiply the matrices. The number of columns in Matrix 1 must be equal to the number of rows in Matrix 2.")

# Transpose of matrices

transpose\_result1 = transpose\_matrix(matrix1)

transpose\_result2 = transpose\_matrix(matrix2)

print("Transpose of Matrix 1:")

for row in transpose\_result1:

print(row)

print("Transpose of Matrix 2:")

for row in transpose\_result2:

print(row)