MACHINE LEARNING

22AIE213

Assignment -1- Report

Team Members -

Amara Gnana Sirishma – AIE22105

Yoshitha Tulasi- AIE22177

Meka Sai Sri Hanish- AIE22130

AIE22105

Set A

Question-1

Pseudo code-

```
function count_pairs_with_sum(lst, target_sum):
    pairs_count = 0
    seen_numbers = set()

    for num in lst:
        complement = target_sum - num
        if complement in seen_numbers:
            pairs_count += 1
        seen_numbers.add(num)

    return pairs_count

function main():
    list_given = [2, 7, 4, 1, 3, 6]
    target_sum = 10
    pairs_count = count_pairs_with_sum(list_given, target_sum)
    print "The count of pairs with sum", target_sum, "in the list is:", pairs_count

if __name__ == "__main__":
    main()
```

Explanation-

This code counts the number of pairs adding up to 10, by calculating the complement of each number. The complement is calculated by subtracting each number from the list from 10.

Then it checks if the complement number is present in the list. If the complement number is present in the list, then it increments the count of the number of pairs.

Question-2

Pseudo code-

```
function calculate_range(real_numbers):
    if len(real_numbers) < 3:
        return "Range determination not possible."
    return max(real_numbers) - min(real_numbers)


function main():
    real_numbers_list = [5, 3, 8, 1, 0, 4]
    minimum_no = min(real_numbers_list)
    maximum_no = max(real_numbers_list)
    range_a = calculate_range(real_numbers_list)
    print("Range of the list:", range_a, "(", maximum_no, "-", minimum_no, ")")


if __name__ == "__main__":
    main()
```

Explanation-

This code calculates the difference between the maximum and minimum value present in a given list. First it checks if the number of elements is less than three in the given list, as told in the problem, and if the condition is true, the error message is displayed. If the condition is not true, then by using the python inbuild functions maximum and minimum values from the list are calculated and the range is determined, by subtracting minimum value from the maximum value.

Question-3

Pseudo code-

```
function matrix_power(matrix, power):
    result = matrix
    for _ in range(power - 1):
        result = [
            [
                sum(a * b for a, b in zip(row, col))
                for col in zip(*matrix)
            ]
            for row in result
        ]
    return result
function main():
    size_of_matrix = int(input("Enter the size of the square matrix "))
    matrixA = []
    for i in range(size_of_matrix):
        row = [int(ele) for ele in input(f"Enter values for row {i + 1} (give space in between the numbers): ").split()]
        matrixA.append(row)
    m = int(input("Enter the positive integer m (power to raise the matrix): "))
    result_matrixA = matrix_power(matrixA, m)
    print(f"Resultant matrix after raising to the power {m}:\n{result_matrixA}")


if __name__ == "__main__":
    main()
```

Explanation-

This code returns a matrix raised to the power of the number m provided by the user. It runs the loop from 0 to power(m)-1. In each iterating the matrix is multiplied by itself. It goes through each row of the result matrix and, for each row, goes through each column in the transposed original matrix. It calculates the sum of products of corresponding elements from

the current row and column, creating a new matrix as a result. The code uses list comprehensions and the zip function to perform this matrix multiplication in a compact way.

Question-4

Pseudo code-

```
function highest_occurrence_char(input_string):
    char_count = {}
    for char in input_string:
        if char.isalpha():
            char_count[char] = char_count.get(char, 0) + 1
    if not char_count:
        return None, 0
    max_char = max(char_count, key=char_count.get)
    max_count = char_count[max_char]
    return max_char, max_count


function main():
    input_string = "hippopotamus"
    maximum_occ_char, max_count = highest_occurrence_char(input_string)
    print("Maximum occurring character: '", maximum_occ_char, "', Occurrence count: ", max_count)


if __name__ == "__main__":
    main()
```

Explanation-

This code shows which character is repeated most of the time in the given string and the number of times it is repeated. We iterate through each character of the string, and if the character is alphabet, its count is updated to the dictionary. If there are no alphabets present in the string then it returns none. Otherwise, it identifies the alphabet with the maximum occurrence using the max function, and returns a tuple containing this character and its count.

AIE22177

Set A

Question-1

Pseudo code-

```
function count_pairs_with_sum(lst, target_sum):
    count = 0
    for i in range(length(lst)):
        for j in range(i + 1, length(lst)):
            if lst[i] + lst[j] == target_sum:
                count += 1
    return count


given_list = [2, 7, 4, 1, 3, 6]
result = count_pairs_with_sum(given_list, 10)
print("Number of pairs of elements with sum equal to 10: " + result)
```

Explanation-

This code defines a function that counts pairs of numbers in a list whose sum matches a given target. It iterates through the list, checking each pair for the specified sum and increments a counter if a match is found. The result is the total count of such pairs.

Question-2

Pseudo code-

```
function calculate_range(numbers):
    if length(numbers) < 3:
        return "Range determination not possible"
    else:
        return max(numbers) - min(numbers)


real_numbers_list = [5, 3, 8, 1, 0, 4]
result = calculate_range(real_numbers_list)
print("Range of the list: " + result)
```

Explanation-

In this code, a function calculates the range of a list of numbers. It first checks if there are at least three numbers in the list and returns a message if not. If there are, it calculates the range by finding the difference between the maximum and minimum values.

Question-3

Pseudo code-

```
function multiply_matrices(A, B):
    result = []
    for i in range(length(A)):
        row = []
        for j in range(length(B[0])):
            element = 0
            for k in range(length(B)):
                element = element + A[i][k] * B[k][j]
            row.append(element)
        result.append(row)
    return result


function matrix_power(A, m):
    result = copy(A)
    for _ in range(m - 1):
        result = multiply_matrices(result, A)
    return result
n = input("Enter the size of the square matrix A: ")
A = []
for i in range(n):
    row = []
    for j in range(n):
        element = input(f"Enter element at position ({i + 1}, {j + 1}): ")
        row.append(element)
    A.append(row)
m = input("Enter a positive integer m: ")
result = matrix_power(A, m)
print("A^" + m + ":\n" + result)
```

Explanation-

This script contains functions to multiply two matrices and raise a matrix to a given power. It uses nested loops for matrix multiplication and iterates through the power to perform repeated matrix multiplication, demonstrating basic matrix operations.

Question-4

Pseudo code-

```
function max_occurrence_count(input_string):
    counts = {}
    for char in input_string:
        if is_alpha(char):
            counts[char] = counts.get(char, 0) + 1

    if not counts:
        return "No alphabets found in the input string"

    max_char = max_key(counts)
    max_count = counts[max_char]

    return max_char, max_count

input_str = "hippopotamus"
result = max_occurrence_count(input_str)
print("The maximally occurring character is '" + result[0] + "' with occurrence count " + result[1] + ".")
```

Explanation-

This code defines a function that determines the character with the maximum occurrence in a given string. It counts the occurrences of each alphabet, identifies the one with the highest count, and returns it along with the count. It showcases this function using the input string "hippopotamus" and prints the maximally occurring character and its count.

AIE22130

Set B

Question-1

Pseudo code-

```
function count_vowels_and_consonants(input_string):
    vowels_List = "AaEeIiOoUu"
    vowel_count = 0
    consonant_count = 0

    for char in input_string:
        if char.is_alpha():
            if char in vowels_List:
                vowel_count += 1
            else:
                consonant_count += 1

    return vowel_count, consonant_count

input_str = input("Enter a string: ")
vowel_count, consonant_count = count_vowels_and_consonants(input_str)
print("Number of Vowels= " + vowel_count + ", Number of consonants= " + consonant_count)
```

Explanation-

The code defines a function, count_vowels_and_consonants, which takes a string as input and counts the number of vowels and consonants. It iterates through each character, checking if it is an alphabetical character and whether it is a vowel or consonant.

Question-2

Pseudo code-

```
function matrix_multiply(mat_A, mat_B):
    if length(mat_A[0]) != length(mat_B):
        return "Error: Matrices A and B are not multipliable."

    result_matrix = [[0 for _ in range(length(mat_B[0]))] for _ in range(length(mat_A))]

    for i in range(length(mat_A)):
        for j in range(length(mat_B[0])):
            for k in range(length(mat_B)):
                result_matrix[i][j] += mat_A[i][k] * mat_B[k][j]

    return result_matrix

mat_A_rows = input("Enter the number of rows for matrix A: ")
mat_A_cols = input("Enter the number of columns for matrix A: ")
mat_A = []

print("Enter elements for matrix A one by one by pressing enter after each element:")
for i in range(mat_A_rows):
    row = [input() for _ in range(mat_A_cols)]
    mat_A.append(row)

mat_B_rows = input("Enter the number of rows for matrix B: ")
mat_B_cols = input("Enter the number of columns for matrix B: ")
mat_B = []

print("Enter elements for matrix B one by one by pressing enter after each element:")
for i in range(mat_B_rows):
```

```
    row = [input() for _ in range(mat_B_cols)]
    mat_B.append(row)

result = matrix_multiply(mat_A, mat_B)

if is_string(result):
    print(result)
else:
    print("Matrix Multiplication Result:")
    for row in result:
        print(row)
```

Explanation-

The code defines a function, matrix_multiply, that performs matrix multiplication for two given matrices. It the user to input the size and elements of two matrices, and then it calculates and prints the result of the matrix multiplication or an error message if the matrices are not compatible for multiplication.

Question-3

Pseudo code-

```
function common_elements_count(list1, list2):
    common_elements = set(list1) & set(list2)
    return length(common_elements)

list1 = [int(x) for x in input("Enter elements for list A separated by space: ").split()]
list2 = [int(x) for x in input("Enter elements for list B separated by space: ").split()]

common_count = common_elements_count(list1, list2)
print("Number of Common Elements: " + common_count)
```

Explanation-

The code defines a function, common_elements_count, which takes two lists as input and calculates the count of common elements between them. It converts the lists to sets, finds the intersection, and returns the length of the resulting set. It asks the user to input two lists, and it prints the count of common elements.

Question-4

Pseudo code-

```
function transpose_matrix(matrix):
    return [[matrix[j][i] for j in range(length(matrix))] for i in range(length(matrix[0]))]


no_of_rows = input("Enter the number of rows for the matrix: ")
no_of_cols = input("Enter the number of columns for the matrix: ")
input_matrix = []


for i in range(no_of_rows):
    row = [input() for ele in input(f"Enter values for row {i + 1} (give space in between the numbers): ").split()]
    if length(row) != no_of_cols:
        print("Error: Invalid number of elements in a row. Please try again.")
        exit(1)
    input_matrix.append(row)


transposed_matrix = transpose_matrix(input_matrix)


print("Transposed Matrix:")
for row in transposed_matrix:
    print(row)
```

Explanation-

This code defines a function, transpose_matrix, that transposes a given matrix. It asks the user to input the size and elements of a matrix, checks for valid input, calculates the transposed matrix, and prints the result. The transposition swaps the rows and columns of the matrix.