Machine Learning

22AIE213

Assignment – 1

Submited by : Team 4

P.Anju chowdary-Aie22170

T.Mahithi Reddy-Aie22178

Akmal-Aie22150

P.Anju Chowdary -Aie22170 (set-2)

Q.1)Write a program to count the number of vowels and consonants present in an input string.

Pseudo code :

```
function Count_Vowels_And_Consonants(Input_String):

Vowel_Count = 0

Consonant_Count = 0

Vowels = set("aeiouAEIOU")

For character in Input_String:

if char .isalpha():

if char in Vowel_Count += 1

else:

Consonant_Count += 1

return Vowel_Count, Consonant_Count

function main():

Entry_Str = input( "Input string:")

Vowels_count, consonant_count = Count_Vowels_And_Consonants(Input_Str)

print("Number of vowels in string:", Vowel_count)

print("String Number string of consonants: ", consonant_count)

if __name__ == "__main__":

main()
```

Explanation :

Program code has function 'Count_Vowels_And_Consonants'; This is designed to count consonants, vowels and consonants. string Initializes the counter, repeats each character, and updates the count as a number or character. The "main" function requests user input, calls the calculation, and prints the result of the number and the number of digits.

It also has a check to ensure that the script is executed directly. If so, it plays an important role to ensure correct execution of the script as a standalone program.

Q.2)Write a program that accepts two matrices A and B as input and returns their product AB. Check if A & B are multipliable; if not, return error message

Pseudo code :

```
function Matrix_Multiplication(A, B):
    num_of_rows_A = len(A)
    num_of_cols_A = len(A[0])
    num_of_rows_B = len(B)
    num_of_cols_B = len(B[0])
    if num_of_cols_A != num_of_rows_B:
        return None
    Result_Matrix = [[0 for _ in range(num_of_cols_B)] for _ in
range(num_of_rows_A)]
    for i in range(num_of_rows_A):
        for j in range(num_of_cols_B):
            for k in range(num_of_cols_A):
                Result_Matrix[i][j] += A[i][k] * B[k][j]
    return Result_Matrix
function Get_Matrix_Input(label):
```

```
    matrix = []
    num_of_rows = input(f"Enter the number of rows for matrix {label}: ")
    num_of_cols = input(f"Enter the number of columns for matrix {label}: ")
    for i in range(num_of_rows):
        row = [input(f"Enter row {i + 1} for matrix {label} separated by space: ")]
        matrix.append(row)
    return matrix
# Main program
A = Get_Matrix_Input("A")
B = Get_Matrix_Input("B")
result = Matrix_Multiplication(A, B)
if result is not None:
    print("Matrix A:")
    for row in A:
        print(row)
    print("\nMatrix B:")
    for row in B:
        print(row)
    print("\nProduct (AB):")
    for row in result:
        print(row)
else:
    print("Error: Matrices A and B are not multipliable.")
```

Explanation :

The Matrix_Multiplication function works out the product of two matrices, which are A and B. It checks if the number of columns in matrix A is equal to the number of rows in matrix B, meaning multiplication is feasible. If this does not hold true, it returns None. Otherwise, it allocates an identity matrix as a product and runs the nested loops to calculate their dot products.

The Get_Matrix_Input function gets input from user on how many rows or columns a matrix labeled either ‘A’ or ‘B’ should have. For each row of the matrix, user is then prompted to enter each one respectively.

In order to get matrices A and B from Get_Matrix_Input function we use them in our main program. We check whether matrices can be multiplied and we then diplay our result if so; otherwise we provide error message.

Q.3)Write a program to find the number of common elements between two lists. The lists contain integers.

Pseudo code :

function Find_Common_Elements(list1, list2):

Set1 = set(list1)

Set2 = set(list2)

Common_Elements = Set1.intersection(Set2)

return len (Common_Elements)

function Get_List_From_User():

while True:

try:

user_input = input("Enter a space-separated list of integers:")

user_list = [int(x) user_input.split() for x]

return user_list

ValueError Only:

print("Invalid input. Please enter a valid number.")

Main program

List_a = Get_List_From_User()

while true:

try:

List_b = Get_List_From_User()

break

only ValueError:

print("Invalid input. Please enter a valid number.") < br>result = Find_Common_Elements(List_a, List_b)

print(f"Number of elements of List A and List B: {result}")

Explanation :

Find_Common_Elements function accepts two lists, performs the following operations : transformation to put them into a set, finds their intersection (points) and returns the number of points. The Get_List_From_User function requests a separate list of numbers from the user. It uses a try-only block to handle possible ValueErrors to ensure that the input contains valid code.

In the main program, the user will be asked to enter two names using Get_List_From_User. The program addresses misconceptions by displaying error messages and prompting for new ideas. Finally, the program calls Find_Common_Elements using the client list and prints the number of elements.

Q.4)Write a program that accepts a matrix as input and returns its transpose

Pseudo code :

Function Matrix_Transpose_(matrix): < br> Number_of-Rows = len(matrix)

Number_of-Columns = len(matrix[0])

Transpose = for j in range [[matrix[j][i](Number_of-Rows)] for i in range(Number_Columns)] < br> Transpose

Function Get_Matrix_Input():

```
Matrix = []

Num_of_Rows = input("Enter the number of rows of the matrix:" )

Num_of_Columns = input("Please enter the number of rows of the matrix Number of rows: ")

for me in range (Num_of_Rows):

rows = [input(f") enter for {i + 1} rows, separated by spaces: ") ]

matrix.append (rows)

For matrix

function print_matrix(matrix, label):

print(label + ":")

For row in matrix:

print (Low)

# Main program

Input_Matrix = Get_Matrix_Input ()

Transpose_of_Matrix(Input_Matrix)

print_matrix(Input_Matrix, "ORIGINAL MATRIX") print_of_matrix, print_matrix) ").
```

Explanation :

Get_Matrix_Input function asks the user for the row and column of matrix numbers, then click on the row and column of matrix numbers, then type the input matrix from the row to ensure that the use gets the integer input.

In the main program, use Get_Matrix_Input to get the matrix. Then use the Transpose_of_Matrix function to create the transposed matrix. Finally, use the print_matrix function to print the original and transposed matrices and display their text.

T.Mahithi Reddy -Aie22178 (set-2)

Q.1)Write a program to count the number of vowels and consonants present in an input string.

Pseudo code :

```
letters = []
function isVowel(letter):
    if letter is 'a' or letter is 'e' or letter is 'i' or letter is 'o' or letter is 'u':
        return True
    else:
        return False
function countVowelsAndConsonants(word):
    vowelCount = 0
    consonantCount = 0

    for letter in word:
        letters.append(letter)
        if isVowel(letter):
            vowelCount = vowelCount + 1
        else:
            consonantCount = consonantCount + 1
    print("The total number of vowels is:", vowelCount)
    print("The total number of consonants is:", consonantCount)
function main():
```

```
    userWord = getInput("Enter any word of your choice: ")
    countVowelsAndConsonants(userWord)
if __name__ == "__main__":
    main()
```

Explanation :

The given code finds out the number of vowels and consonants present in the given input string given by the user.

For loop is used in the code snippet to iterate every letter in the string and add the count of the vowels or consonants present respectively and finally the main function to call the counting function

Q.2)Write a program that accepts two matrices A and B as input and returns their product AB. Check if A & B are multipliable; if not, return error message.

Pseudo code :

```
function multiplyMatrices(matrixA, matrixB):
    rowsA = length(matrixA)
    columnsA = length(matrixA[0])
    rowsB = length(matrixB)
    columnsB = length(matrixB[0])
    if columnsA != rowsB:
        return "Oops!! Matrices are not multipliable."
    productMatrix = createMatrix(rowsA, columnsB, 0)
    for i = 0 to rowsA:
        for j = 0 to columnsB:
```

```
            for k = 0 to columnsA:
                productMatrix[i][j] += matrixA[i][k] * matrixB[k][j]
    return productMatrix
function createMatrix(rows, columns, initialValue):
    matrix = []
    for i = 0 to rows:
        row = [initialValue] * columns
        matrix.append(row)
    return matrix
function main():
    rowsA = input("Enter the number of rows in matrix A: ")
    columnsA = input("Enter the number of columns in matrix A: ")
    matrixA = inputMatrix(rowsA, columnsA)
    rowsB = input("Enter the number of rows in matrix B: ")
    columnsB = input("Enter the number of columns in matrix B: ")
    matrixB = inputMatrix(rowsB, columnsB)
    product = multiplyMatrices(matrixA, matrixB)
    if isString(product):
        print(product)
    else:
        print("Product of matrices A and B:")
        for row in product:
            print(row)
function inputMatrix(rows, columns):
    matrix = []
```

```
    print("Enter elements of matrix row-wise:")
    for i = 0 to rows:
        row = parseInputString(input())
        matrix.append(row)
    return matrix
function parseInputString(inputString):
    return [parseInt(x) for x in inputString.split()]
function isString(variable):
    return type(variable) is string
if __name__ == "__main__":
    main()
```

Explanation :

The "multiply_matrices" function in the code is for taking the user input from the user regarding the number of rows, columns, and their elements respectively also it determines whether a matrix is eligible for multiplication or not.

Q.3)Write a program to find the number of common elements between two lists. The lists contain integers.

Pseudo code :

```
function findCommonElements(listA, listB):
    setA = createSet(listA)
    setB = createSet(listB)
    if (setA intersect setB):
        print("COMMON ELEMENTS: ", setA intersect setB)  # Printing the common elements if found
    else:
```

```
        print("NO COMMON ELEMENTS FOUND")  # Printing message if no common elements are found

function createSet(inputList):

    return set(inputList)

function main():

    listA = parseInputList(input("ENTER THE ELEMENTS IN FIRST LIST: "))

    listB = parseInputList(input("ENTER THE ELEMENTS IN SECOND LIST: "))

    findCommonElements(listA, listB)

function parseInputList(inputString):

    return [parseInt(x) for x in inputString.split()]

if __name__ == "__main__":

    main()
```

Explanation :

From the code, the "common_elements" function takes the list input from the user and by using the intersection of both the sets it sounds out if there are any common elements in the given if not printing messages respectively.

The function "parseInputList" is used for returning the common elements from the lists given by the user.

Q.4)Write a program that accepts a matrix as input and returns its transpose

Pseudo code :

```
function transposeMatrix(matrix):

    num_rows = length(matrix)

    num_columns = length(matrix[0])

    transposed_matrix = createMatrix(num_columns, num_rows, 0)
```

```
    for i = 0 to num_rows:
        for j = 0 to num_columns:
            transposed_matrix[j][i] = matrix[i][j]
    return transposed_matrix
function createMatrix(rows, columns, initialValue):
    matrix = []
    for i = 0 to rows:
        row = [initialValue] * columns
        matrix.append(row)
    return matrix
function main():
    rows = parseInt(input("ENTER NUMBER OF ROWS IN A MATRIX: "))
    columns = parseInt(input("ENTER NUMBER OF COLUMNS IN A MATRIX: "))
    print("ENTER ELEMENTS IN A MATRIX ROW WISE : ")
    matrix = inputMatrix(rows, columns)
    transposedMatrix = transposeMatrix(matrix)
    print("ORIGINAL MATRIX :")
    for row in matrix:
        print(row)
    print("TRANSPOSE OF MATRIX: ")
    for row in transposedMatrix:
        print(row)
function inputMatrix(rows, columns):
    matrix = []
    for i = 0 to rows:
```

```
        row = parseInputString(input())
        matrix.append(row)
    return matrix
function parseInputString(inputString):
    return [parseInt(x) for x in inputString.split()]
if __name__ == "__main__":
    main()
```

Explanation :

The code starts with the "transposeMatrix" function which accepts the row and column count along with their elements respectively
The "createMatrix" function is used for storing the output matrix which is the transpose of the user input matrix in which the rows and columns are exchanged and at last returning the transpose matrix which is called in the "main" function in the code.

Akmal -Aie22150 (set -2)

Q.1)Write a program to count the number of vowels and consonants present in an input string.

Pseudo code :

```
function count_vowels_and_consonants(input_string):
vowels = "aeiouAEIOU"
consonants_count = 0
vowels_count = 0
for char in input_string:
on the off chance that char.isalpha():
```

```
in case char in vowels:

vowels_count += 1

else:

consonants_count += 1

return vowels_count, consonants_count

function main():

user_input = input(" Enter a String : ")

vowels_count, consonants_count = count_vowels_and_consonants(user_input)

print(f" No of vowels : {vowels_count}")

print(f" No of Consonants : {consonants_count}")

if __title__ == "__main__":

main()
```

Explanation :

The count_vowels_and_consonants work takes an input string and tallies the number of vowels and consonants. It repeats through each character within the string, checks in case it is an letter set character, and increases the particular counters based on whether it could be a vowel or consonant. The checks are at that point returned.

The fundamental work collects a client input string, calls the tallying work, and prints the tallies of vowels and consonants. The in case __title__ == "__primary__": piece guarantees that the most work is executed as it were when the script is run directly.

Q.2)Write a program that accepts two matrices A and B as input and returns their product AB. Check if A & B are multipliable; if not, return error message.

Pseudo code :

```
function input_matrix(rows, columns):

framework = []

print(f"enter the components in a {rows}x{columns} matrix:")
```

```
for i in range(rows):
push = []
for j in range(columns):
component = float(input(f" enter the component position ({i+1}, {j+1}): "))
row.append(element)
matrix.append(row)
return matrix
function matrix_multiply(A, B):
on the off chance that len(A[0]) != len(B):
return "Mistake: Lattices are not multipliable"
result = [[0 for _ in range(len(B[0]))] for _ in range(len(A))]
for i in range(len(A)):
for j in range(len(B[0])):
for k in range(len(B)):
result[i][j] += A[i][k] * B[k][j]
return result
function display_matrix(matrix, matrix_name):
print(f"nMatrix {matrix_name}:")
for push in matrix:
print(row)
# Fundamental program
rows_A = input("Enter thenumber of lines for network A: ")
columns_A = input("Enter the number of columns for lattice A: ")
matrix_A = input_matrix(rows_A, columns_A)
rows_B = input("Enter the number of columns for framework B: ")
```

```
columns_B = input("Enter the number of columns for lattice B: ")

matrix_B = input_matrix(rows_B, columns_B)

result = matrix_multiply(matrix_A, matrix_B)

if isinstance(result, str):

print(result)

else:

display_matrix(matrix_A, "A")

display_matrix(matrix_B, "B")

display_matrix(result, "AB (Product)")
```

Explanation :

The input_matrix work takes the number of lines and columns as input, prompts the client to enter framework components, and returns the made matrix.The matrix_multiply work checks in case the frameworks A and B are multipliable and, in the event that so, performs framework duplication, returning the result matrix.

The display_matrix work prints the network with a given name.In the most program, the user is provoked to input lattices A and B, and their increase result is shown on the off chance that conceivable. The program checks and handles cases where frameworks are not multipliable, giving an mistake message.

Q.3)Write a program to find the number of common elements between two lists. The lists contain integers.

Pseudo code :

```
function common_elements_count(list1, list2):

set1 = set(list1)

set2 = set(list2)

common_elements = set1.intersection(set2)

return len(common_elements)

# Fundamental program
```

```
list1 = input(" enter the components within the list1 seperated by space ").split()

list2 = input(" enter the components within the list2 seperated by space ").split()

list1 = [int(x) for x in list1]

list2 = [int(x) for x in list2]

result = common_elements_count(list1, list2)

print(f" number of common elemnts : {result}")
```

Explanation :

The common_elements_count work takes two records as input, changes over them into sets, finds their crossing point (common components), and returns the tally of common elements.

In the most program, the client is provoked to input components for two records, and the input is at that point changed over into numbers records. The common_elements_count work is called with these records, and the result is printed, showing the number of common components between the two lists.

Q.4)Write a program that accepts a matrix as input and returns its transpose

Pseudo code :

```
function transpose_matrix(matrix):

columns = len(matrix)

columns = len(matrix[0])

transpose_result = [[0] * lines for _ in range(columns)]

for i in range(rows):

for j in range(columns):

transpose_result[j][i] = matrix[i][j]

return transpose_result

function get_matrix_input():

lines = input(" enter the no of columns ")
```

```
columns = input(" enter the no of columns ")

lattice = []

print(" enter the components in network ")

for i in range(rows):

push = [int(x) for x in input().split()]

matrix.append(row)

return matrix

function print_matrix(matrix, label):

print(f"{label} Matrix:")

for push in matrix:

print(row)

# Primary program

matrix = get_matrix_input()

transpose_result = transpose_matrix(matrix)

print_matrix(matrix, "Unique ")

print_matrix(transpose_result, " Transposed ")
```

Explanation :

The transpose_matrix work performs network transposition by making an purge lattice and populating it based on the initial matrix's dimensions.

The get_matrix_input work collects client input for the number of rows and columns and after that accumulates framework sections, guaranteeing they are integers.

In the most program, the user inputs a framework, and the transposition is gotten utilizing the transpose_matrix work. Both the first and transposed lattices are shown utilizing the print_matrix work.