Machine Learning (22AIE213)

Team No. 9

Assignment 1- Report

Team Members:-

*R Hemanth Kumar*

BL.EN.U4AIE22138 (Team Leader)

*Vishnu Prasad B*

BL.EN.U4AIE22164

*Himanshu Yadav*

BL.EN.U4AIE22167

**Branch and section:** AIE ‘E’

**Date Of submission:** 04-02-2024

**REPORT BY: Hemanth Kumar AIE22138 (even set)**

Q1) Write a program to count the number of vowels and consonants present in an input string.

**Approach:-**

First created an list containing vowels, then called a function countVowels() that takes the user input(string) from the user. The function runs a loop through the string, each time a vowel is found, the variable vowelCount gets incremented by 1 and returns the vowelCount to the main function where the count is printed. Function countConsonants() takes the user input(string) from the user. Same as the previous function, it runs a loop through the string, each time a vowel is not found from the vowelList and a blankspace, the variable countConsonants gets incremented by 1 and returns the countConsonants to the main function.

**Pseudocode:-**

SET vowels to ['a','e','i','o','u']

PROCEDURE countVowels:

SET vowelCCount to 0

FOR each element in the string:

IF element is in vowels:

increment vowelCount +1

RETURN vowelCount

PROCEDURE countConsonants:

SET consonantsCount to 0

FOR each element in the string and has a blankspace:

IF element is not in vowels:

increment consonantsCount +1

RETURN consonantsCount

PROMPT inputString from user

DISPLAY number of vowel

DISPLAY number of consonants

PROCEDURE matrixMultiplication having matrixA and matrixB:

IF row of matrixA is equal to column of matrixB:

SET matrixC as function call emptyMatrix with the rows and columns:

FOR each element in range 0 to rows of matrixA:

FOR each element in range 0 to columns of matrixB:

SET sum to 0:

FOR each element in range 0 to rows of MatrixB:

SET sum as matrixA multiplied to matrixB

SET matrixC index to sum

RETURN matrixC

ELSE:

RETURN false

Q2) Write a program that accepts two matrices A and B as input and returns their product AB.Check IF A & B aremultipliable; IF not, return error message.

**Approach:-**

My approach consists of dynamic empty matrix creation by EmptyMatix() which creates a empty matrix with the row of MatrixA x column of MatrixB which will then help in editing the result matrix MatrixC. Function convertStringtoMatrix() converts the user input which is in string to matrix, first a loop is initiated to process through the string and filters out the numeric values using isnumeric() and STOREs it in temp, then by the rows and columns as parameters converts the list into a 2-dimensional matrix. The rows are found using len(matrixA) and columns are found using len(matrixA[0]). The multiplication is done by the function MatrixMultiplication() that takes the user input of matrixA and matrixB. First it checks IF the matrix can be multiplied or not, to be able to, matrixA’s column size must be same as MatrixB IF satisfied, then empty matrixC is created, and then using three loop and iteration variables I,j,k individually represents rows, columns and row-columns multiplication. Loops thought the matrix and STOREs the sum value of a row-columns calculation individually to MatrixC. IF the multiplication is not possible then returns false value which will then be passed to the main program where IF false IF returned, message is printed saying that the matrices cannot be multiplied.

**Pseudocode:-**

PROCEDURE convertStringto Matrix having rows and columns:

PROMPT string

DECLARE matrix as list

DECLARE temp as list

FOR each element in string:

IF the element is a number:

INSERT into temp

SET x to 0

FOR each element in range, ranging from 0 to rows:

SET tempmatrix as emptylist:

FOR each element in range, ranging from 0 to columns:

SET A to temp of xth index

INSERT a into tempmatrix

increment x by 1

APPEND tempmatrix to matrix

RETURN matrix

PROCEDURE emptyMatrix having rows and columns:

SET EmptyMatrix as emptylist

FOR each element in range, ranging from 0 to rows:

SET tempmatrix as emptylist:

FOR each element in range, ranging from 0 to columns:

APPEND 0 to tempMatrix in the index

APPEND tempmatrix to matrix

RETURN matrix

PROMPT user to input matrixA

PROMPT user to input matrixB

STORE the result

IF result is false:

DISPLAY cannot be multiplied due to dimension mismatch

Q3) Write a program to find the number of common elements between two lists. The lists contain integers.

**Approach:-**

The user input will have the datatype string even though list is entered, to solve this convertStringtoArray is used. Which filters only the numeric value and appends into the tempList which will then be returned to the main program. Two lists are created as such. To find the common elements, first declared an commonlist[] that will STORE the common elements. Then overwrited the list1 as the combination of list1 and list2. A loop in run, IF the list1.count() for an element is more than 1 then the element in appended to commonlist and remove the element from the list, after the loop has completed executing, the commonList[] may have duplicate values, this is solved by converting the list to set and then returning the set to the main program.

**Pseudocode**:

PROCEDURE commonelement having list1 and list2:

ADD list1 and list2 and STORE in list1

DECLARE commonlist as list

FOR each element in list1:

IF count of the selected list is more than 1:

APPEND the element to commonlist

FOR each element in list1 and list2:

IF the element is equal to the commonelement:

REMOVE the commonelement from list1

CONVERT commonlist to SET

RETURN commonlist

PROCEDURE convertStringtoList():

PROMPT string

DECLARE templist as list

FOR each element in string:

IF the element is a number:

INSERT into templist

CONVERT tempList to sets

RETURN templist

DISPLAY function call commonelement

Q4) Write a program that accepts a matrix as input and returns its transpose.

**Approach:-**

EmptyMatix() creates a empty matrix with the row of MatrixA x column of MatrixB which will then help in editing the result tranposeMatrix. Function convertStringtoMatrix() converts the user input which is in string to matrix, first a loop is initiated to process through the string and filters out the numeric values using isnumeric() and STOREs it in temp, then by the rows and columns as parameters converts the list into a 2-dimensional matrix. The rows are found using len(matrix) and columns are found using len(matrix[0]). The multiplication is done by the function Transpose(), that rearranges the elements, for example changing the index [1][0] STOREs directly to the transpose matrix of index [0][1]. Finaly the transpose matrix is returned to the main program.

**PseudoCode:-**

PROCEDURE transpose having matrix:

SET transposeMatrix to function call emptymatrix with rows and columns

FOR each element in range 0 to rows:

FOR each element in range 0 to columns:

SET interchanged index of matrix to transpose matrix

RETURN transposeMatrix

PROCEDURE emptyMatrix having rows and columns:

SET EmptyMatrix as emptylist

FOR each element in range, ranging from 0 to rows:

SET tempmatrix as emptylist:

FOR each element in range, ranging from 0 to columns:

APPEND 0 to tempMatrix in the index

APPEND tempmatrix to matrix

RETURN matrix

PROCEDURE convertStringtoArray having rows and columns:

PROMPT string

DECLARE matrix as list

DECLARE temp as list

FOR each element in string:

IF the element is a number:

INSERT into temp

SET x to 0

FOR each element in range, ranging from 0 to rows:

SET tempmatrix as emptylist:

FOR each element in range, ranging from 0 to columns:

SET A to temp of xth index

INSERT a into tempmatrix

increment x by 1

APPEND tempmatrix to matrix

RETURN matrix

PROMPT user to input rows and columns

DISPLAY the transposed matrix

**REPORT BY: Vishnu Prasad B AIE22164 (even set)**

QUESTION 1:

The approch of this question is quite simple i have used two functions `vowel\_count` and `consonant\_count` for counting vowels and consonants . The `vowel\_count` function iterates through each character in the string increments a counter whenever a vowel amoung 'a', 'e', 'i', 'o', 'u'. Next the `consonant\_count` function removes spaces from the input string then initializes a consonant counter which increments it for each non vowel character. The code then accepts a string from the user then it calculates and prints the count of vowels and consonants. To understand it further we can use the pesudo code given below

Function vowel\_count(input\_str):

count = 0

For each letter in input\_str:

If letter is 'a' or 'e' or 'i' or 'o' or 'u':

Increment count by 1

Return count

Main program:

Display "Enter the String: "

Accept user input and convert it to lowercase

Call vowel\_count function with the input string and store the result in vowel\_count variable

Display "Number of vowels: ", vowel\_count

Remove spaces from the input string

Calculate the number of consonants by subtracting vowel\_count from the total number of alphabets

Display "Number of consonants: ", consonant\_count

QUESTION 2:

The approch of this question is little complicated in this there is a matrix multiplication function named `multiply`. This function takes two matrices A and B along with their dimensions and returns their product and tells if multiplication is possible. The code then prompts the user to input the dimensions and elements of matrices A and B. Afterward it calls the `multiply` function to calculate their product matrix. The code includes a check to ensure that the matrices are valid for multiplication . If it is not then it prints a message stating that multiplication is not possible. Finally, the resulting product matrix is displayed. The matrix multiplication is performed using nested loops to iterate through the matrices elements. To understand this further we can see the pesudo code .

Function multiply(A, B, ROW\_A, Column\_A, ROW\_B, Column\_B):

If Column\_A is not equal to ROW\_B:

Display "Multiplication not possible"

Return None

Else:

Initialize matrix C with zeros

For each row i in A:

For each column j in B:

For each index k:

C[i][j] += A[i][k] \* B[k][j]

Return C

Main program:

Input: ROW\_A, Column\_A for matrix A

For each row i in range(ROW\_A):

For each column j in range(Column\_A):

Input element A[i][j]

Input: ROW\_B, Column\_B for matrix B

For each row i in range(ROW\_B):

For each column j in range(Column\_B):

Input element B[i][j]

Call multiply function with matrices A, B, ROW\_A, Column\_A, ROW\_B, Column\_B

Print the resulting matrix product C

QUESTION 3:

The approch of this question is simple in this there is a function named `common\_elements` that calculates and returns the count of common elements between two lists. The code uses nested loops to iterate through the elements of both lists and increments the counter (`similarity\_count`) whenever a match is found. The code then askes the user to input the lengths and elements for both lists. The `common\_elements` function is called and the resulting count of common elements is printed. To understand this further we can see the pesudo code.

Function common\_elements(list1, list2):

similarity\_count = 0

For each element i in list1:

For each element j in list2:

If i is equal to j:

Increment similarity\_count by 1

Break out of the inner loop

Return similarity\_count

Main program:

Initialize empty lists list1 and list2

Input: list1\_length

Input: list1 elements

For each element n1 in list1:

Append n1 to list1

Input: list2\_length

Input: list2 elements

For each element n2 in list2:

Append n2 to list2

Call common\_elements function with lists list1 and list2

Print the number\_of\_common\_elements

QUESTION 4:

In this approach there is a function called `Transpose` that finds out the transpose of a given matrix. The program asks the user to input the number of rows and columns for the matrix and then accept the matrix elements. The function uses a nested loop structure to traverse through the rows and columns of the original matrix and constructs the transpose of the matrix by swapping the positions of elements. The original matrix and its transposed matrix are prints it. To understand this further we can see the pesudo code.

Function Transpose(matrix):

Initialize result matrix with zeros

For each row i in matrix:

For each column j in matrix:

Assign the transpose element to result[j][i]

Return result

Main program:

Input: rows

Input: columns

Initialize empty matrix

Print "Enter matrix elements:"

For each row i in range(rows):

Initialize empty row

For each column j in range(columns):

Input element

Append element to row

Append row to matrix

Call Transpose function with matrix

Print "Original Matrix:"

For each row in matrix:

Print row

Print "Transposed Matrix:"

For each row in transposed\_matrix:

Print row

**REPORT BY: Himanshu Yadav AIE22167 (odd set)**

Q1) Consider the given list as [2,7,4,1,3,6].Write a program to count pairs of elements with sum equal to 10.

Approach:-

Function count\_pairs takes the parameter as the input list, loops are used to iterate thought the string and find the count of pairs and target sum each time, then returns the value to the main

Pseudocode:-

function count\_pairs parameters input\_list:

set target\_sum as 10

set pairs\_count as 0

for elements from 0 to length of input\_list:

for elements from i+1 to lenght of input\_list:

if input\_list[i] add input\_list[j] is target\_sum:

increment pairs\_count by 1

return pairs\_count

declare test\_list

call countpairs with test\_list as parameter and store in total\_pairs

display total\_pairs

Q2) .Write a program that takes a list of real numbersas input and returns the range(difference between minimum and maximum) of the list.Check for list being lessthan 3 elementsin whichcase return anerror message(Ex: “Range determination not possible”).Given a list[5,3,8,1,0,4],therange is 8(8-0).

**Approach:-**  
function list\_range takes the input\_list as the parameter, in this function it checks that the length of the list is less than 3 or not, range\_of \_list will be computed by subrtracting the min for inputlist to the maximum of the inputlist, the result will then be displayed in the main function

**Pseudocode:-**

function list\_range parameters input\_list:

if length of input\_list is lesser than 3:

return not possible

set range\_pf\_list as max of input\_list - minimum of input\_list

return range\_of\_list

declare test\_list

call list\_range with test\_list as parameter and store in range\_of\_list

display range\_of\_list

Q3).Write a program that accepts a square matrix A and a positive integer m as arguments and returns Am.

**Approach:-**

This is nearly just like how we multiply the matrices just we pass the power parameter and loop the process of multiplication for the corresponding power, but each the the matrix is multiplied the result is overwritten each loop therefore resulting in the A^m

**Pseudocode:-**

function calculate\_matrix\_power paramenter matrix and power:

set result\_matrix as copy of original matrix

for elements from power - 1:

declare temp\_matrix as list

for elements from 0 to rows:

declare temp\_row as list:

for elements from 0 to columns:

set temp\_value as 0

for elements in range of rows:

add,multiply,set tempvalue, matrix with result\_\_matrix

append tempvalue to temp\_row

set resultmatrix as temp\_matrix

return result\_matrix

initialise inputmatrix

set m as 3

call calculate\_matrix\_power

Q4).Write a program to count the highest occurring character & its occurrence count in an inputstring. Consideronly alphabets. Ex: for “hippopotamus”as inputstring, the maximally occurring character is ‘p’& occurrence count is 3.Display result

**Approach:-**

Function max\_occurence\_character takes the string as parameter, a loop is initiated in which for each iteration check thourhg the character of the string and stores the character and count in the dictionary as key and value. Finally the max count is displayed

**Pseudocode:-**

function max\_occurenece\_character parameters input\_string:

declare charcount as dictionary

declare max\_character as string

set max\_count as 0

for characters in the input\_string:

if characters in char\_count:

increment characount[char] by charcount[char]+1

else:

set char\_count[char]=1

if char\_count[char] is greater than max\_count:

set max\_char as char

set max\_count as char\_count[char]

return max\_char and max\_count

set input\_str

call max\_occurence\_character

display results