MACHINE LEARNING ASSIGNMENT 1

R G ROHIT

K RAHUL RAO

BHARATH TEJA

Amrita Vishwa Vidyapeetham

22AIE213 MACHINE LEARNING

Introduction

In this report, we have discussed the different Python functions designed to perform various tasks to make our lives easier. These functions cover a range of operations from counting pairs in an array to determining the highest frequency character in a string. Each function serves a purpose and demonstrates the concepts such as the usage of dictionaries, sets, iteration, and conditionals.

**Functions used in Rohit’s question**

Function 1: Counting Pairs

The counting\_pairs function takes an array of integers as input and counts the number of pairs within the array that sum up to a specific value. It utilizes a set to efficiently keep track of previously encountered numbers and iterates through the array to identify pairs. We used sets to keep count and the value into the set. The difference is being calculated and the number of times a particular answer repeats is the amount of pairs that have a total sum of 10. The count initially is initialized to zero so in case there are 3 number of similar values in total, the final number of pairs will be 2.

Function 2: Range Determination

The range function calculates the range of a given array, defined as the difference between the maximum and minimum values within the array. It checks if the length of the array is sufficient to determine the range and returns an appropriate message if not. We used max and min functions to find the maximum and minimum values respectively in the array so that we can subtract to find the range.

Function 3: Matrix Multiplication

The matrix\_multiply function performs matrix multiplication for two input matrices represented as nested lists. It utilizes nested loops to iterate through the rows and columns of the matrices and calculates the resulting matrix by summing the products of corresponding elements.

Function 4: Matrix Power

The matrix\_power function raises a given matrix to a specified power using matrix multiplication. It repeatedly calls the matrix\_multiply function to compute the result.

Function 5: Highest Occurrence

The highest\_occurrence function determines the character with the highest frequency in a given string. We are setting a dictionary to count the occurrences of each character and identifies the character with the maximum frequency. We store the number of time a certain letter repeats and the use the max to find the maximum value which is basically the letter with maximum number of occurences.

**Pseudo Code**

Function 1: Counting Pairs

counting\_pairs(arr1):

Initialize count as 0

Initialize an empty set checked

For each num in arr1:

Calculate difference = 10 - num

If difference is in checked:

Increment count by 1

Add num to checked set

Return count

Function 2: Range Determination

range(arr2):

If length of arr2 is less than 3:

Return "Range determination not possible"

Else:

Return the difference between the maximum and minimum values in arr2

Function 3: Matrix Multiplication

matrix\_multiply(A, B):

Initialize an empty list result3

For each row in A:

Initialize an empty list row

For each column in B:

Initialize sum as 0

For each element k in range(length of B):

Update sum by adding A[row][k] \* B[k][column]

Append sum to the row

Append row to result3

Return result3

matrix\_power(A, m):

Set result3 to A

Iterate from 1 to m-1:

Update result3 by calling matrix\_multiply(result3, A)

Return result3

Function 4: Highest Occurrence

highest\_occurrence(input\_string):

Initialize an empty dictionary counter1

For each letter in input\_string:

If letter is in counter1:

Increment the value of counter1[letter] by 1

Else:

Set counter1[letter] to 1

Find the letter with maximum frequency in counter1

Return a string indicating the character with highest frequency.

**Algorithm**

Function 1: Counting Pairs

1. Initialize count to 0

2. Create an empty set called checked

3. Iterate through each num in arr1:

a. Calculate difference as 10 minus num

b. Check if difference is in the checked set:

i. If yes, increment count by 1

c. Add num to the checked set

4. Return count

Function 2: Range Determination

1. If the length of arr2 is less than 3:

a. Return "Range determination not possible"

2. Else:

a. Determine the maximum value in arr2

b. Determine the minimum value in arr2

c. Return the difference between the maximum and minimum values

Function 3: Matrix Multiplication

1. Initialize an empty list called result3

2. Iterate through each row in matrix A:

a. Initialize an empty list called row

b. Iterate through each column in matrix B:

i. Initialize sum as 0

ii. Iterate through each element k in range(length of B):

- Update sum by adding A[row][k] multiplied by B[k][column]

iii. Append sum to the row

c. Append the row to result3

3. Return result3

1. Set result3 to A

2. Iterate i from 1 to m-1:

a. Update result3 by calling matrix\_multiply(result3, A)

3. Return result3

Function 4: Highest Occurrence

1. Initialize an empty dictionary called counter1

2. Iterate through each letter in input\_string:

a. If letter is already a key in counter1:

i. Increment the value of counter1[letter] by 1

b. Else:

i. Set counter1[letter] to 1

3. Find the letter with the maximum frequency in counter1

4. Return a string indicating the character with the highest frequency

Functions used in Rahul’s questions

Question 1: finding number of vowels in a given string

The find\_vowels() function takes a string as input from the main function then we define a list

Consisting of vowels.then we initialize count=0 because we still didn’t run the loop and we didnt

Any vowels, it’s the same with consonants.then we iterate every element in string1 and check if

Matches with elements from list vowels if it does vowels count+1 if not consonant count+1 then

Then we return the count of vowels and count of consonants.

Pseudo code =

FUNCTION find\_vowels(string1):

vowels = ["a", "e", "i", "o", "u"]

count\_of\_vowels = 0

count\_of\_consonants = 0

FOR character in string1:

IF character in vowels:

count\_of\_vowels = count\_of\_vowels + 1

else:

count\_of\_consonants = count\_of\_consonants + 1

RETURN (count\_of\_vowels, count\_of\_consonants)

Question 2: to find common elements in 2 lists

First we define list1 and list2 in the function def common\_list() later we initialize count =0 because we didn’t run the loop and number of common elements found is 0. Iterate in list1 and compare every element in list 1 with every element in list 2 and if the element matches count = count +1 then we return the value to the function.

Pseudo code:

FUNCTION common\_list(list1, list2):

count\_of\_common\_elements = 0

FOR element in list1:

IF element in list2:

count\_of\_common\_elements = count\_of\_common\_elements + 1

RETURN count\_of\_common\_elements

Question 3:to find transpose of a matrix

First we take a matrix as input from main function and we make a copy of that which stores nothing.

Later we iterate to get the total number of rows and columns after getting it we check if index I is not equal to j then we strore matrix2[j][i] in temp,input matrix [i][j] in matrix2[j][i],temp in inputmatrix[i][j].

Pseudo code:

FUNCTION find\_transpose(input\_matrix):

matrix2 = input\_matrix.copy()

FOR i in range(len(input\_matrix)):

FOR j in range(len(input\_matrix[0])):

IF i != j:

TEMP= matrix2[j][i]

matrix2[j][i] = input\_matrix[i][j]

input\_matrix[i][j] = TEMP

RETURN matrix2

Question 4:to multiply to matrices

We take two matrices as an input from userThere is a simple concept to be noted for python it

Is number of elements for us it is number of rows.

We use three loops iterate through matrix a , matrix b order 0, matrix b

Result [i][j]=matrix a[i][k]\*matrix b[k][j]+result[i][j]

Pseudo code:

FUNCTION matrix\_multiplication(Matrix\_a, Matrix\_b):

RESULT = [[0 for column in range(len(Matrix\_b[0]))] for row in range(len(Matrix\_a))]

FOR i in range(len(Matrix\_a)):

FOR j in range(len(Matrix\_b[0])):

FOR k in range(len(Matrix\_b)):

result[i][j] += Matrix\_a[i][k] \* Matrix\_b[k][j] return result

**Function’s used in Bharath Teja’s Questions**

Question1)

Pseudo code:

function find\_sum\_pairs(numbers\_list, target\_sum):

pair\_count = 0

for i from 0 to length(numbers\_list) – 1:

for j from i + 1 to length(numbers\_list) – 1:

if numbers\_list[i] + numbers\_list[j] == target sum:

pair\_count += 1

print(“pair”, Pair\_count)

print(numbers\_list[i], “--”, numbers\_list[j])

print(“Numbers of pairs with sum”, target\_sum, “is:”, pair\_count)

# Input values and call the function with the list and target sum

numbers\_list = [2, 4, 7, 1, 3, 6]

target\_sum = 10

find\_sum\_pairs(numbers\_list, target\_sum)

Explanation:

In the above code, I used two loops which searches for pairs of numbers in the given list that add up to specified target sum and prints the pairs of those numbers along with the count of the pairs.

Question2)

Pseudo code:

function find\_min\_and\_max(input\_list):

if input\_list is empty:

print(“Empty list. Range determination not possible.”)

return None

min\_value = max\_value = input\_list[0]

for num in input\_list[1:]:

if num < min\_value:

min\_value = num

if num > max\_value:

max\_value = num

return min\_value, max\_value

function calculate\_range(min\_value, max\_value):

return max\_value – min\_value

function determine\_and\_print\_range(input\_list):

if length(input\_list) < 2:

print(“Insufficient elements for range determination.”)

return None

print(“Range determination is possible.”)

min\_val, max\_val = find\_min\_and\_max(input\_list)

range\_val = calculate\_range(min\_val, max\_val)

print(“Minimum value:”, min\_val)

print(“Maximum value:”, max\_val)

print(“Range:”, range\_val)

return range\_val

# Example usage

my\_list = [5, 3]

result = determine\_and\_print\_range(my\_list)

Explanation:

In the above code, I used one function which takes a list of numbers as input and checks if it is empty and returns minimum and maximum values.

In the second function, it takes minimum and maximum values as input and calculates the range, and the calculated range is returned.

In the third function, it prints the minimum and maximum values, as well as the range of the given list of numbers.

Question3)

Pseudo code:

function matrix\_power(A, power):

order = length(A)

identity\_matrix = create\_identity\_matrix(order)

for \_ in range(power):

identity\_matrix = matrix\_multiply(identity\_matrix, A)

return identity\_matrix

function matrix\_multiply(A, B):

rows\_A, cols\_A = dimension(A)

rows\_B, cols\_B = dimension(B)

if cols\_A! = rows\_B:

print(“Matrices cannot be multiplied.”)

return None

result = create\_empty\_matrix(rows\_A, cols\_B)

for i from 0 to rows\_A -1:

for j from 0 to cols\_B – 1:

for k from 0 to cols\_A – 1:

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

return result

# Example usage

matrix\_A = [[1, 2, 3],[5, 6, 7],[1, 2, 4]]

power\_value = 4

result\_matrix = matrix\_power(matrix\_A, power\_value)

print(result\_matrix)

Explanation:

In the above code, I used one function which takes matrix and a power value as input.

In the second function, it performs the matrix multiplication for the specified number of times. And print the appropriate matrix(i.e result matrix).

Question4)

Pseudo code:

function find\_highest\_occurence(input\_str):

char\_frequency = create\_empty\_dictionary()

for char in input\_str:

# if the character is an alphabet, update its frequency in the dictionary

if is\_alphabet(char):

if char is in char\_frequency:

char\_frequency[char] += 1

else:

char\_frequency[char] = 1

# Find the character with the maximum frequency

most\_common\_char = get\_key\_with\_max\_value(char\_frequency)

# Output the character with its occurrence count

max\_occurence\_count = char\_frequency[most\_common\_char]

print(“ The most common alphabet is ” ,most\_common\_char, “ with a count of: ”, max\_occurence\_count)

# Example usage

given\_string = “ Bharathteja ”

find\_highest\_occurence(given\_string)

Explanation:

In the above code, I used one function which takes a string as input and iterates through each character in the given input string. And prints most frequently occurring alphabet in a given string. So, this finds the most frequently occurring alphabet in a given string.