Python Operations without numpy or sklearn libraries

Q1: Given two matrices please print the product of those two matrices

```
Ex 1: A   = [[1 3 4]
             [2 5 7]
             [5 9 6]]
      B   = [[1 0 0]
             [0 1 0]
             [0 0 1]]
      A*B = [[1 3 4]
             [2 5 7]
             [5 9 6]]


Ex 2: A   = [[1 2]
             [3 4]]
      B   = [[1 2 3 4 5]
             [5 6 7 8 9]]
      A*B = [[11 14 17 20 23]
             [23 30 37 44 51]]


Ex 3: A   = [[1 2]
             [3 4]]
      B   = [[1 4]
             [5 6]
             [7 8]
             [9 6]]
      A*B =Not possible
```

In [79]:

```
#MATRIX MULTIPLICATION
#DEFINING FUNCTION
def matrix_mul(A,B):
    M=[]
    for i in range(r1):
        ele=[]
        for j in range(c2):
            ele.append(int(0))
        M.append(ele)

    for i in range(len(A)):
        for j in range(len(B[0])):
            for k in range(len(B)):
                M[i][j]+=A[i][k]*B[k][j]
    return(M)



#INSERTING MARTIX A[]
A=[]
r1=int(input('Enter the no of rows for matrix A....'))
c1=int(input("Enter the no of columns for matrix A...."))
print("Enter the entries rowwise....")
for i in range(r1):
    ele=[]
    for j in range(c1):
        ele.append(int(input()))
    A.append(ele)

print("The entered matrix A is\n ")
for i in range(r1):
    for j in range(c1):
        print(A[i][j] ,end="  ")
    print()
print()


#INSERTING MATRIX B[]
B=[]
r2=int(input('Enter the no of rows for matrix B....'))
c2=int(input("Enter the no of columns for matrix B...."))
print("Enter the entries rowwise....")
for i in range(r2):
    ele=[]
    for j in range(c2):
        ele.append(int(input()))
    B.append(ele)

print("The entered matrix B is\n ")
for i in range(r2):
    for j in range(c2):
        print(B[i][j] ,end="  ")
    print()
print()

#CHECKING WHETHER BOTH MATRICES CAN BE MULTIPLIED OR NOT...
if(c1==r2):
    X=[]
    X=matrix_mul(A, B)
    print("The multiplication of those two matrices is...\n")
    for x in X:
        print(x)
else:
    print("The matrix multiplication is not apllicable for given matrices")
```

```
Enter the no of rows for matrix A....2
Enter the no of columns for matrix A....2
Enter the entries rowwise....
1
2
3
4
The entered matrix A is

1  2
3  4

Enter the no of rows for matrix B....2
Enter the no of columns for matrix B....2
Enter the entries rowwise....
1
2
3
4
The entered matrix B is

1  2
3  4

The multiplication of those two matrices is...

[7, 10]
[15, 22]
```

Q2: Proportional Sampling - Select a number randomly with probability proportional to its magnitude from the given array of n elements

Consider an experiment, selecting an element from the list A randomly with probability proportional to its magnitude. assume we are doing the same experiment for 100 times with replacement, in each experiment you will print a number that is selected randomly from A.

```
Ex 1: A = [0 5 27 6 13 28 100 45 10 79]
let f(x) denote the number of times x getting selected in 100 experiments.
f(100) > f(79) > f(45) > f(28) > f(27) > f(13) > f(10) > f(6) > f(5) > f(0)
```

In [6]:

```
#PROPORTIONAL SAMPLING
from random import uniform

#PICKING A NUMBER FROM LIST A[] USING PROPORTIONAL SAMPLING ALGORITHM
def pick_a_number_from_list(A):
    total=0
    a=0
    lst=[]
    for i in range (len(A)):
        total+=A[i]
        #print('total is ',total)
    for i in range (len(A)):
        lst.append(a+A[i]/total)
        a=a+A[i]/total
    #print("\nCumulated normalised value is...\n",lst)
    random_number=uniform(0.0,1.0)
    #print("Random number is\n ",random_number)
    for i in range (0,len(A)):
        if (random_number<lst[i]):
            return A[i]

#GETTING INPUT FOR LIST A[]
def sampling_based_on_magnitued():
    A=[]
    n=int(input("Enter the numbers of values you are going to enter into the list"))
    for i in range(0,n):
        A.append(int(input(f"Enter the {i}th element to the list...")))
    print(f'\nList A is {A}\n')
   # A = [0,5,27,6,13,28,100,45,10,79]
    for i in range(1,100):
        number = pick_a_number_from_list(A)
        print(f'The picked number is...{number}\n')

sampling_based_on_magnitued()
```

```
Enter the numbers of values you are going to enter into the list5
Enter the 0th element to the list...100
Enter the 1th element to the list...6
Enter the 2th element to the list...3
Enter the 3th element to the list...2
Enter the 4th element to the list...1

List A is [100, 6, 3, 2, 1]

The picked number is...100

The picked number is...100

The picked number is...100

The picked number is...100

The picked number is...100
```

Q3: Replace the digits in the string with

consider a string that will have digits in that, we need to remove all the not digits and replace the digits with #

```
    Ex 1: A = 234               Output: ###
    Ex 2: A = a2b3c4            Output: ###
    Ex 3: A = abc               Output:   (empty string)
    Ex 5: A = #2a$#b%c%561#     Output: ####
```

In [13]:

```
#REPLACE THE DIGITS IN THE STRING WITH '#'
import re

#REPLACE_DIGITS FUNCTION
def replace_digits(S):
    for i in A:
        NS='#'*len(re.sub(r'\D','',i))
        print(f'The inputted string is {i} and its output is: {NS}')

#GETTING STRING INPUT
A=[]
n=int(input("Enter the numbers of strings you are going to enter..."))
for i in range(0,n):
    A.append(input("\nEnter the {i}th string..."))
    print(f'\nStr A is {A}\n')

#CALLING REPLACE_DIGITS FUNCTION
replace_digits(A)
```

```
Enter the numbers of strings you are going to enter...3

Enter the {i}th string...123

Str A is ['123']


Enter the {i}th string...1a2b3c

Str A is ['123', '1a2b3c']


Enter the {i}th string...abc@#

Str A is ['123', '1a2b3c', 'abc@#']

The inputted string is 123 and its output is: ###
The inputted string is 1a2b3c and its output is: ###
The inputted string is abc@# and its output is:
```

Q4: Students marks dashboard

consider the marks list of class students given two lists
Students = ['student1','student2','student3','student4','student5','student6','student7','student8','student9','student10']
Marks = [45, 78, 12, 14, 48, 43, 45, 98, 22, 80]
from the above two lists the Student[0] got Marks[0], Student[1] got Marks[1] and so on

your task is to print the name of students **a. Who got top 5 ranks, in the descending order of marks**
b. Who got least 5 ranks, in the increasing order of marks
d. Who got marks between >25th percentile <75th percentile, in the increasing order of marks

```
Ex 1:
Students=['student1','student2','student3','student4','student5','student6','student7','student8','student9','student10']
Marks = [45, 78, 12, 14, 48, 43, 47, 98, 22, 80]
a.
student8  98
student10 80
student2  78
student5  48
student7  47
b.
student3 12
student4 14
student9 22
student6 43
student1 45
c.
student9 22
student6 43
student1 45
student7 47
student5 48
```

In [12]:

```
#STUDENTS MARKS DASHBOARD USING LAMBDA FUNCTION
students = ['student1','student2','student3','student4','student5','student6','student7','student8','student9','student10']
marks = [45, 78, 12, 14, 48, 43, 47, 98, 22, 80]
combined=list(zip(students,marks))
sl=sorted(combined, key=lambda a:a[1])
#print(sorted(marks))


#TOP 5 STUDENTS
print('\nThe top 5 ranks students are...')
for i in range(len(sl[:5])):
    print(sl[-i-1])


#LEAST 5 STUDENTS
print('\nThe least 5 ranks students are...')
for i in range(len(sl[:5])):
    print(sl[i])


#STUDENTS BETWEEN 25 AND 75 PERCENTILE
max1=max(marks)
min1=min(marks)
diff=max1-min1
m25th=diff*0.25
m75th=diff*0.75
print('\nStudents between 25 and 75 percentile are...')
for i in range(len(sl)):
    if (sl[i][1]>m25th and sl[i][1]<m75th):
        print(sl[i])
```

```
The top 5 ranks students are...
('student8', 98)
('student10', 80)
('student2', 78)
('student5', 48)
('student7', 47)

The least 5 ranks students are...
('student3', 12)
('student4', 14)
('student9', 22)
('student6', 43)
('student1', 45)

Students between 25 and 75 percentile are...
('student9', 22)
('student6', 43)
('student1', 45)
('student7', 47)
('student5', 48)
```

In [12]:

```
#STUDENTS MARKS DASHBOARD
students = ['student1','student2','student3','student4','student5','student6','student7','student8','student9','student10']
marks = [45, 78, 12, 14, 48, 43, 47, 98, 22, 80]

def display_dash_board(students, marks):
    l=[]
    ascending=[]
    ascending=sorted(marks)
    decending=sorted(marks,reverse=True)
    #print(marks)
    #print(ascending)
    #print(decending)

    #TOP 5 STUDENTS
    print('The top 5 ranks students are...')
    for i in range (len(marks)):
        l.append([students[i],marks[i]])
    for i in decending[0:5]:
        for j in range(len(decending)):
            if (l[j][1]==i):
                print(l[j][0],i)
    print()

    #LEAST 5 STUDENTS
    print('The least 5 ranks students are...')
    for i in ascending[0:5]:
        #print(i)
        for j in range(len(ascending)):
            #print(j)
            if (l[j][1]==i):
                print(l[j][0],i)
    print()

    #STUDENTS BETWEEN 25 AND 75 PERCENTILE
    max1=max(marks)
    min1=min(marks)
    diff=max1-min1
    m25th=diff*0.25
    m75th=diff*0.75
    print('Students between 25 and 75 percentile are...')
    for j in ascending:
        if(j>m25th and j<m75th):
            pos=marks.index(j)
            print(l[pos])
    print()

display_dash_board(students, marks)
```

```
The top 5 ranks students are...
student8 98
student10 80
student2 78
student5 48
student7 47

The least 5 ranks students are...
student3 12
student4 14
student9 22
student6 43
student1 45

Students between 25 and 75 percentile are...
['student9', 22]
['student6', 43]
['student1', 45]
['student7', 47]
['student5', 48]
```

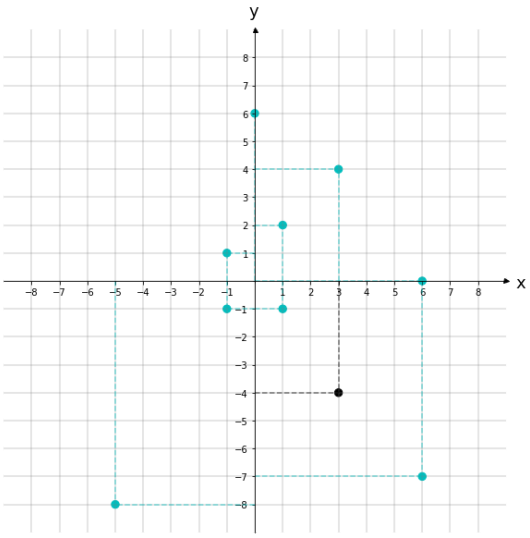
Q5: Find the closest points

Consider you have given n data points in the form of list of tuples like S=[(x1,y1),(x2,y2),(x3,y3),(x4,y4),(x5,y5),..,(xn,yn)] and a point P=(p,q)
Your task is to find 5 closest points(based on cosine distance) in S from P

Cosine distance between two points (x,y) and (p,q) is defind as $cos^{-1}(\frac{(x\cdot p+y\cdot q)}{\sqrt{(x^2+y^2)}\cdot\sqrt{(p^2+q^2)}})$

Ex:

S= [(1,2),(3,4),(-1,1),(6,-7),(0, 6),(-5,-8),(-1,-1),(6,0),(1,-1)]
P= (3,-4)



Output:
(6,-7)
(1,-1)
(6,0)
(-5,-8)
(-1,-1)

Hint - If you write the formula correctly you'll get the distance between points (6,-7) and (3,-4) = 0.065

In [100]:

```
#FINDING 5 CLOSEST POINTS TO A GIVEN POINT USING COSINE DISTANCE
import math

#FUNCTION TO FIND COSINE DISTANCE OF A GIVEN POINTS FROM A PERTICULAR POINT
def closest_points_to_p(S, P):
    cosine_dist=[]
    for x,y in S:
        p=P[0]
        q=P[1]
        Numerator=x*p+y*q
        Denominator=math.sqrt(x**2+y**2)*math.sqrt(p**2+q**2)
        cosine_dist.append(math.acos(Numerator/Denominator))    #use cosine distance formula
    zipped=list(zip(S,cosine_dist))                             #zipping two lists
    sorted_list=sorted(zipped, key=lambda a:a[1])               #sorting zipped list
    lst=[]
    for i in range(len(sorted_list[:5])):                       #append 5 closest points, to a given point, to a list
        lst.append(sorted_list[i][0])
    return lst


#DEFINING POINTS AND CALLING FUNCTION
S= [(1,2),(3,4),(-1,1),(6,-7),(0, 6),(-5,-8),(-1,-1),(6,0),(1,-1)]
P= (3,-4)
points = closest_points_to_p(S, P)
print("The 5 closed points are...")
print(points)                                                   #print the returned values.its list of tuples
```

```
The 5 closed points are...
[(6, -7), (1, -1), (6, 0), (-5, -8), (-1, -1)]
```

Q6: Find Which line separates oranges and apples

consider you have given two set of data points in the form of list of tuples like

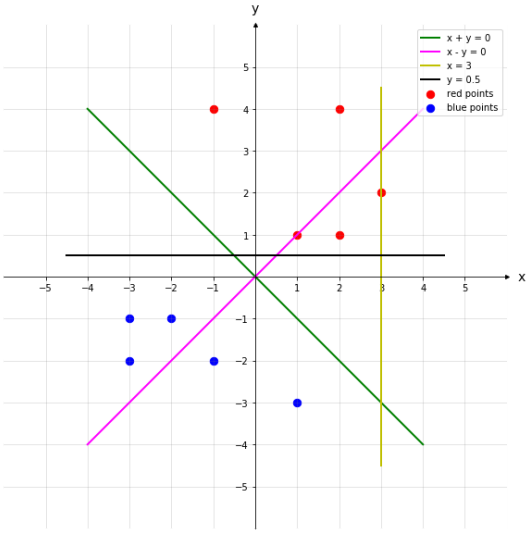
```
Red =[(R11,R12),(R21,R22),(R31,R32),(R41,R42),(R51,R52),..,(Rn1,Rn2)]
Blue=[(B11,B12),(B21,B22),(B31,B32),(B41,B42),(B51,B52),..,(Bm1,Bm2)]
```

and set of line equations(in the string formate, i.e list of strings)

```
Lines = [a1x+b1y+c1,a2x+b2y+c2,a3x+b3y+c3,a4x+b4y+c4,..,K lines]
Note: you need to string parsing here and get the coefficients of x,y and intercept
```

your task is to for each line that is given print "YES"/"NO", you will print yes, if all the red points are one side of the line and blue points are other side of the line, otherwise no

Ex:
Red= [(1,1),(2,1),(4,2),(2,4), (-1,4)]
Blue= [(-2,-1),(-1,-2),(-3,-2),(-3,-1),(1,-3)]
Lines=["1x+1y+0","1x-1y+0","1x+0y-3","0x+1y-0.5"]



Output:
YES
NO
NO
YES

In [107]:

```
#FIND WHICH LINE SEPERATE ORANGE AND APPLE
import math

def i_am_the_one(red,blue,line):

    for i in red:
        substitute=line.replace('x','*'+str(i[0]))
        substitute=substitute.replace('y','*'+str(i[1]))
        total=eval(substitute)
        if total>0:
            continue
        else:
            return 'NO'

    for i in blue:
        substitute=line.replace('x','*'+str(i[0]))
        substitute=substitute.replace('y','*'+str(i[1]))
        total=eval(substitute)
        if total<0:
            continue
        else:
            return 'NO'

    return "YES"

#DEFINING VALUES AND CALLING FUNCTION
Red= [(1,1),(2,1),(4,2),(2,4),(-1,4)]
Blue= [(-2,-1),(-1,-2),(-3,-2),(-3,-1),(1,-3)]
Lines=["1x+1y+0","1x-1y+0","1x+0y-3","0x+1y-0.5"]

for i in Lines:
    yes_or_no = i_am_the_one(Red, Blue, i)
    print((f"For the line {i}, the answer is..."),yes_or_no)
```

```
For the line 1x+1y+0, the answer is... YES
For the line 1x-1y+0, the answer is... NO
For the line 1x+0y-3, the answer is... NO
For the line 0x+1y-0.5, the answer is... YES
```

Q7: Filling the missing values in the specified formate

You will be given a string with digits and '_'(missing value) symbols you have to replace the '_' symbols as explained

```
Ex 1: _, _, _, 24 ==> 24/4, 24/4, 24/4, 24/4 i.e we. have distributed the 24 equally to all 4 places

Ex 2: 40, _, _, _, 60 ==> (60+40)/5,(60+40)/5,(60+40)/5,(60+40)/5,(60+40)/5 ==> 20, 20, 20, 20, 20 i.e. the sum of (60+40) is dis
tributed qually to all 5 places

Ex 3: 80, _, _, _, _  ==> 80/5,80/5,80/5,80/5,80/5 ==> 16, 16, 16, 16, 16 i.e. the 80 is distributed qually to all 5 missing valu
es that are right to it

Ex 4: _, _, 30, _, _, _, 50, _, _
==> we will fill the missing values from left to right
    a. first we will distribute the 30 to left two missing values (10, 10, 10, _, _, _, 50, _, _)
    b. now distribute the sum (10+50) missing values in between (10, 10, 12, 12, 12, 12, 12, _, _)
    c. now we will distribute 12 to right side missing values (10, 10, 12, 12, 12, 12, 4, 4, 4)
```

for a given string with comma seprate values, which will have both missing values numbers like ex: "_, _, x, _, _, _" you need fill the missing values

Q: your program reads a string like ex: "_, _, x, _, _, _" and returns the filled sequence

Ex:

```
Input1: "_,_,_,24"
Output1: 6,6,6,6

Input2: "40,_,_,_,60"
Output2: 20,20,20,20,20

Input3: "80,_,_,_,_"
Output3: 16,16,16,16,16

Input4: "_,_,30,_,_,_,50,_,_"
Output4: 10,10,12,12,12,12,4,4,4
```

In [23]:

```
#FILLING THE MISSING VALUE IN SPECIFIED FORMAT
def operations(a,x,y):
    if x==-1:
        value=float(a[y])/(y+1)
        for i in range(x+1,y+1):
            a[i]=value
    elif y==-1:
        value=float(a[x])/(len(a)-x)
        for i in range(x,len(a)):
            a[i]=value
    else:
        value=(float(a[x])+float(a[y]))/(y-x+1)
        for i in range(x,y+1):
            a[i]=value
    return a

#CREATING A NEW LIST WHICH CONTAINS INDEX FOR ORIGINAL VALUE PRESENT IN THE LIST
def curve_smoothing(string):
    a=string.replace("",'').split(',')
    b=[i for i, value in enumerate(a) if value!='_']
    if b[0]!=0:
        b=[-1]+b
    if b[-1]!=len(a)-1:
        b=b+[-1]
    for (x,y) in zip(b[:-1],b[1:]):
        operations(a,x,y)
    return a

#STRING INPUT AND FUNCTION CALLING
S= ["_,_,30,_,_,_,50,_,_" , "_,_,_,24", "40,_,_,_,60", "80,_,_,_,_" ]
for i in S:
    print(curve_smoothing(i))
```

```
[10.0, 10.0, 12.0, 12.0, 12.0, 12.0, 4.0, 4.0, 4.0]
[6.0, 6.0, 6.0, 6.0]
[20.0, 20.0, 20.0, 20.0, 20.0]
[16.0, 16.0, 16.0, 16.0, 16.0]
```

Q8: Filling the missing values in the specified formate

You will be given a list of lists, each sublist will be of length 2 i.e. [[x,y],[p,q],[l,m]..[r,s]] consider its like a martrix of n rows and two columns 1. the first column F will contain only 5 uniques values (F1, F2, F3, F4, F5) 2. the second column S will contain only 3 uniques values (S1, S2, S3)

```
your task is to find
a. Probability of P(F=F1|S==S1), P(F=F1|S==S2), P(F=F1|S==S3)
b. Probability of P(F=F2|S==S1), P(F=F2|S==S2), P(F=F2|S==S3)
c. Probability of P(F=F3|S==S1), P(F=F3|S==S2), P(F=F3|S==S3)
d. Probability of P(F=F4|S==S1), P(F=F4|S==S2), P(F=F4|S==S3)
e. Probability of P(F=F5|S==S1), P(F=F5|S==S2), P(F=F5|S==S3)
```

Ex:

```
[[F1,S1],[F2,S2],[F3,S3],[F1,S2],[F2,S3],[F3,S2],[F2,S1],[F4,S1],[F4,S3],[F5,S1]]

a. P(F=F1|S==S1)=1/4, P(F=F1|S==S2)=1/3, P(F=F1|S==S3)=0/3
b. P(F=F2|S==S1)=1/4, P(F=F2|S==S2)=1/3, P(F=F2|S==S3)=1/3
c. P(F=F3|S==S1)=0/4, P(F=F3|S==S2)=1/3, P(F=F3|S==S3)=1/3
d. P(F=F4|S==S1)=1/4, P(F=F4|S==S2)=0/3, P(F=F4|S==S3)=1/3
e. P(F=F5|S==S1)=1/4, P(F=F5|S==S2)=0/3, P(F=F5|S==S3)=0/3
```

In [60]:

```
#PROBABILITY
A =[['F1','S1'],['F2','S2'],['F3','S3'],['F1','S2'],['F2','S3'],['F3','S2'],['F2','S1'],['F4','S1'],['F4','S3'],['F5','S1']]
D1={}.fromkeys(['F1S1','F2S1','F3S1','F4S1','F5S1','F1S2','F2S2','F3S2','F4S2','F5S2','F1S3','F2S3','F3S3','F4S3','F5S3'],0)
D2={}.fromkeys(['S1','S2','S3'],0)

def compute_conditional_probabilites(A):
    for i in range(len(A)):
        v=A[i][0]+A[i][1]
        D1[v]+=1
        D2[A[i][1]]+=1
    #print(D1)
    #print(D2)
    a=int(input("Enter the F(x)..."))
    b=int(input("Enter the S(x)..."))
    if a>5 or b>3:
        print("Key error...")
        return 0
    else:
        print(f"The probability of F{a}S{b} is...",(D1[(f'F{a}S{b}')])/(D2[(f'S{b}')]))

compute_conditional_probabilites(A)
```

```
Enter the F(x)...2
Enter the S(x)...1
The probability of F2S1 is... 0.25
```

Q9: Given two sentances S1, S2

You will be given two sentances S1, S2 your task is to find

```
a. Number of common words between S1, S2
b. Words in S1 but not in S2
c. Words in S2 but not in S1
```

Ex:

```
S1= "the first column F will contain only 5 uniques values"
S2= "the second column S will contain only 3 uniques values"
Output:
a. 7
b. ['first','F','5']
c. ['second','S','3']
```

In [84]:

```
#SET OPERATIONS ON TWO STRINGS
def string_features(S1, S2):
    x=set(S1.split(" "))
    #print(x)
    y=set(S2.split(" "))
    #print(y)

    a=len(x.intersection(y))
    print(f"\nNumber of common words between S1 andS2 is {a}\n")
    b=list(x-y)
    print(f'Words in S1 but not in S2 {b}\n')
    c=list(y-x)
    print(f'Words in S2 but not in S1 {c}')

#INPUTTING STRING AND CALLING FUNCTION
S1=input("Enter the first string...\n")
S2=input("\nEnter the second string...\n")
#S1= "the first column F will contain only 5 uniques values"
#S2= "the second column S will contain only 3 uniques values"
string_features(S1, S2)
```

```
Enter the first string...
i am doing applied ai assignments

Enter the second string...
these assignments from applied ai is enhancing my logical and programming knowledge

Number of common words between S1 andS2 is 3

Words in S1 but not in S2 ['i', 'am', 'doing']

Words in S2 but not in S1 ['knowledge', 'from', 'enhancing', 'my', 'is', 'these', 'and', 'logical', 'programming']
```

Q10: Given two sentances S1, S2

You will be given a list of lists, each sublist will be of length 2 i.e. [[x,y],[p,q],[l,m]..[r,s]] consider its like a martrix of n rows and two columns

a. the first column Y will contain interger values
b. the second column Y_{score} will be having float values
Your task is to find the value of $f(Y, Y_{score}) = -1 * \frac{1}{n}\Sigma_{foreachY,Y_{score}pair}(Ylog10(Y_{score}) + (1-Y)log10(1-Y_{score}))$ here n is the number of rows in the matrix

```
Ex:
[[1, 0.4], [0, 0.5], [0, 0.9], [0, 0.3], [0, 0.6], [1, 0.1], [1, 0.9], [1, 0.8]]
output:
0.4243099
```

$\frac{-1}{8} \cdot ((1 \cdot log_{10}(0.4) + 0 \cdot log_{10}(0.6)) + (0 \cdot log_{10}(0.5) + 1 \cdot log_{10}(0.5)) + \ldots + (1 \cdot log_{10}(0.8) + 0 \cdot log_{10}(0.2)))$

In [14]:

```
#FINDING LOG LOSS
from math import log
def compute_log_loss(A):
    sum1=0
    for row in A:
        sum1+=(row[0]*log(row[1],10))+((1-row[0])*log((1-row[1]),10))
    loss=-1*(sum1/len(A))
    return loss

#INPUTTING LIST AND FUNCTION CALLING
n=int(input("Enter the no of element in the list..."))
A=[]
for i in range(n):
    ele=[float(input(f"\nEnter the Y{i}...")),float(input(f"Enter the Yscore{i}..."))]
    A.append(ele)

#A = [[1, 0.4], [0, 0.5], [0, 0.9], [0, 0.3], [0, 0.6], [1, 0.1], [1, 0.9], [1, 0.8]]
loss = compute_log_loss(A)
print('\nThe log loss is...',loss)
```

```
Enter the no of element in the list...4

Enter the Y0...1
Enter the Yscore0...0.4

Enter the Y1...0
Enter the Yscore1...0.5

Enter the Y2...0
Enter the Yscore2...0.9

Enter the Y3...0
Enter the Yscore3...0.3

The log loss is... 0.46346799108044046
```