Python: without numpy or sklearn

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]
             [18 24 30 36 42]]

Ex 3: A   = [[1 2]
             [3 4]]
      B   = [[1 4]
```

```
               [5 6]
               [7 8]
               [9 6]]
        A*B =Not possible
```

```
def matrix_mul(A, B):
    '''
       This function multiplies two matrix and returns the result.
    '''
    # write your code

    x=len(B[0])

    for e in B:
        if x==len(e): #checks if all the rows of B are of same length
            continue
        else:
            print("**Matrix multiplication not possible")
            return

    row_B=len(B)

    for y in A:
        if len(y)!=row_B: #checks if no of column of A is equal to no of rows in B
            print("Matrxi multiplication not possible")
            return
        else:
            continue

    final_list=[]
    for ele_A in A:
        l=[]
        for j in range(len(B[0])):
```

```
            Sum=0
            for i in range(row_B):
                Sum+=ele_A[i]*(B[i])[j]
            l.append(Sum)
        final_list.append(l)
    return(final_list)

#A   = [[1, 2] ,[3,4]]
#B   = [[1,2,3,4,5],[5,6,7,8,9]]

A   = [[1,3,4],[2,5,7],[5,9,6]]
B   = [[1,0,0],[0,1,0], [0,0,1]]
print("Matrix[A]*Matrix[B]  =",matrix_mul(A, B))
```

```
Matrix[A]*Matrix[B]  = [[1, 3, 4], [2, 5, 7], [5, 9, 6]]
```

Q2: 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)
```

```
from random import uniform

def pick_a_number_from_list(A):
```

```
    Sum=0
    for i in A:
        Sum+=i
    #print(Sum)
    ls_dash=[]
    for e in A:
        ls_dash.append(e/Sum)

    ls_tilda={}
    for i in range(0,len(ls_dash)):
        if i!=0:
            ls_tilda[A[i]]=ls_dash[i]+ls_tilda[A[i-1]]
        else:
          ls_tilda[A[i]]=ls_dash[i]

    r=uniform(0,1)

    for el in ls_tilda:
        if r<ls_tilda[el]:
            return el

def sampling_based_on_magnitued():
    '''
        Added the count of occurence of each number for the given input
        Please modify or comment print statements for different array input
    '''
    l=[]
    for i in range(1,100):
        number = pick_a_number_from_list([0,5,27,6,13,28,100,45,10,79])
        l.append(number)
        print(number)

    print("0 =",l.count(0))
    print("5 =",l.count(5))
    print("27 =",l.count(27))
    print("6 =",l.count(6))
    print("13 =",l.count(13))
    print("28 =",l.count(28))
```

```
        print("100 =",l.count(100))
        print("45 =",l.count(45))
        print("10 =",l.count(10))
        print("79 =",l.count(79))

    sampling_based_on_magnitued()
```

```
100
100
100
5
28
13
100
5
100
100
45
79
79
100
79
27
79
28
79
45
100
27
100
28
45
100
100
100
27
100
100
45
79
79
79
100
45
45
10
6
79
100
79
```

```
100
28
27
79
28
0 = 0
5 = 2
27 = 10
6 = 1
13 = 5
28 = 12
100 = 26
45 = 13
10 = 3
79 = 27
```

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: ####
```

```
def replace_digits(p):
    q=""
    for e in p:

        if (ord(e)>=48 and ord(e)<=57):
            q+="#"

    return(q)

replace_digits("#2a$#b%c%561#")
```

```
'####'
```

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, 35, 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, 35, 80]
a.
student8  98
student10 80
student2  78
```

```
student5  48
student7  47
b.
student3 12
student4 14
student9 35
student6 43
student1 45
c.
student9 35
student6 43
student1 45
student7 47
student5 48
```

```
def display_dash_board(students, marks):

    my_dict={}
    for i in range(0,len(marks)):
        my_dict[students[i]]=marks[i] #mapping students and marks inot a dictionary


    sorted_top={k: v for k, v in sorted(my_dict.items(),key=lambda x:x[1], reverse=True)}#dictionary sorting descending orde
    top_5_students=list(sorted_top.items())[:5] #top 5 students


    my_dict={k: v for k, v in sorted(sorted_top.items(),key=lambda x:x[1])}#dictionary sorting ascending order
    q=list(my_dict.items())
    least_5_students=list(my_dict.items())[:5]


   #d=len(q)-1
```

```
    x=25/100 #change this value for different percentile range
    y=75/100 #change this value for different percentile range
    min_marks=q[0][1]
    max_marks=q[len(q)-1][1]
    diff=max_marks-min_marks

    # calulating percentile position for xPercentile

    xPercentile=x*diff
    yPercentile=y*diff


    students_within_25_and_75=[]
    for element in q:
        if element[1]>xPercentile and element[1]<yPercentile:
            students_within_25_and_75.append(element)

    return top_5_students,least_5_students,students_within_25_and_75
    #print("25-75",students_within_25_and_75)


students=['student1','student2','student3','student4','student5','student6','student7','student8','student9','student10']
marks=[45, 78, 12, 14, 48, 43, 47, 98, 35, 80]

top_5_students, least_5_students, students_within_25_and_75=display_dash_board(students, marks)

print(" Top 5 students   ",top_5_students,"\n","Least 5 students ", least_5_students,"\n","Students within 25 and 75 percent
```

```
Top 5 students    [('student8', 98), ('student10', 80), ('student2', 78), ('student5', 48), ('student7', 47)]
Least 5 students  [('student3', 12), ('student4', 14), ('student9', 35), ('student6', 43), ('student1', 45)]
Students within 25 and 75 percentile [('student9', 35), ('student6', 43), ('student1', 45), ('student7', 47), ('studen
```

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)

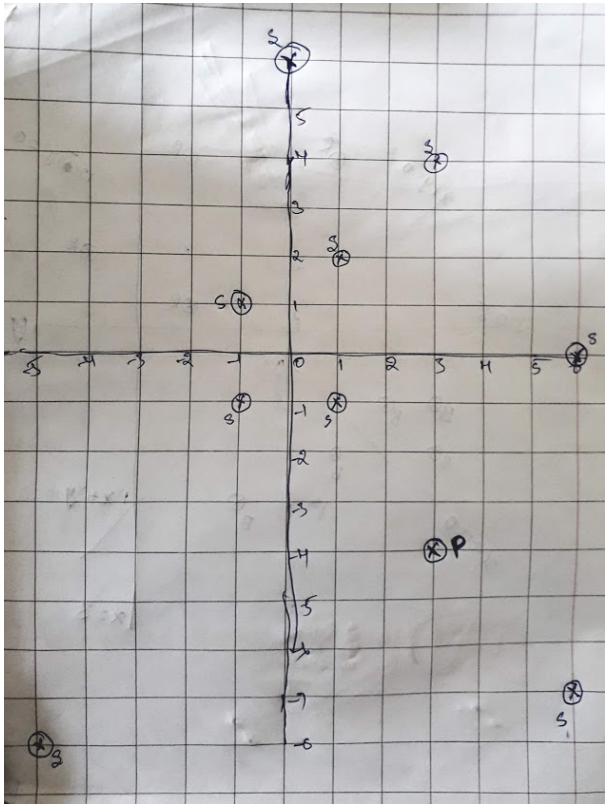
your task is to find 5 closest points(based on cosine distance) in S from P

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)
```

```
import math

# write your python code here
# you can take the above example as sample input for your program to test
# it should work for any general input try not to hard code for only given input examples
# you can free to change all these codes/structure


# here S is list of tuples and P is a tuple ot len=2
def closest_points_to_p(S, P):

    l=[]
    for e in S:
        l.append(math.acos((e[0]*P[0]+e[1]*P[1])/((math.sqrt(math.pow(e[0],2)+math.pow(e[1],2))) *(math.sqrt(math.pow(P[0],2

    closest_points_to_p=[]
    for q in zip(S,l):
        closest_points_to_p.append(q)

    closest_points_to_p.sort(key=lambda x:x[1])

    j=[]
    for e in closest_points_to_p:
        j.append(e[0])



    return j[:5]  # its list of tuples

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(points) #print the returned values
```

```
print(points) #print the returned values
```

```
[(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)]
```

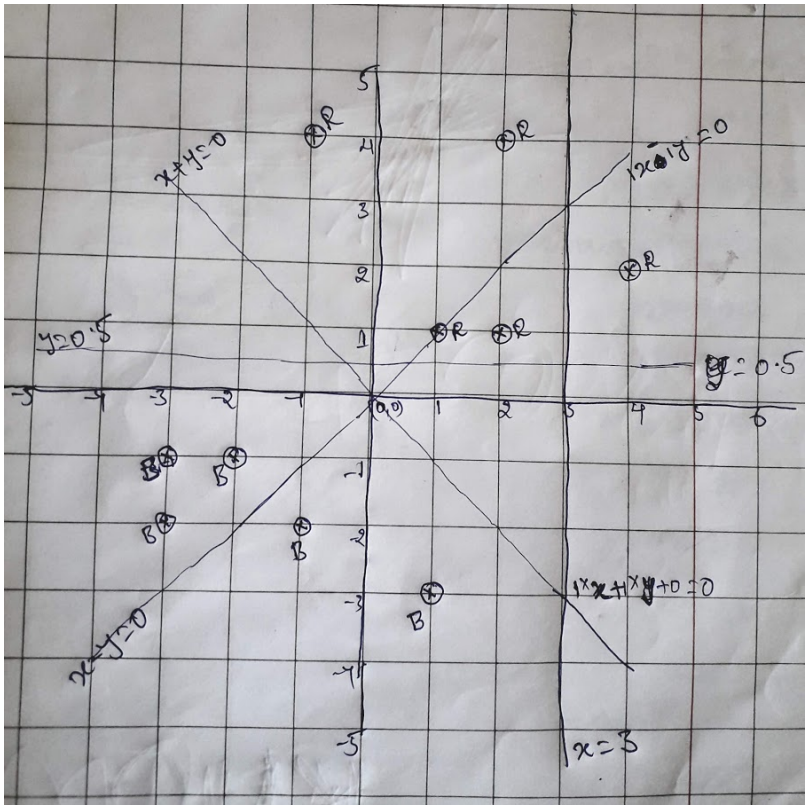
and set of line equations(in the string formate, i.e list of strings)

```
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
```

```
import math
import re


def i_am_the_one(red,blue,line):

    j=[]
    q=re.split('[x|y]', line)
    #print(q)
    for e in q:
        j.append(float(e))
    #print(j)

    li_red_pos=[]
    li_red_neg=[]
    for el in red:
        temp=j[0]*el[0]+j[1]*el[1]+j[2]
        #print(temp)
        if temp>0:
            li_red_pos.append(el)
        elif temp<0:
            li_red_neg.append(el)
        else:
            return "No"
    if (len(red)!=len(li_red_pos) and len(red)!=len(li_red_neg) ):
        return "No"
    else:
        if len(red)==len(li_red_pos):
            r_sign="+ve"
        else:
            r_sign="-ve"

    li_blue_pos=[]
    li_blue_neg=[]
    for el in blue:
        temp=j[0]*el[0]+j[1]*el[1]+j[2]
        #print(temp)
        if temp>0:
            li_blue_pos.append(el)
```

```
            elif temp<0:
                li_blue_neg.append(el)
            else:
                return "No"
        if (len(blue)!=len(li_blue_pos) and len(blue)!=len(li_blue_neg) ):
            return "No"

        else:
            if len(blue)==len(li_blue_pos):
                b_sign="+ve"
            else:
                b_sign="-ve"

        if r_sign!=b_sign:
            return "Yes"
        else:
            return "No"



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(yes_or_no) # the returned value
```

```
Yes
No
No
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)

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 missi

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
```

```
def curve_smoothing(S):

    n1=0
```

```
        n1=0
        n2=0
        x=0
        y=0
        numexist=0
        li=S.split(',')

        while '_' in li:
            count=0
            for i in range(x,len(li)):
                count+=1
                if li[i]!='_':
                    n2=li[i]
                    if numexist >0:
                        count+=1
                    temp=(n1+int(n2))/count

                    for e in range(y,i+1):
                        li[e]=int(temp)
                        x=i+1
                        y=i
                        n1=temp
                        numexist+=1
                    break
                elif li[i]=='_' and i==(len(li)-1):

                    c=n1/(count+1)

                    for j in range(y,i+1):
                        li[j]=int(c)

                    break


        return li

    S=["_,_,_,24","40,_,_,_,60","80,_,_,_,_","_,_,30,_,_,_,50,_,_"]

    for e in S:
        smoothed_values= curve_smoothing(e)
```

```
    smoothed_values= curve_smoothing(e)
    print(smoothed_values)
```

```
 [6, 6, 6, 6]
 [20, 20, 20, 20, 20]
 [16, 16, 16, 16, 16]
 [10, 10, 12, 12, 12, 12, 4, 4, 4]
```

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
```

```
def compute_conditional_probabilites(A,F,S):
    count_S=0
    count_F=0
```

```
    count_F=0
    for e in A:
        if e[1]==S:
            count_S+=1
        if e[0]==F and e[1]==S:
            count_F+=1

    return str(count_F)+"/"+str(count_S)


A = [['F1','S1'],['F2','S2'],['F3','S3'],['F1','S2'],['F2','S3'],['F3','S2'],['F2','S1'],['F4','S1'],['F4','S3'],['F5','S1']]

print("P(F=F1|S==S1) ",compute_conditional_probabilites(A,"F1","S1"))
print("P(F=F1|S==S2) ",compute_conditional_probabilites(A,"F1","S2"))
print("P(F=F1|S==S3) ",compute_conditional_probabilites(A,"F1","S3"))

print("="*25)

print("P(F=F2|S==S1) ",compute_conditional_probabilites(A,"F2","S1"))
print("P(F=F2|S==S2) ",compute_conditional_probabilites(A,"F2","S2"))
print("P(F=F2|S==S3) ",compute_conditional_probabilites(A,"F2","S3"))

print("="*25)

print("P(F=F3|S==S1) ",compute_conditional_probabilites(A,"F3","S1"))
print("P(F=F3|S==S2) ",compute_conditional_probabilites(A,"F3","S2"))
print("P(F=F3|S==S3) ",compute_conditional_probabilites(A,"F3","S3"))

print("="*25)

print("P(F=F4|S==S1) ",compute_conditional_probabilites(A,"F4","S1"))
print("P(F=F4|S==S2) ",compute_conditional_probabilites(A,"F4","S2"))
print("P(F=F4|S==S3) ",compute_conditional_probabilites(A,"F4","S3"))

print("="*25)

print("P(F=F5|S==S1) ",compute_conditional_probabilites(A,"F5","S1"))
print("P(F=F5|S==S2) ",compute_conditional_probabilites(A,"F5","S2"))
print("P(F=F5|S==S3) ",compute_conditional_probabilites(A,"F5","S3"))
```

```
P(F=F1|S==S1)  1/4
P(F=F1|S==S2)  1/3
P(F=F1|S==S3)  0/3
========================
P(F=F2|S==S1)  1/4
P(F=F2|S==S2)  1/3
P(F=F2|S==S3)  1/3
========================
P(F=F3|S==S1)  0/4
P(F=F3|S==S2)  1/3
P(F=F3|S==S3)  1/3
========================
P(F=F4|S==S1)  1/4
P(F=F4|S==S2)  0/3
P(F=F4|S==S3)  1/3
========================
P(F=F5|S==S1)  1/4
P(F=F5|S==S2)  0/3
P(F=F5|S==S3)  0/3
```

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']

```
def string_features(S1, S2):
    # your code
    J1=S1.split()
    J2=S2.split()
    b=[]
    c=[]
    a=0


    for e in J1:

        if e not in J2:
            b.append(e)
        else:
            a+=1

    for e in J2:
        if e not in J1:
            c.append(e)



    return a, b, c

S1= "the first column F will contain only 5 uniques values"
S2= "the second column S will contain only 3 uniques values"
a,b,c = string_features(S1, S2)
print(a,b,c)
```

```
7 ['first', 'F', '5'] ['second', 'S', '3']
```

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
```

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

```
def compute_log_loss(A):

    summation=0
    for e in A:
        summation+=((e[0]*math.log(e[1],10))+(1-e[0])*math.log((1-e[1]),10))

    loss=((-1)*summation)/len(A)
    return loss

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(loss)
```

```
0.42430993457031635
```