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]
             [23 30 36 42 51]]

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

```
1 # write your python code here
2 # you can take the above example as sample input for your program to test
3 # it should work for any general input try not to hard code for only given input exampl
4
5
6 # you can free to change all these codes/structure
7 # here A and B are list of lists
8 def matrix_mul(A, B):
9     resultOfMultiplication = [[0, 0, 0, 0],
```

```
        [0, 0, 0, 0],
        [0, 0, 0, 0]];
    for i in range(len(A)):
        for j in range(len(B[0])):
            for k in range(len(B)):
                resultOfMultiplication[i][j] += A[i][k] * B[k][j]
    return(resultOfMultiplication)
A   = [[1, 2], [3, 4]]
# take a 3x4 matrix
B   = [[1 ,2 ,3, 4 ,5],[5 ,6 ,7 ,8 ,9]]
matrix_mul(B,A)
```

```
[[7, 10, 0, 0], [23, 34, 0, 0], [0, 0, 0, 0]]
```

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)
```

```
import random
# 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 exampl


# you can free to change all these codes/structure
values=[0, 5 ,27, 6, 13, 28, 100, 45, 10,79]

def pick_a_number_from_list(A):
    sum=0;
    #first finding the sum of the array elements in the list
    for i in A:
        sum+=i;
    x=0
    proportionaity=[]
    for i in A:
        proportionaity.append(x+i/sum)
        x=x+i/sum;
    #proportionaity contains cumulative sum
    r=random.uniform(0,1)
    for i in range (0,len(proportionaity)):
        # print(i)
        if r<=proportionaity[i]:
            return A[i]
```

```
26
27 def sampling_based_on_magnitued():
28     for i in range(1,100):
29         number = pick_a_number_from_list(values)
30         print(number)
31
32 sampling_based_on_magnitued()
```

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

```
28
27
100
79
45
100
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: ####
```

```
import re
# 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 exampl

# you can free to change all these codes/structure
# String: it will be the input to your program
def replace_digits(String):
    # write your code
    c=''
    for i in range(len(String)):
      if String[i].isdigit():
        c+='#'
    return(c) # modified string which is after replacing the # with digits

replace_digits('abcd1234')
```

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

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','
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
```

```
# 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 exampl
import math
dicti={}
def display_dash_board(students, marks):

    for i in range(len(marks)):
        dicti.setdefault(marks[i],[])
        dicti[marks[i]].append(students[i]);
    top_5 = []
    cnt=1;
    for i in sorted(dicti.keys(),reverse=True):
      if cnt <= 5:
        top_5.append(i)
        cnt+=1
      else:
        break
    print("Showing below top 5 Students");
    for i in range(0,len(top_5)):
      print(dicti[top_5[i]],top_5[i])
    least_5 = []
    cnt=1;
```

```
    for i in sorted(dicti.keys()):
      if cnt <= 5:
        least_5.append(i)
        cnt+=1
      else:
        break
    print("Showing below least 5 Students")
    for i in range(0, len(least_5)):
      print(dicti[least_5[i]],least_5[i])
    mark=sorted(dicti.keys());
    twentyfifth = math.floor(len(mark) / 4)
    seventyfifth = math.floor(3 * len(mark) / 4)
    middle = mark[twentyfifth : seventyfifth]
    print("Showing below the list of students between the 25th and 75th percentile");
    for i in range(0, len(middle)):
      print(dicti[middle[i]],middle[i])
Students=['student1','student2','student3','student4','student5','student6','student7',
Marks = [45, 78, 12, 14, 48, 43, 47, 98, 35, 80]

display_dash_board(Students, Marks)
```

```
Showing below top 5 Students
['student8'] 98
['student10'] 80
['student2'] 78
['student5'] 48
['student7'] 47
Showing below least 5 Students
['student3'] 12
['student4'] 14
['student9'] 35
['student6'] 43
['student1'] 45
Showing below the list of students between the 25th and 75th percentile
['student9'] 35
['student6'] 43
['student1'] 45
['student7'] 47
['student5'] 48
```

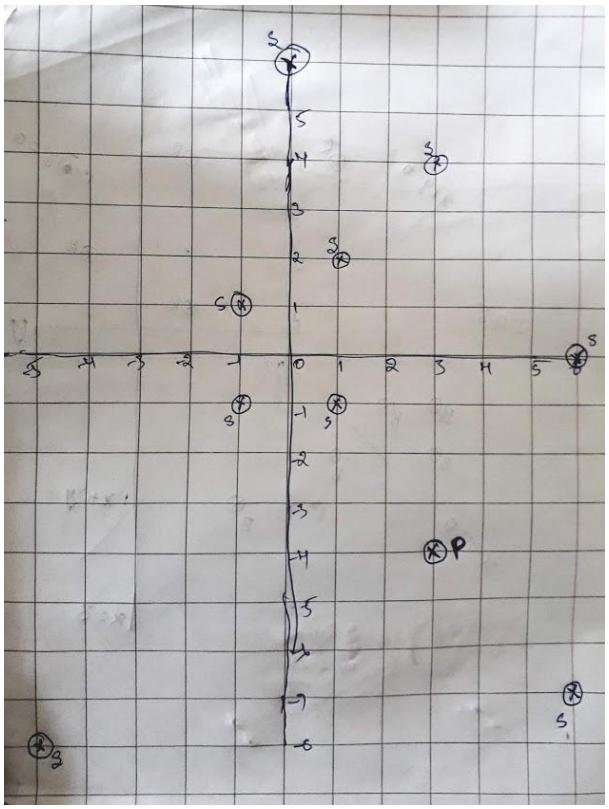
Q5: Find the closest points

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 $cos^{-1}\left(\frac{(x\cdot p+y\cdot q)}{\sqrt{(x^2+y^2)}\cdot\sqrt{(p^2+q^2)}}\right)$

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 exampl
# you can free to change all these codes/structure

# here S is list of tuples and P is a tuple ot len=2
def acosinedist(X,P):
    num = X[0] * P[0] + X[1] * P[1];
    den = math.sqrt(X[0] * X[0] + X[1] * X[1]) * math.sqrt(P[0] * P[0] + P[1] * P[1])
    return math.acos(num/den)
def closest_points_to_p(S, P):
    S.sort(key=lambda x: acosinedist(x,P))
    closest_points_to_p =S[:5];
    return closest_points_to_p  # 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
```

```
[(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

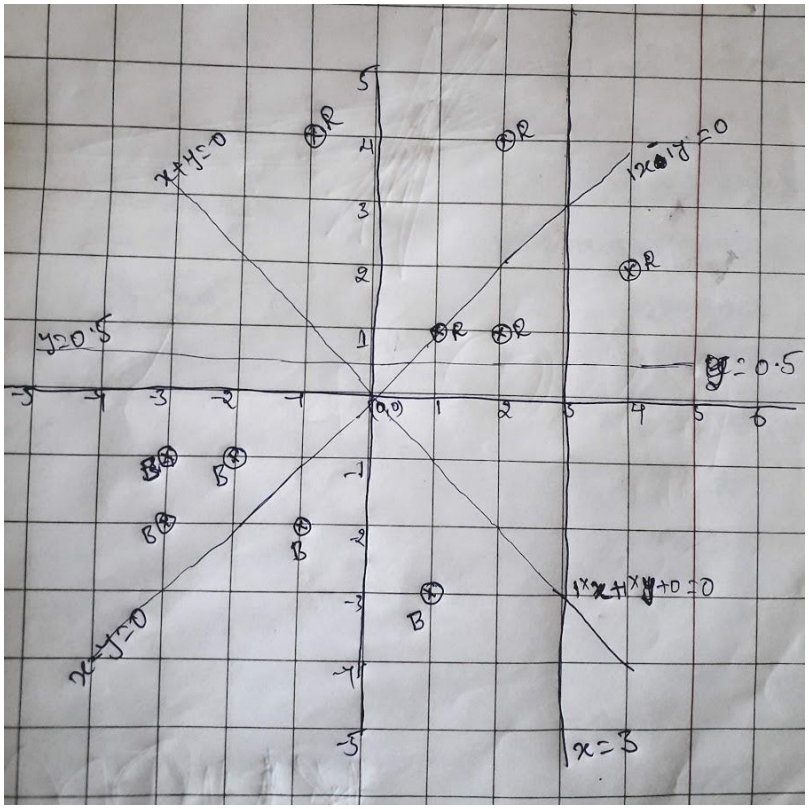
```
Red =[(R11,R12),(R21,R22),(R31,R32),(R41,R42),(R51,R52),..,(Rn1,Rn2)]
Blue=[(B11,B12),(B21,B22),(B31,B32),(B41,B42),(B51,B52),..,(Bm1,Bm2)]
```

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
```

```
1 import re
```

```
def ArePointsSameSideOfLine(a, b, c, x1, y1, x2, y2):
    line1 = 0 # Variable to store a * x1 + b * y1 - c
    line2 = 0 # Variable to store a * x2 + b * y2 - c
    line1 = a * x1 + b * y1 - c
    line2 = a * x2 + b * y2 - c

# If line1 and line2 have same sign then return true
    if ((line1 * line2) > 0):
        return True
    return False
def print_YES_NO(Red,Blue,lines):
    for i in lines:
        z = re.findall(r'[\d\.\+\-]+', i)
        flag=True
        for i in range(0,len(Red)-1):
          flag = flag and ArePointsSameSideOfLine(float(z[0]), float(z[1]), float(z[2])
        flagblue=True
        flagblue=flagblue and not(ArePointsSameSideOfLine(float(z[0]), float(z[1]), flo
        for i in range(0,len(Blue)-1):
          flagblue = flagblue and ArePointsSameSideOfLine(float(z[0]), float(z[1]), flo
        if(flagblue and flag):
          print('YES')
        else:
          print('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","0x+1y-0.5", "1x+0y-3" ]
print_YES_NO(Red,Blue,lines);
```

```
YES
NO
YES
NO
```

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

Ex 2: 40, _, _, _, 60 ==> (60+40)/5,(60+40)/5,(60+40)/5,(60+40)/5,(60+40)/5 ==> 20, 20,

Ex 3: 80, _, _, _, _  ==> 80/5,80/5,80/5,80/5,80/5 ==> 16, 16, 16, 16, 16 i.e. the 80 is

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, _, _, _,
    b. now distribute the sum (10+50) missing values in between (10, 10, 12, 12, 12, 12,
    c. now we will distribute 12 to right side missing values (10, 10, 12, 12, 12, 12, 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
```

```
# 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 string

def curve_smoothing(string):
    x = string.split(",")
    lefti=0
    righti=0
    for righti in range(1,len(x)):
      if x[righti] != '_':
        if x[lefti] != '_':
          avg=(int(x[lefti])+int(x[righti]))/(righti-lefti+1)
          for j in range(lefti,righti+1):
            x[j]=avg
          lefti=righti
        else:
          avg=(int(x[righti]))/(righti-lefti+1)
          for j in range(lefti,righti+1):
             x[j]=avg
          lefti=righti
    if x[righti]=='_':
      avg=int(x[lefti])/(righti-lefti+1)
      for i in range(lefti,righti+1):
        x[i]=avg
    for i in range(0,len(x)):
      x[i]=int(x[i])
    return x
S= "200,_,_,_,_,5,_"
print(curve_smoothing(S))
```

```
[34, 34, 34, 34, 34, 17, 17]
```

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
```

```
# 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 string


# you can free to change all these codes/structure

dicti1 = {
'F1S1': 0,
'F5S2': 0,
'F1S3': 0,
'F1S2': 0,
'F2S2': 0,
'F2S1': 0,
'F3S1': 0,
'F3S2': 0,
'F2S3': 0,
'F3S3': 0,
'F4S1': 0,
'F5S1': 0,
'F4S2': 0,
'F4S3': 0,
'F5S3': 0,
}

```

```
dicti2 = {
'S1': 0,
'S2': 0,
'S3': 0
}


def compute_conditional_probabilites(A):
    for i in range(len(A)):
       k = A[i][0] + A[i][1]
       dicti1[k] += 1
       dicti2[A[i][1]] += 1
A = [['F1','S1'],['F2','S2'],['F3','S3'],['F1','S2'],['F2','S3'],['F3','S2'],['F2','S1'
compute_conditional_probabilites(A)
print("F1")
print('Probability of P(F=F1|S==S1)', (dicti1['F1S1']/dicti2['S1']))
print('Probability of P(F=F1|S==S2)', (dicti1['F1S2']/dicti2['S2']))
print('Probability of P(F=F1|S==S3)', (dicti1['F1S3']/dicti2['S3']))
print("F2")
print('Probability of P(F=F2|S==S1)', (dicti1['F2S1']/dicti2['S1']))
print('Probability of P(F=F2|S==S2)', (dicti1['F2S2']/dicti2['S2']))
print('Probability of P(F=F2|S==S3)', (dicti1['F2S3']/dicti2['S3']))
print("F3")
print('Probability of P(F=F3|S==S1)', (dicti1['F3S1']/dicti2['S1']))
print('Probability of P(F=F3|S==S2)', (dicti1['F3S2']/dicti2['S2']))
print('Probability of P(F=F3|S==S3)', (dicti1['F3S3']/dicti2['S3']))
print("F4")
print('Probability of P(F=F4|S==S1)', (dicti1['F4S1']/dicti2['S1']))
print('Probability of P(F=F4|S==S2)', (dicti1['F4S2']/dicti2['S2']))
print('Probability of P(F=F4|S==S3)', (dicti1['F4S3']/dicti2['S3']))
print("F5")
print('Probability of P(F=F5|S==S1)', (dicti1['F5S1']/dicti2['S1']))
print('Probability of P(F=F5|S==S2)', (dicti1['F5S2']/dicti2['S2']))
print('Probability of P(F=F5|S==S3)', (dicti1['F5S3']/dicti2['S3']))
compute_conditional_probabilites(A)
```

```
F1
Probability of P(F=F1|S==S1) 0.25
Probability of P(F=F1|S==S2) 0.3333333333333333
Probability of P(F=F1|S==S3) 0.0
F2
Probability of P(F=F2|S==S1) 0.25
Probability of P(F=F2|S==S2) 0.3333333333333333
Probability of P(F=F2|S==S3) 0.3333333333333333
F3
Probability of P(F=F3|S==S1) 0.0
Probability of P(F=F3|S==S2) 0.3333333333333333
Probability of P(F=F3|S==S3) 0.3333333333333333
F4
Probability of P(F=F4|S==S1) 0.25
Probability of P(F=F4|S==S2) 0.0
Probability of P(F=F4|S==S3) 0.3333333333333333
F5
Probability of P(F=F5|S==S1) 0.25
Probability of P(F=F5|S==S2) 0.0
Probability of P(F=F5|S==S3) 0.0
```

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']
```

```
# 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 string

# you can free to change all these codes/structure


#using dictionnary because dictionary is ordered and in list ,it takes linear time to s
def string_features(S1, S2):
  x=S1.split(" ");
  dicti1={}
  for i in range(0,len(x)):
    dicti1[x[i]]=1
  dicti12={}
  y=S2.split(" ");
  for i in range(0,len(x)):
    dicti2[y[i]]=1
  common=[]
  for i in range(0,len(y)):
    if y[i] in dicti1.keys():
      common.append(y[i])
  print("common",common)
  S1NotS2=[]
  for i in range(0,len(x)):
    if x[i] not in dicti2.keys():
      S1NotS2.append(x[i])
  print(S1NotS2)
  S2NotS1=[]
  for i in range(0,len(y)):
    if y[i] not in dicti1.keys():
      S2NotS1.append(y[i])
  print(S2NotS1)
  return len(common),S1NotS2,S2NotS1
```

```
34
35 S1= "the first column F will contain only 5 uniques values"
36 S2= "the second column S will contain only 3 uniques values"
37 a,b,c = string_features(S1, S2)
```

```
common ['the', 'column', 'will', 'contain', 'only', 'uniques', 'values']
['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
$f(Y, Y_{score}) = -1 * \frac{1}{n}\Sigma_{foreach Y, Y_{score} pair}(Y log10(Y_{score}) + (1-Y) log10(1-Y_{score}))$
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
```

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

```
1 # write your python code here
2 # you can take the above example as sample input for your program to test
3 # it should work for any general input try not to hard code for only given input string
4
5
6 # you can free to change all these codes/structure
7 import math
8 def compute_log_loss(A):
9     Y = len(A)
10    l = 0
11    for i in range(Y):
12        l += A[i][0]*math.log10(A[i][1]) + (1-A[i][0])*math.log10(1-A[i][1])
13    loss=(-1*l)/Y
14    return loss
15
16 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]]
17 loss = compute_log_loss(A)
18 print(loss)
```

```
0.42430993457031635
```

✓ 0s completed at 10:51 PM

Could not connect to the reCAPTCHA service. Please check your internet connection and reload to get a reCAPTCHA challenge.