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
```

```
# 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 A and B are list of lists
def matrix_mul(A, B):
    #result=[[0,0,0],[0,0,0],[0,0,0]]
    result=[[0 for i in range(len(B[0]))] for j in range(len(A))]
    if(len(A[0])==len(B)):

      for i in range(len(A)):
          for j in range(len(B[0])):
             for k in range(len(B)):
                result[i][j]+=A[i][k]*B[k][j]
      for r in result:
        print (r)
    else:
      return("NOT POSSIBLE")
A   = [[1,3,4],[2,5,8],[5,9,6]]
B   = [[1,0,0],[0,1,0],[0,0,1]]
matrix_mul(A, B)
```

```
     [1, 3, 4]
     [2, 5, 8]
     [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
# 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
def pick_a_number_from_list(A):
    # your code here for picking an element from with the probability propotional to its m
    #.
    #.
    #.
    return #selected_random_number

def sampling_based_on_magnitued():
    for i in range(1,100):
        number = pick_a_number_from_list(A)
        print(number)

sampling_based_on_magnitued()
```

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 examples

# you can free to change all these codes/structure
# String: it will be the input to your program
def replace_digits(String):
    l=len(String)
    res=""
    for i in range (0,l):
        c=String[i]
        x=ord(c)
        if(x>47 and x<58):
            res=res+"#"
    return(res)
replace_digits(String)
```

```
'#'
```

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 examples

# you can free to change all these codes/structure
def display_dash_board(students, marks):
    l1=len(students)
    # Creating a list madeup of elements of marks arranged in Descending order
    markdesc=[]
    for i in range(l1):
        markdesc.append(marks[i])
    markdesc.sort(reverse=True)
    # Creating a list madeup of elements of marks arranged in Ascending order
    markasc=[]
    for i in range(l1):
        markasc.append(marks[i])
    markasc.sort()
    print("a.")
    for i in range(0,5):
        for j in range(0,l1):
            if (markdesc[i]==marks[j]):
                print(students[j]," ",markdesc[i])
    print("b.")
    for i in range(0,5):
        for j in range(0,l1):
            if (markasc[i]==marks[j]):
                print(students[j]," ",markasc[i])
    print("c.")
    for i in range(0,l1):
        if (markasc[i]>25 and markasc[i]<75):
            for j in range(0,l1):
                if(markasc[i]==marks[j]):
                    print(students[j]," ",markasc[i])
    return
students=['student1','student2','student3','student4','student5','student6','student7','st
marks = [45, 78, 12, 14, 48, 43, 47, 98, 35, 80]
display_dash_board(students, marks)
```

```
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
```

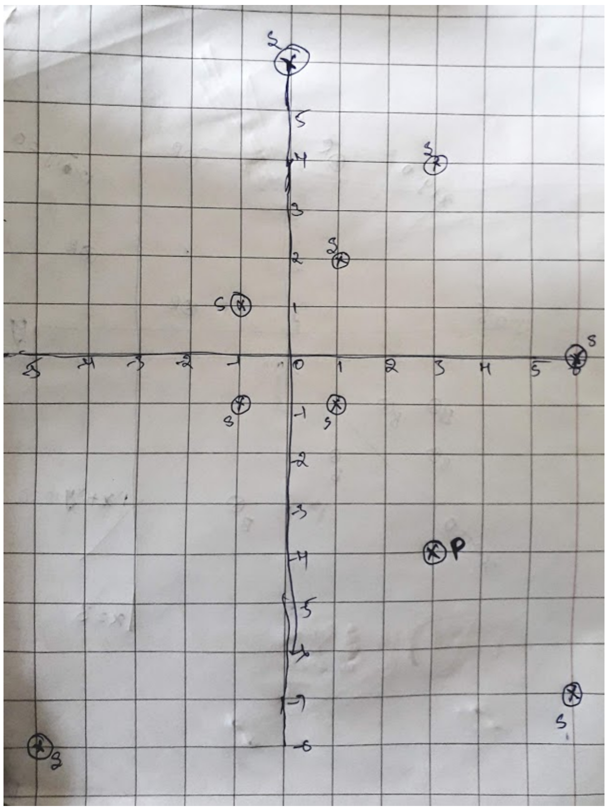
Q5: Find the closest points

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

def closest_points_to_p(S, P):
  SList=list(S)
  PList=list(P)
  dist=[]
  for i in range(0,len(SList)):
    numo=SList[i][0]*PList[0]+SList[i][1]*PList[1]
    deno1=math.sqrt(SList[i][0]*SList[i][0]+SList[i][1]*SList[i][1])
    deno2=math.sqrt(PList[0]*PList[0]+PList[1]*PList[1])
    deno=deno1*deno2
    d=math.acos(numo/deno)
    dist.append(d)
  distSort=[]
  for x in dist:
    distSort.append(x)
  distSort.sort()
  for i in range(0,5):
    for j in range(0,len(dist)):
        if (distSort[i]==dist[j]):
            print(tuple(SList[j]))


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

```
P= (3,-4)
closest_points_to_p(S, P)
```

```
    (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

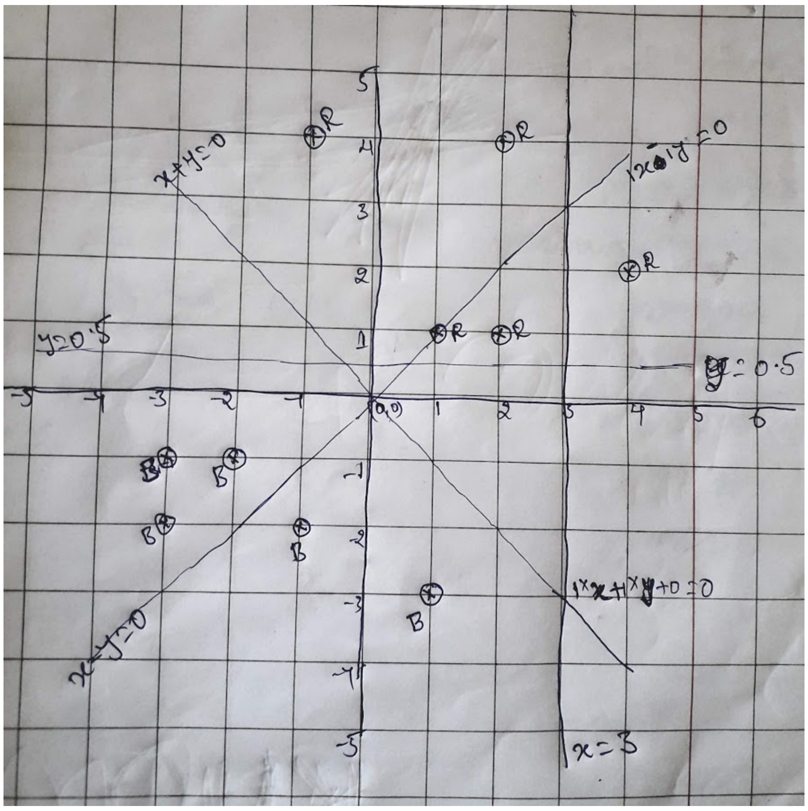
```
Red =[(R11,R12),(R21,R22),(R31,R32),(R41,R42),(R51,R52),..,(Rn1,Rn2)]
Blue=[(B11,B12),(B21,B22),(B31,B32),(B41,B42),(B51,B52),..,(Bm1,Bm2)]
```

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
# 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 strings


# you can free to change all these codes/structure
def i_am_the_one(red,blue,line):
    l=[]
    for i in range(len(line)):
        res=re.split('x|y', line[i]) # "Abhisheak is good boy" split(" ") -> ["Abhisheak",
        l.append(res)
#print(l)  # l=[['1', '+1', '+0'], ['1', '-1', '+0'], ['1', '+0', '-3'], ['0', '+1', '-0.5
    lineParse=[]
    for i in range(len(l)):
        #print("i=",i)
        #print("l[i]=",l[i])
        n=[]
        x=float(l[i][0])
        n.append(x)
        y=float(l[i][1])
        n.append(y)
        c=float(l[i][2])
        n.append(c)
        #print("X=",x, " y=",y," c=",c," n=",n)
        lineParse.append(n)
        #print("lineParse=",lineParse)
        #print(lineParse) # lineParse=[[1.0, 1.0, 0.0], [1.0, -1.0, 0.0], [1.0, 0.0, -3.0]

    for i in range(len(lineParse)):
        flagline=0
        flagred=0
        flagblue=0
        r1x=red[0][0] # l1=1,1,0   l2=1,-1,0   l3=1,0,-3    l4-> 0,1,-5
        r1y=red[0][1]
        b1x=blue[0][0]
        b1y=blue[0][1]
        r1=r1x*lineParse[i][0]+r1y*lineParse[i][1]+lineParse[i][2] # l1 -> 2    l2 ->0
        b1=b1x*lineParse[i][0]+b1y*lineParse[i][1]+lineParse[i][2] # l1 -> -3   l2 ->-1
        if(r1>0):
            flagred=1
        elif(r1<0):
            flagred=-1
        if(b1>0):
            flagblue=1
        elif(bl<0):
            flagblue=-1

    #l1 fr->1 fb->-1 ->proceed
    #l2 fr->0 fb->-1 -> fail: any one flag->0 or two flags are equal
    #l3 fr->-1 fb->1 ->proceed
    #l4 fr->1 fb->-1 ->proceed

    #flag=red -> stop
    #flag g-> go
    #flag y-> watch

        if ((flagred==flagblue) or (flagred==0 or flagblue==0)):
            flagline=1
        #print("first if")
        else:
            for j in range(1,len(red)):
                flagred2=0
                rx=red[j][0]
                ry=red[j][1]
                r=rx*lineParse[i][0]+ry*lineParse[i][1]+lineParse[i][2]
                if(r>0):
                    flagred2=1
                elif(r<0):
                    flagred2=-1
                if(flagred2!=flagred):
                    flagline=1
                    #print("Second red if")
                    break
            for j in range(1,len(blue)):
                flagblue2=0
                bx=blue[j][0]
                by=blue[j][1]
                b=bx*lineParse[i][0]+by*lineParse[i][1]+lineParse[i][2]
                if(b>0):
                    flagblue2=1
                elif(b<0):
                    flagblue2=-1
                if(flagblue2!=flagblue):
                    flagline=1
                    #print("Second blue if")
                    break
        if(flagline==0):
            print("YES")
        else:
            print("No")


    return

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"]

i_am_the_one(Red, Blue, Lines)
```

```
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

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 strings


# you can free to change all these codes/structure
# you can free to change all these codes/structure
def curve_smoothing(string):
    Str=string.split(",")
    #print(Str)
    Str1=[]
    for i in Str:
        if(i=='_'):
            Str1.append(0)
        else:
            Str1.append(int(i))
    #print(Str1)
    res=[0]*len(Str1)
    sum=0
    k=0
    for i in range(0,len(Str1)):
        #print("i=",i," Str1[",i,"]=",Str1[i])
        if(Str1[i]>0):
            sum=sum+Str1[i]
            l=i-k
            #print("l=",l)
            #print("k=",k)
            for j in range(k,i+1):
                #print("j=",j)
                res[j]=(sum//(l+1))
                #print("res=",res)
            k=i
            #print("k=",k)
            sum=res[i]
            #print("sum=",sum)
            i=i-1
        if(i==len(Str1)-1 and sum>0):
            sum=sum+Str1[i]
            l=i-k
            #print("l=",l)
            #print("k=",k)
            for j in range(k,i+1):
                #print("j=",j)
                res[j]=(sum//(l+1))
                #print("res=",res)
            k=i
            #print("k=",k)
            sum=res[i]
            #print("sum=",sum)
            i=i-1
    #print("res=",res)
    result=""
    for i in range(0,len(res)):
        result=result+str(res[i])+","
    print(result.strip(","))

    return

S=  "_,_,30,_,_,_,50,_,_"
curve_smoothing(S)
```

```
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
```

```
# 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 strings
import math as mt
dictionary1 = {
'F1S1': 0,
'F2S1': 0,
'F3S1': 0,
'F4S1': 0,
'F5S1': 0,
'F1S2': 0,
'F2S2': 0,
'F3S2': 0,
'F4S2': 0,
'F5S2': 0,
'F1S3': 0,
'F2S3': 0,
'F3S3': 0,
'F4S3': 0,
'F5S3': 0,
}

dictionary2 = {
'S1': 0,
'S3': 0,
}


# you can free to change all these codes/structure
def compute_conditional_probabilites(A):
   for i in range(len(A)):
        for j in range(i+1,len(A)):
            c=1
            k = A[i][0] + A[i][1]
            dictionary1[k] = c

            if A[i][0]+A[i][1] == A[j][0]+A[j][1]:
                k = A[i][0] + A[i][1]
                c+=1
                dictionary1[k] = c
        dictionary2[A[i][1]] += 1
   k = A[len(A)-1][0] + A[len(A)-1][1]
   dictionary1[k] = 1

   return dictionary1,dictionary2

    # print the output as per the instructions
A = [['F1','S1'],['F2','S2'],['F3','S3'],['F1','S2'],['F2','S3'],['F3','S2'],['F2','S1'],[
d1,d2 = compute_conditional_probabilites(A)
print(d1,d2)

for key,value in dictionary1.items():
    if 'S1' in key:
        x=str(value/dictionary2['S1'])
        print('P(F={})'.format(key[:2])+'/S=={})= '.format(key[2:])+ x)
    if 'S2' in key:
        x=str(value/dictionary2['S2'])
        print('P(F={})'.format(key[:2])+'/S=={})= '.format(key[2:])+ x)
    if 'S3' in key:
        x=str(value/dictionary2['S3'])
        print('P(F={})'.format(key[:2])+'/S=={})= '.format(key[2:])+ x)
```

```
{'F1S1': 1, 'F2S1': 1, 'F3S1': 0, 'F4S1': 1, 'F5S1': 1, 'F1S2': 1, 'F2S2': 1, 'F3S2'
P(F=F1)/S==S1)= 0.25
P(F=F2)/S==S1)= 0.25
P(F=F3)/S==S1)= 0.0
P(F=F4)/S==S1)= 0.25
P(F=F5)/S==S1)= 0.25
P(F=F1)/S==S2)= 0.3333333333333333
P(F=F2)/S==S2)= 0.3333333333333333
P(F=F3)/S==S2)= 0.3333333333333333
P(F=F4)/S==S2)= 0.0
P(F=F5)/S==S2)= 0.0
P(F=F1)/S==S3)= 0.0
P(F=F2)/S==S3)= 0.3333333333333333
P(F=F3)/S==S3)= 0.3333333333333333
```

```
P(F=F4)/S==S3)= 0.3333333333333333
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 strings

# you can free to change all these codes/structure
def string_features(S1, S2):
    State1=S1.split(" ")
    State2=S2.split(" ")
    common=0
    for i in State1:
        for j in State2:
            if(i.lower()==j.lower()):
                common=common+1
    print(common)
    first=[]
    for i in State1:
        flag=0
        for j in State2:
            if(i.lower()==j.lower()):
                flag=1
        if(flag==0):
            first.append(i)
    second=[]
    for i in State2:
        flag=0
        for j in State1:
            if(i.lower()==j.lower()):
                flag=1
        if(flag==0):
```

```
        if(flag==0):
            second.append(i)
    print(first)
    print(second)

    return

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)
```

```
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
```

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

```
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 strings


# you can free to change all these codes/structure
def compute_log_loss(A):
  summ=0
  for i in range(len(A)):
      #x=math.log10(A[i][1])
      #y=math.log10((1-A[i][1]))
      summ=summ+(A[i][0]*math.log10(A[i][1]))+((1-A[i][0])*math.log10((1-A[i][1])))
      #print("i=",i," summ=",summ)
  result=(summ/len(A))*-1
  return result
```

```
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
```

New Section

✓ 0s completed at 8:57 PM ● ✕

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