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
```

In [85]:

```
# 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 A and B are list of lists

def matrix_mul(A, B):
    Final_result =[]

    if(len(A[0]) == len(B)):
        print("Matrix Multiplication possible!")

        for i in range(len(A)):  #2 To be done twice , 2 lists to be created and append
            temp_list = [] #initializing empty list for every iteration
            for k in range(len(B[i])): # 5 iterations for every list
                temp = 0
                for j in range(len(A)):  # 2 Multiplications , adding to temp for 2 ite
                    temp = temp + (A[i][j] * B[j][k])
                temp_list.append(temp)
```

```
            Final_result.append(temp_list)

    else:
        print("Matrix Multiplication not possible!")

    return (Final_result)
A   = [[1,2],
       [3,4]]

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

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

# A   = [[1, 3, 4],
#        [2, 5, 7],
#        [5, 9, 6]]
# B   = [[1, 0, 0],
#        [0, 1, 0],
#        [0, 0, 1]]


matrix_mul(A, B)
```

```
Matrix Multiplication possible!
```

Out[85]:
```
[[11, 14, 17, 20, 23], [23, 30, 37, 44, 51]]
```

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)
```

In [24]:
```
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 exampl


# you can free to change all these codes/structure
def pick_a_number_from_list(A):
    total_sum = sum(A) #saves sum of all elements in the list

    cum_sum = 0
    cum_sum_dict = {} # initializing empty dictionary
    for i in range(len(A)):        #for every element
```

```
        cum_sum += A[i]/total_sum   #calculates cum sum for every element
        cum_sum_dict[A[i]] = cum_sum #adds to dictionary



    rand = uniform(0.0,1.0)  #random value from range 0.0 to 1.0

    for num,csum in cum_sum_dict.items():
        if rand <= csum:     #comparing rand with cumulative sum
            Final_num = num    # storing the number where rand <= cumulative sum of tha
            break              #breaking loop once found

    return Final_num

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

A = [0 ,5 ,27 ,6 ,13 ,28 ,100 ,45 ,10 ,79]
sampling_based_on_magnitued()
```

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

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

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 [91]:
```python
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):

    #https://www.tutorialspoint.com/python/python_reg_expressions.htm
    #re.sub(pattern, repl, string, max=0)
    new_string = re.sub(r'\D', "", String)      #replace not digit with blank

    #re.sub(pattern, repl, string, max=0)
    new_string = re.sub(r'\d', "#", new_string) #replace digit with #
    return(new_string)


String = "#2a$#b%c%561#"

replace_digits(String)
```

Out[91]:
```
'####'
```

Q4: Students marks dashboard

consider the marks list of class students given two lists
Students =
['student1','student2','student3','student4','student5','student6','student7','student8','student9','student1(
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','s

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
```

In [207…

```
# 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
# you can free to change all these codes/structure
def display_dash_board(students, marks):
    # write code for computing top top 5 students
    students_marks = {}

    for idx , student in enumerate(Students):
        students_marks[Marks[idx]] = Students[idx]  #dictionary of marks , students

    top_5_students = {k: v for k, v in sorted(students_marks.items(),reverse=True)[:5]}
    print("a.")
    for i,j in top_5_students.items():

        print(j,i)

    least_5_students = {k: v for k, v in sorted(students_marks.items(),reverse=False)[:
    print("\nb.")
    for i,j in least_5_students.items():
        print(j,i)

    index_25 = math.floor(0.25 * (len(Students)+1))    #index of 25th precentile


    index_75 = math.floor(0.75 * (len(Students)+1))    #index of 75th precentile


    students_25_75 = {k: v for k, v in sorted(students_marks.items(),reverse=False)}
    print("\nc.")
    for i,j in enumerate(students_25_75.items()):
        if (i+1 > index_25 and i+1 < index_75):      #prints the students between the c
            print(j[1] , j[0])
    return


Students=['student1','student2','student3','student4','student5','student6','student7',
Marks = [45, 78, 12, 14, 48, 43, 47, 98, 35, 80]

display_dash_board(Students, Marks)
```

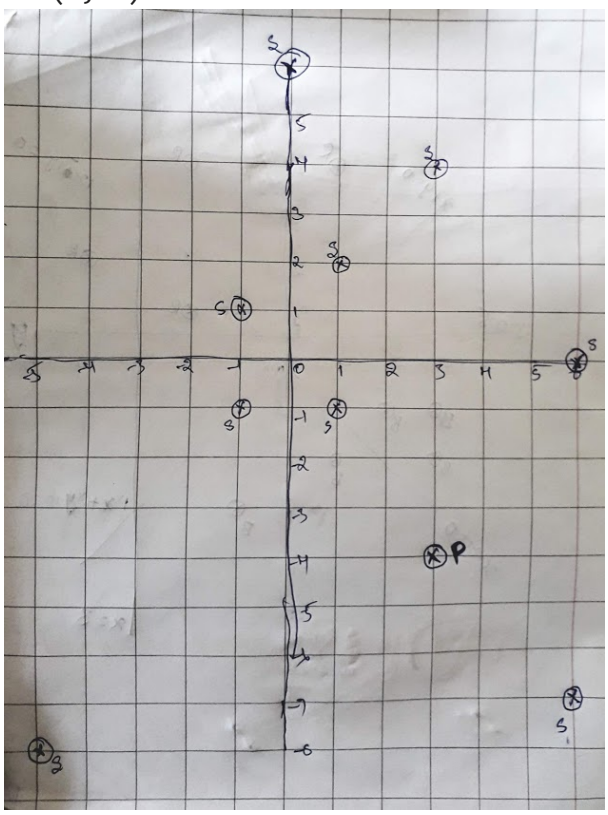
a.

```
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)
```

In [51]:

```
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 closest_points_to_p(S, P):
    Cosine_distance_dict = {}

    for i in range(len(S)):
        x= S[i][0]     #storing points
        y= S[i][1]
        p = P[0]       #base point
        q = P[1]

        Cosine_distance = math.acos( ((x*p)+(y*q)) / (( math.sqrt(math.pow(x,2) + math.
        Cosine_distance_dict[Cosine_distance] = (x,y)    #saving results in dictionary

    Nearest_five = {k:v for k,v in sorted(Cosine_distance_dict.items())[:5]} #sorting l
    for dist , pt in Nearest_five.items():
        print(pt)

    return   # 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)

closest_points_to_p(S, P)
# points = closest_points_to_p(S, P)
# print() #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

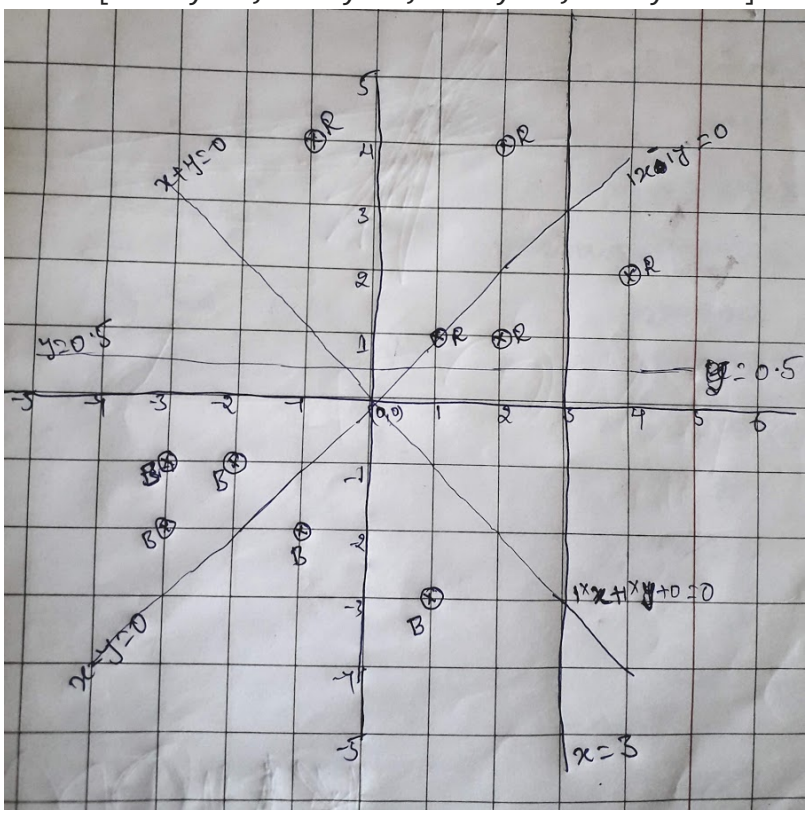
```
Red =[(R11,R12),(R21,R22),(R31,R32),(R41,R42),(R51,R52),..,(Rn1,Rn2)]
Blue=[(B11,B12),(B21,B22),(B31,B32),(B41,B42),(B51,B52),..,(Bm1,Bm2)]
```

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

In [159…

```
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 string


# you can free to change all these codes/structure
def i_am_the_one(red,blue,line):

    #https://stackoverflow.com/questions/6508043/regular-expression-to-find-any-number-
    line_var = [float(d) for d in re.findall(r'[+-]?\d+(?:\.\d+)?', line)]

    lst_red = []
    lst_blue = []

    for i in range(len(red)):
        p = red[i][0]
        q = red[i][1]
        a = line_var[0]
```

```
        b = line_var[1]
        d = line_var[2]



        dist_red = (a*p) + (b*q) - (d *-1)     # all points for red
        if(dist_red >  0 ):
            lst_red.append(True)
        else:
            lst_red.append(False)

    for j in range(len(blue)):
        p = blue[j][0]
        q = blue[j][1]
        a = line_var[0]
        b = line_var[1]
        d = line_var[2]



        dist_blue = (a*p) + (b*q) - (d *-1)   # all points for red
        if(dist_blue >  0 ):
            lst_blue.append(True)
        else:
            lst_blue.append(False)

    if ((all(lst_red) ==True and all(lst_blue)==False) or (all(lst_red) ==False and all
        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"]

for i in Lines[:4]:
    yes_or_no = i_am_the_one(Red, Blue, i)
    print() # 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) is distributed
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 values 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 [81]:

```
# 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
def curve_smoothing(string):

    lst = []

    for i in S.split(","):
        if (i=="_"):
            lst.append(0)
        else:
            lst.append(int(i))      #converted string to list of numbers , replace blan

    firstdigit = 0
    for i in range(len(lst)):   #loop runs till first digit

             if (lst[i] != 0):
                    firstdigit = i
                    for j in range(firstdigit+1):
                        lst[j] = lst[i]/(i+1)
                    break                           #break used to stop the loop after
```

```
    second_digit = 0
    for k in range(firstdigit+1,len(lst)): #3   #Loop runs from  first digit to second
            if (lst[k] != 0):
                    second_digit = k
                    sum_cent = lst[firstdigit]+lst[k]
                    for l in range(firstdigit,k+1):
                        lst[l] = (sum_cent)/(k-firstdigit+1)
                    break
    third = lst[second_digit]
    for a in range (second_digit,len(lst)):  #Loop runs from  third digit to end
            if (lst[len(lst)-1] == 0):              #checks if last digit is zero
                    lst[a] = third/(len(lst)-second_digit)
    final_lst = []

    for z in lst:
        final_lst.append(int(z))       #converts each element to int

    in_string = ""
    in_string = ','.join([str(i) for i in final_lst])  #converts to string comma sepera

    return in_string

S=  "_,_,30,_,_,_,50,_,_"



smoothed_values= curve_smoothing(S)
print(smoothed_values)
```

```
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

In [183…

```
# 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
def compute_conditional_probabilites(A):
    lst = []
    for i in range(len(A)):
        lst.append(A[i][1])  # store all S

    #https://stackoverflow.com/questions/3496518/using-a-dictionary-to-count-the-items-
    counts_S = dict()
    for i in lst:
        counts_S[i] = counts_S.get(i, 0) + 1  # store count of S in a dictionary


    counts_FS = dict()
    for i in range(len(A)):
        F = A[i][0]
        S = A[i][1]
        concat_FS = F+S
        counts_FS[concat_FS] =    counts_FS.get(concat_FS,0)+1   # Store count of F+S i


    for i in range(5):
        print (str(i+1)+".", end = " ")
        for j in range(3):
            get_S = counts_S.get("S"+str(j+1))    #get value of S
            get_FS = counts_FS.get("F"+str(i+1)+"S"+str(j+1),0)  #get value of FS
            prob = round(get_FS/get_S,2)     #Calculate conditional probability
            print ("P(F=F"+str(i+1)+"|S==S"+str(j+1)+")=",prob,end=" ")    #Print answe
        print("\n")



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

compute_conditional_probabilites(A)
```

```
1. P(F=F1|S==S1)= 0.25 P(F=F1|S==S2)= 0.33 P(F=F1|S==S3)= 0.0

2. P(F=F2|S==S1)= 0.25 P(F=F2|S==S2)= 0.33 P(F=F2|S==S3)= 0.33

3. P(F=F3|S==S1)= 0.0 P(F=F3|S==S2)= 0.33 P(F=F3|S==S3)= 0.33

4. P(F=F4|S==S1)= 0.25 P(F=F4|S==S2)= 0.0 P(F=F4|S==S3)= 0.33

5. P(F=F5|S==S1)= 0.25 P(F=F5|S==S2)= 0.0 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']
```

In [134…

```
# 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
def string_features(S1, S2):
    a = 0
    b = S1.split(" ")            #create list of s1
    c = S2.split(" ")            #create list of s2
    for i in S1.split(" "):      #for every word in S1
        for j in S2.split(" "):  #for every word in S2
            if (i == j ):        #for every match
                a+=1             #to store count of matches
                b.remove(i)      #remove matched words from b
                c.remove(i)      #remove matched words from c
                break


    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. " , a)
print("b. " , b)
print("c. " , c)
```

```
a.  7
b.  ['first', 'F', '5']
c.  ['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

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

In [152…

```python
# 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

import math

# you can free to change all these codes/structure
def compute_log_loss(A):
    n = len(A) #length of list
    temp = 0
    for i in range(len(A)): #for individual list in the entire list
        temp += ((A[i][0]*math.log10(A[i][1])) + ((1-A[i][0])*math.log10(1-A[i][1]))) #
    loss = -1*(temp / n) #final loss calculation
    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
```