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
```

In [68]:
```
# write your python code here
# you can take the above example as sample input for your program to te
st
# it should work for any general input try not to hard code for only gi
ven input examples
A = [[1,3,4],[2,5,7],[5,9,6]]
B = [[1,0,0],[0,1,0],[0,0,1]]
A_cross_B = [[0 for row in range(len(A))] for col in range(len(B[0]))]
#intializing the resultant matrix
# you can free to change all these codes/structure
# here A and B are list of lists
#print(A_cross_B)
#print(len(A))
#print(len(B[0]))
def matrix_mul(A, B):
    # write your code
    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)):
                    A_cross_B[i][j] += A[i][k] * B[k][j]
        return(A_cross_B)
    else:
        return("Multiplication can't be performed")

matrix_mul(A, B)
```

Out[68]:
```
[[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 ex
periments.
f(100) > f(79) > f(45) > f(28) > f(27) > f(13) > f(10) > f(6) >
f(5) > f(0)
```

In [3]:

```
import random
# write your python code here
# you can take the above example as sample input for your program to te
st
# it should work for any general input try not to hard code for only gi
ven input examples

A = [0,5,27,6,13,28,100,45,10,79]
# you can free to change all these codes/structure
weight = []
def weight_calculation(num_list):
    cumulative_sum = sum(num_list)
    for each in num_list:
        weight.append(each / cumulative_sum)
    print(weight)

weight_calculation(A)
def pick_a_number_from_list(A):
    return random.choices(A,weight,k=1)[0]

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

```
sampling_based_on_magnitude()
#weight_calculation(number_list)
```

```
[0.0, 0.01597444089456869, 0.08626198083067092, 0.019169329073482427,
0.04153354632587859, 0.08945686900958466, 0.3194888178913738, 0.1437699
6805111822, 0.03194888178913738, 0.2523961661341853]
100
27
100
79
100
100
28
27
28
100
27
100
45
79
100
100
27
100
27
5
79
100
100
100
79
79
100
100
79
100
79
100
100
13
```

```
13
100
27

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

```
100
45

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

Q3: Replace the digits in the string with

Consider a string that will have digits in that, we need to remove all the characters which are 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 [1]:
```
import re
# write your python code here
# you can take the above example as sample input for your program to te
st
# it should work for any general input try not to hard code for only gi
ven input examples
String1 = input("Enter the string: \n")
# 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
    a = re.sub("\D","",String)
    a = re.sub("[0-9]","#",a)
    #re.sub("[0-9]","#",String)
    return(a) # modified string which is after replacing the # with dig
its

replace_digits(String1)
```

```
Enter the string:
2a45sad
```

Out[1]:
```
'###'
```

Q4: Students marks dashboard

Consider the marks list of class students given in two lists
Students =
['student1','student2','student3','student4','student5','student6','student7','student8','student9','studer
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','student
5','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
```

In [2]:
```
# write your python code here
```

```
# you can take the above example as sample input for your program to te
st
# it should work for any general input try not to hard code for only gi
ven input examples
Students=['student1','student2','student3','student4','student5','stude
nt6','student7','student8','student9','student10']
Marks = [45, 78, 12, 14, 48, 43, 47, 98, 35, 80]
key_val = dict()
# you can free to change all these codes/structure

def display_dash_board(students, marks):
    i=0
    for each in students:
        key_val[each]=marks[i]
        i+=1
    sorted_kv = sorted(key_val.items(),key = lambda kv: kv[1])
    print("Top 5 Ranks")
    for i in range(1,6):
        print(sorted_kv[-i])
    print("Least 5 Ranks")
    for i in range(0,5):
        print(sorted_kv[i])
    n = len(marks)
    first_quartile = int(n/4) #if (n/4).is_integer() else int(n/4)+1
    third_quartile = int(3*(n/4)) #if 3*(n/4).is_integer() else int((3*
(n/4)))+1
    print(">25 and < 75")
    inter = sorted_kv[first_quartile:third_quartile]
    for i in range(0,len(inter)):
        print(inter[i])

display_dash_board(Students,Marks)


"""
    top_5_students = # compute this
    # write code for computing top least 5 students
    least_5_students = # compute this
    # write code for computing top least 5 students
```

```
    students_within_25_and_75 = # compute this

    return top_5_students, least_5_students, students_within_25_and_75

top_5_students, least_5_students, students_within_25_and_75 = display_d
ash_board(students, marks)
print(# those values)"""
```

```
Top 5 Ranks
('student8', 98)
('student10', 80)
('student2', 78)
('student5', 48)
('student7', 47)
Least 5 Ranks
('student3', 12)
('student4', 14)
('student9', 35)
('student6', 43)
('student1', 45)
>25 and < 75
('student9', 35)
('student6', 43)
('student1', 45)
('student7', 47)
('student5', 48)
```

Out[2]:
```
'\n    top_5_students = # compute this\n    # write code for computing
top least 5 students\n    least_5_students = # compute this\n    # writ
e code for computing top least 5 students\n    students_within_25_and_7
5 = # compute this\n    \n    return top_5_students, least_5_students,
students_within_25_and_75\n\ntop_5_students, least_5_students, students
_within_25_and_75 = display_dash_board(students, marks)\nprint(# those
values)'
```

Q5: Find the closest points

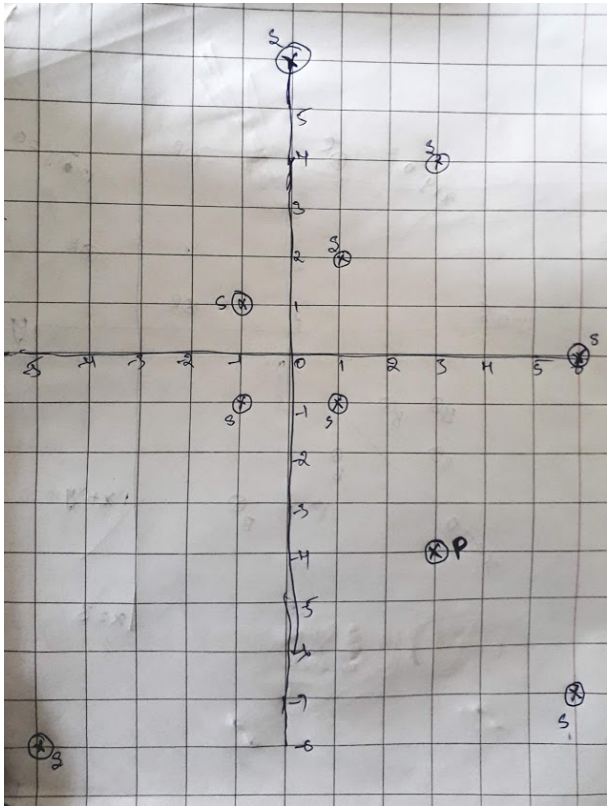
Consider you are given n data points in the form of list of tuples like S=[(x1,y1),(x2,y2),(x3,y3),(x4,y4),(x5,y5),..,(xn,yn)] and a point P=(p,q)

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 defined 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)
```

In [4]:

```python
import math

# write your python code here
# you can take the above example as sample input for your program to te
st
# it should work for any general input try not to hard code for only gi
ven 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):
    distance = {}
    p = P[0]
    q = P[1]

    for each in S:
        x = each[0]
        y = each[1]

        num = x*p + y*q
        a = math.sqrt(x**2 + y**2)
        b = math.sqrt(p**2 + q**2)
        den = a*b
        val = num/den
        val = math.acos(val)
        point = (each[0],each[1])
        distance[point] = val

    sorted_points = sorted(distance.items(),key = lambda x:x[1])
    #print(distance)
    #print(sorted_points)
    for i in range(0,5):
```

```
        print(sorted_points[i][0])

    #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 the returned values
```

```
(6, -7)
(1, -1)
(6, 0)
(-5, -8)
(-1, -1)
```

Q6: Find which line separates oranges and apples

Consider you are 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)]
```

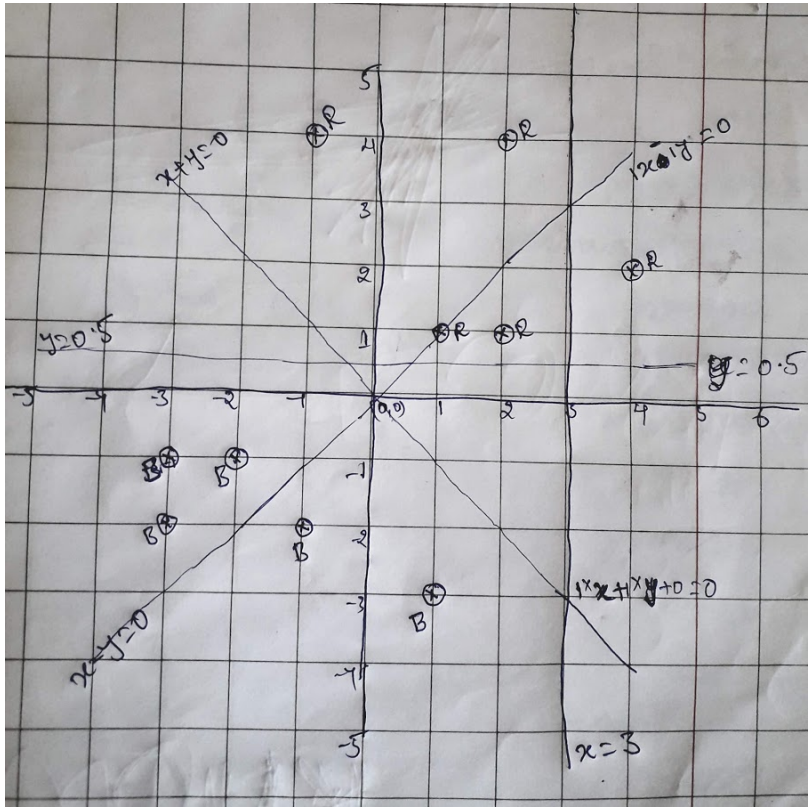
and set of line equations(in the string format, i.e list of strings)

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

Your task here is to print "YES"/"NO" for each line given. You should print YES, if all the red points are one side of the line and blue points are on other side of the line, otherwise you should print 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 [5]:
```
import math
import re
# write your python code here
# you can take the above example as sample input for your program to te
```

```
st
# it should work for any general input try not to hard code for only gi
ven input strings


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

    line_ch = re.sub(r'[a-z]+',"",line)
    line_ch = re.sub(r'[0-9]+',"",line_ch)

    line_ch = list(line_ch)


    line1 = re.split('\+|\-', line)
    x_coff = line1[0]
    x_coff = re.sub(r'[a-z]+',"",x_coff)
    y_coff = line1[1]
    y_coff = re.sub(r'[a-z]+',"",y_coff)
    intercept = line1[2]
    for each in red:
        x, y = each[0],each[1]

    equation = ""

    equation += str(x_coff)+"*"+"x"
    equation += str(line_ch[0])+str(y_coff)+"*"+"y"+str(line_ch[1])+str
(intercept)
    red_list = []
    blue_list = []


    for i in range(0,len(red)):
        x = red[i][0]
        y = red[i][1]
        equation1 = eval(str(equation))
        red_list.append(equation1)

    for i in range(0,len(blue)):
```

```
        x = blue[i][0]
        y = blue[i][1]
        equation1 = eval(str(equation))
        blue_list.append(equation1)
    #print(red_list)
    #print(blue_list)
    m,n,o,p = all(x > 0 for x in red_list),all(x < 0 for x in red_list
),all(x > 0 for x in blue_list),all(x < 0 for x in blue_list)

   # print(m,n,o,p)

    if (m&o) or (m&p) or (n&o) or (n&p):
        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 format

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 distri
buted 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 di
stributed qually to all 5 places

Ex 3: 80, _, _, _, _  ==> 80/5,80/5,80/5,80/5,80/5 ==> 16, 16, 1
6, 16, 16 i.e. the 80 is distributed qually to all 5 missing val
ues 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 value
s (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 (1
0, 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 [4]:
```
# write your python code here
# you can take the above example as sample input for your program to te
st
# it should work for any general input try not to hard code for only gi
ven input strings

final_string = []
# you can free to change all these codes/structure
def curve_smoothing(string):
    string = string.replace("_","0")
    split_string = string.split(",")
    #split_string.replace("_","0")
    for i in range(0,len(split_string)):
        try:
            split_string[i] = int(split_string[i])
        except ValueError:
            continue

    position = 0
    next_value = 0
    end = 0
    end_value = 0

    while position < len(split_string):
        if split_string[position]!=0 or (position+1 == len(split_string
)):
            if split_string[position]!=0:
                next_value = int(split_string[position])
            else:
                next_value = 0
            fill = (next_value + end_value) / (position - end + 1)
            for i in range(end,position+1):
                split_string[i]=fill
            end_value = fill
            end = position
        position += 1
    return split_string
```

```
S  =  "_,_,30,_,_,_,50,_,_"
S1 = "40,_,_,_,60"
S2 = "80,_,_,_,_"
S3 = "_,_,_,24"
L = [S,S1,S2,S3]
for each in L:
    smoothed_values = curve_smoothing(each)
    print(smoothed_values)
```

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

Q8: Find the probabilities

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,S
1],[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 [5]:

```
# write your python code here
# you can take the above example as sample input for your program to te
st
# it should work for any general input try not to hard code for only gi
ven input strings

F = ["F1","F2","F3","F4","F5"]
S = ["S1","S2","S3"]

# you can free to change all these codes/structure
def compute_conditional_probabilites(A):
    # your code
    S1_count,S2_count,S3_count = 0,0,0
    for each in A:
        if each[1]=="S1":
            S1_count+=1
        elif each[1]=="S2":
            S2_count+=1
        else:
            S3_count+=1
    for eachF in F:
        for eachS in S:
            if eachS == "S1":
                den = S1_count
            elif eachS == "S2":
                den = S2_count
```

```
            else :
                den = S3_count
            print("P(F={0}|S={1})={2}/{3}".format(eachF,eachS,A.count([
eachF,eachS]),den))




A = [["F1","S1"],["F2","S2"],["F3","S3"],["F1","S2"],["F2","S3"],["F3",
"S2"],["F2","S1"],["F4","S1"],["F4","S3"],["F5","S1"]]
compute_conditional_probabilites(A)
```

```
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: Operations on sentences

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

In [6]:

```
# write your python code here
# you can take the above example as sample input for your program to te
st
# it should work for any general input try not to hard code for only gi
ven input strings

# you can free to change all these codes/structure
def string_features(S1, S2):
    list1 = S1.split(" ")
    list2 = S2.split(" ")
    set1 = set(list1)
    set2 = set(list2)
    lst_intersec = [val for val in list1 if val in list2]
    a = len(lst_intersec)
    b = set1 - set2
    c = set2 - set1

    S2 = S2.split(" ")
    return a, list(b), list(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', '5', 'F'] ['S', 'second', '3']
```

Q10: Error Function

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
$f(Y, Y_{score}) = -1 * \frac{1}{n}\Sigma_{foreachY,Y_{score}pair}(Ylog10(Y_{score}) + (1-Y)log10(1-Y_{sco}$
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:
0.44982
```

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

In [7]:
```
# write your python code here
# you can take the above example as sample input for your program to te
st
# it should work for any general input try not to hard code for only gi
ven input strings
import math

# you can free to change all these codes/structure
def compute_log_loss(A):
    summation = 0
    n = len(A)
    for each in A:
        Y = float(each[0])
        Y_score = float(each[1])
```

```
        # print(Y,Y_score)
        summation += Y*math.log10(Y_score) + (1-Y)*math.log10(1-Y_score
)
        loss = (-1*1/n)*summation


    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
```