# 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
```

```
#Q1: Given two matrices please print the product of those two matrices

import time
SIZE = 10

#Enter data rowise
row1 = int(input("Enter the Number of rows for matrix A:- "))
col1 = int(input("Enter number of columns for matrix A:- "))
```

```
A = []

for i in range(row1):
    temp = []
    for j in range(col1):
        temp.append(int(input()))
    A.append(temp)

row2 = int(input("Enter the Number of rows for matrix B:- "))
col2 = int(input("Enter number of columns for matrix B:- "))

B = []

for i in range(row2):
    temp = []
    for j in range(col2):
        temp.append(int(input()))
    B.append(temp)

#Real Execution start from here
start = time.time()

def matrix_mul(A, B):
    """
    Function for multiplcation of two matrix
    Parameters:-Takes Two Matrix as parameters
    Returns:Multiplication matrix as result

    """
    if(col1 == row2): #checks the condition for matrix multiplication
        result = [[0 for i in range(SIZE)]
                for j in range(SIZE)]

        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]
        print("Result of AxB is:- ")
        for m in range(row1):
            for n in range(col2):
                print(result[m][n], end=" ")
            print()
    else:
        print("AxB Operaton Not Possible") #if above condition fails then this line got ex


matrix_mul(A, B) #function call
print(f'Time: {time.time() - start}')
```

```
Enter the Number of rows for matrix A:- 3
Enter number of columns for matrix A:- 3
1
3
4
```

```
2
5
7
5
9
6
Enter the Number of rows for matrix B:- 3
Enter number of columns for matrix B:- 3
1
0
0
0
1
0
0
0
1
Result of AxB is:-
1 3 4
2 5 7
5 9 6
Time: 0.005192756652832031
```

## 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
import time
start = time.time()
# Q2 Select a number randomly with probability proportional to its magnitude from the give


A = [0, 5 ,27 ,6 ,13 ,28, 100 ,45 ,10, 79]
def pick_a_number_from_list(A):
    sum=0
    cum_sum=[]
    for i in range(len(A)):  #Finding Commulative sum
        sum = sum + A[i]
        cum_sum.append(sum)
    r = int(random.uniform(0,sum)) #uniform function return random value between 0 and sum
    number=0
    for index in range(len(cum_sum)):
        if(r>=cum_sum[index] and r<cum_sum[index+1]):
            return A[index+1]
    return number


def sampling_based_on_magnitued():
    '''
    Runing loop 100 times and calling pick_a_number_from_list()
    function at each iteration

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

sampling_based_on_magnitued()
print(f'Time: {time.time() - start}')
```

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

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

## 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: ####
```

```
# Q3: Replace the digits in the string with #

import re
import time
String=input("Enter a String:- ")

start = time.time()

def replace_digits(String):
    '''
    Function replaces the digit with #

    '''
    for word in String:
        if word.isdigit(): #checking is given char in string is digit or not
            print("#",end="")


replace_digits(String)
print()
print(f'Time: {time.time() - start}')
```

```
Enter a String:- a2b3c4
###
Time: 0.0005576610565185547
```

## 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
```

```
#Q4: Students marks dashboard

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

import time
```

```
students=[]
marks=[]

n=int(input("How may number of recoord you want to insert:- "))
for i in range(0,n):
    name=(input("Enter Student Name:- "))
    students.append(name)
    mark=int(input("Enter Student Marks:- "))
    marks.append(mark)

start = time.time()

def display_dash_board(students, marks):
    '''
    Function prints the student data according to condtions givem in problem statement

    '''
    temp=[]
    temp=marks.copy()  #data is copied for temporary use so actual data may stay as it is
    temp.sort(reverse=True)
    top_5_students=temp[:5]
    temp.sort()
    least_5_students=temp[:5]
    students_within_25_and_75=[]
    for mark in marks:
        if mark>25 and mark<75: #checks for the marks between 25 and 75
            students_within_25_and_75.append(mark)

    return top_5_students, least_5_students, students_within_25_and_75

top_5_students, least_5_students, students_within_25_and_75 = display_dash_board(students,

print("a.")
for ele in top_5_students:   #printing top 5 student
    print(students[marks.index(ele)]+"  "+str(ele))

print("b.")
for ele in least_5_students: #printing last 5 students
    print(students[marks.index(ele)]+"  "+str(ele))

print("c.")
students_within_25_and_75.sort()
for ele in students_within_25_and_75: #printing student having marks between 25 and 75
    print(students[marks.index(ele)]+"  "+str(ele))
1
print(f'Time: {time.time() - start}')
```

```
How may number of recoord you want to insert:- 10
Enter Student Name:- student1
Enter Student Marks:- 45
Enter Student Name:- student2
Enter Student Marks:- 78
Enter Student Name:- student3
Enter Student Marks:- 12
Enter Student Name:- student4
Enter Student Marks:- 4
Enter Student Name:- student5
Enter Student Marks:- 48
Enter Student Name:- student6
Enter Student Marks:- 43
Enter Student Name:- student7
Enter Student Marks:- 47
Enter Student Name:- student8
Enter Student Marks:- 98
Enter Student Name:- student9
Enter Student Marks:- 35
Enter Student Name:- student10
Enter Student Marks:- 80
a.
student8  98
student10  80
student2  78
student5  48
student7  47
b.
student4  4
student3  12
student9  35
student6  43
student1  45
c.
student9  35
student6  43
student1  45
student7  47
student5  48
Time: 0.0014774799346923828
```

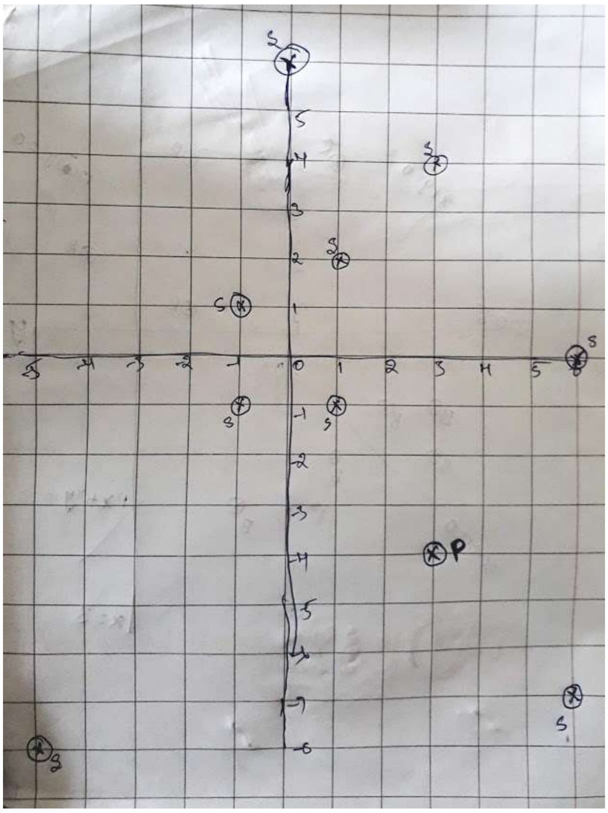
## Q5: Find the closest points

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)
```

```
#Q5: Find the closest points

import math
import time
start = time.time()
def clst_pts(S, P):
    '''
    Function taking x and y cordinate as input
    Returns Final list containing closed points

    '''
    clst_pts = []
    final_list = []

    for point in S:
        dnmntr = math.sqrt((point[0] ** 2) + (point[1] ** 2)) * math.sqrt((P[0] ** 2) + (P
        nmrtr = point[0] * P[0] + point[1] * P[1]

        if dnmntr != 0:
            cosine_distance_for_this_point = math.acos(nmrtr / dnmntr)
            clst_pts.append((cosine_distance_for_this_point, point))

    for item in sorted(clst_pts, key=lambda x: x[0])[:5]: #using lambda function to apply
        final_list.append(item[1])

    return final_list

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

```
P = (3, -4)

clst_pts = clst_pts(S, P)
print("Closest point-cosine-distance - top 5:", *[point for point in clst_pts], sep="\n")

print(f'Time: {time.time() - start}')
```

```
Closest point-cosine-distance - top 5:
(6, -7)
(1, -1)
(6, 0)
(-5, -8)
(-1, -1)
Time: 0.0008704662322998047
```

## 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)]
```

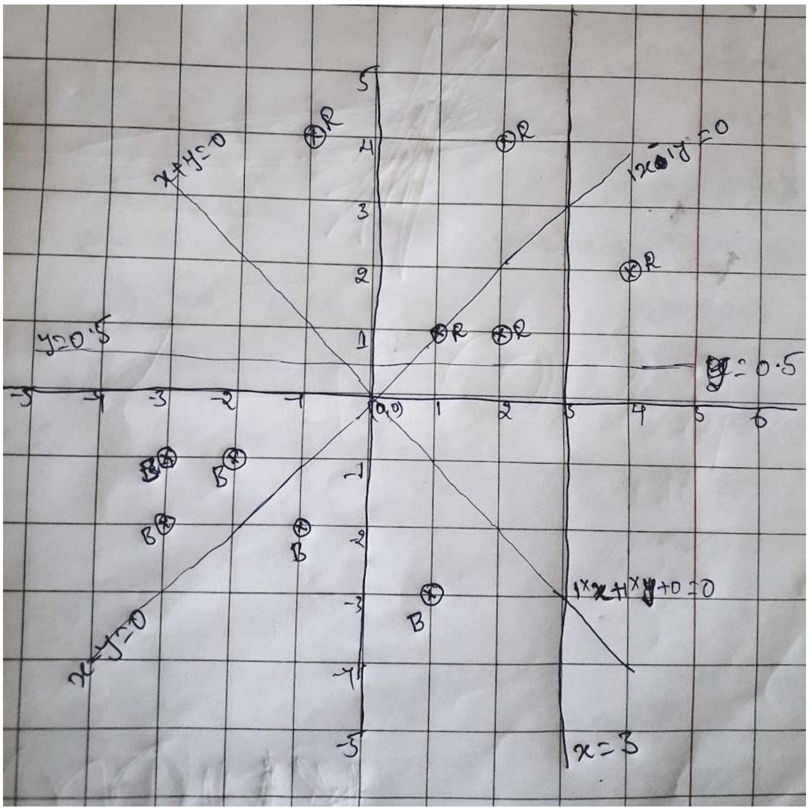
and set of line equations(in the string formate, i.e list of strings)

```
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
```

```
#Q6: Find Which line separates oranges and apples

import math
import time
start = time.time()

def i_am_the_one(red,blue,line):
    '''
    Function return the result by evaluating equation
    with cordinates give by user

    '''


    for i in red:
        eq=line.replace('x','*'+str(i[0]))
        eq=eq.replace('y','*'+str(i[1]))
        answer=eval(eq)
        if answer>0:
            pass
        else:
            return "NO"


    for j in blue:
        eq1=line.replace('x','*'+str(j[0]))
        eq1=eq1.replace('y','*'+str(j[1]))
        answer1=eval(eq1)
        if answer1<0:
            pass
        else:
            return "NO"
    return "Yes"

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"]

#by using belowed code you can give manual input

# Red=[]
# Blue=[]

# n=int(input("How many points you want to check: "))
# print("Enter Red Points")
# print()
# for i in range(0,n):
#     temp=[]
#     x=int(input("X cordinate:- "))
#     temp.append(x)
#     y=int(input("Y cordinate:- "))
#     temp.append(y)
#     Red.append(temp)

# print("Enter Blue Points")
# print()
# for i in range(0,n):
#     temp=[]
#     x=int(input("X cordinate:- "))
#     temp.append(x)
#     y=int(input("Y cordinate:- "))
#     temp.append(y)
#     Blue.append(temp)




for i in Lines:
    yes_or_no = i_am_the_one(Red, Blue, i)
    print(yes_or_no)

print(f'Time: {time.time() - start}')
```

```
Yes
NO
NO
Yes
Time: 0.0018417835235595703
```

## 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 i

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,
```

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
```

```
#Q7: Filling the missing values in the specified formate
import time
start = time.time()
def curve_smoothing(string):
    a = S.split(',')
    count = 0
    middle_store = 0
    for i in range(len(a)):
        if a[i] == '_':
            count = count + 1  # find number of blanks to the left of a number
        else:
            for j in range(i + 1):
                # if there are n blanks to the left of the number speard the number equal
                a[j] = str((float(a[i]) / (count + 1)))
            middle_store = i
            middle_store_value = float(a[i])
            break

        # blanks in the middle
    denominator = 1
    flag = 0
    for k in range(middle_store + 1, len(a)):
        if a[k] != '_':
            denominator = (k + 1 - middle_store)
            flag = k
            break
    flag_value = float(a[flag])
    for p in range(middle_store, flag + 1):
        a[p] = str((middle_store_value+flag_value) / denominator)

    # blanks at the right
    last_value = float(a[flag])
    for q in range(flag, len(a)):
        a[q] = str(last_value / (len(a) - flag))

    return a

S=  "_,_,30,_,_,_,50,_,_"
smoothed_values= curve_smoothing(S)
for ele in smoothed_values:
    print((ele),end=",")

print(f'Time: {time.time() - start}')
```

```
10.0,10.0,12.0,12.0,12.0,12.0,4.0,4.0,4.0,Time: 0.0026674270629882812
```

## 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

```
#Q8: Filling the missing values in the specified formate

import time
start = time.time()
def compute_conditional_probabilites(F, S):
    numerator = 0
    denominator = 0
    for i in range(len(A)):
        if(A[i][1] == S):
            denominator = denominator + 1
            if(A[i][0] == F):
                numerator = numerator + 1
    print('P(F = {} | S == {}) = {}/{}'.format(F, S, str(numerator), str(denominator)))

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

for i in ['F1', 'F2', 'F3', 'F4', 'F5']:
    for j in ['S1', 'S2', 'S3']:
        compute_conditional_probabilites(i, j)

print(f'Time: {time.time() - start}')
```

```
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
Time: 0.006081104278564453
```

## 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']
```

```
#Q9: Given two sentances S1, S2

import time

# S1= "the first column F will contain only 5 uniques values"
# S2= "the second column S will contain only 3 uniques values"
S1=input("Enter String1:- ")
S2=input("Enter String2:- ")

start = time.time()

def string_features(S1, S2):
    b=[]
    c=[]
    a=[]
    S1_temp=S1.split(" ")
    S2_temp=S2.split(" ")
    for ele in S1_temp:
        if ele in S2_temp:
            a.append(ele)
    for ele in S1_temp:
        if ele not in S2_temp:
            b.append(ele)
    for ele in S2_temp:
        if ele not in S1_temp:
            c.append(ele)
    return len(a), b, c

a,b,c = string_features(S1, S2)
print(a)
print(b)
print(c)

print(f'Time: {time.time() - start}')
```

```
Enter String1:- the first column F will contain only 5 uniques values
Enter String2:- the second column S will contain only 3 uniques values
7
['first', 'F', '5']
['second', 'S', '3']
Time: 0.0014641284942626953
```

## 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
```

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

```
#Q10: Given two sentances S1, S2

from math import log
import time
start = time.time()


def compute_log_loss(A):
    cross_entropy = 0
    for row in A:
        cross_entropy += (row[0] * log(row[1], 10) + ((1 - row[0]) * log(1 - row[1], 10)))

    log_loss = -1 * cross_entropy / len(A)
    return log_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]]
print(compute_log_loss(A))

print(f'Time: {time.time() - start}')
```

```
0.42430993457031635
Time: 0.000453948974609375
```

✓ 0s completed at 6:22 PM ● ×