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 [17]:

```
#Reference : Geeks for geeks for matrix input


Row_1=int(input("Enter the number of rows for first matrix : ")) # Enter number of rows for first
matrix
Col_1=int(input("Enter the number of columns for first matrix : ")) # Enter number of Columns firs
t matric
Row_2=int(input("Enter the number of rows for second matrix : ")) #Enter the number of rows for se
cond matrix
Col_2=int(input("Enter the number of columns for second matrix : ")) #Enter the number of columns
for secondmatric
a = [[0 for col in range(Col_1)] for row in range(Row_1)] # initialize zeros
b = [[0 for col in range(Col_2)] for row in range(Row_2)]
c = [[0 for col in range(Row_1)] for row in range(Col_2)]
F=[]

if Row_1==Col_2: # will enter the loop only if number of rows = number of columns of other matrix

    for i in range(Row_1):
        print("Enter the elements of the First matrix row wise : ")
        for j in range(Col_1):

            a[i][j]=int(input("Enter the elements of the first matrix row wise: ")) # enter first m
atrix

    for i in range(Row_2):
        print("Enter the elements of the Second matrix row wise : ")
        for j in range(Col_2):

            b[i][j]=int(input("Enter the elements of the second matrix row wise: ")) #Enter second
matrix
```

```
    for i in range(Row_1):
        for j in range(Col_1):
            c[i][j]=0
            for k in range(Col_2):
                c[i][j]+=a[i][k]*b[k][j]
            F.append(c[i][j])




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


print(F)
```

```
Enter the number of rows for first matrix : 2
Enter the number of columns for first matrix : 2
Enter the number of rows for second matrix : 2
Enter the number of columns for second matrix : 2
Enter the elements of the First matrix row wise :
Enter the elements of the first matrix row wise: 1
Enter the elements of the first matrix row wise: 2
Enter the elements of the First matrix row wise :
Enter the elements of the first matrix row wise: 3
Enter the elements of the first matrix row wise: 4
Enter the elements of the Second matrix row wise :
Enter the elements of the second matrix row wise: 5
Enter the elements of the second matrix row wise: 6
Enter the elements of the Second matrix row wise :
Enter the elements of the second matrix row wise: 7
Enter the elements of the second matrix row wise: 8
[19, 22, 43, 50]
```

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 [15]:

```
#Reference :#https://stackoverflow.com/questions/16489449/select-element-from-array-with-
probability-proportional-to-its-value
#https://medium.com/@appliedaicourse_56208/faqs-of-python-mandatory-assignment-7ada380ec770

import random
from bisect import bisect

A = [0,5,27,6,13,28,100,45,10,79]
A.sort()
def pick_a_number_from_list(A):
    cumsum = [sum(A[:i+1]) for i in range(len(A))]
    out = []
    rnd = random.random()                       #Generate a number between 0 to 1
    index = bisect(cumsum, rnd*cumsum[-1])   # it gives us the index of A
    #print(index)
    out.append(A[index])
```

```
    return out


def sampling_based_on_mag():
    for num in range(1,100):
        num=pick_a_number_from_list(A)
        print(num , end='')

sampling_based_on_mag()
```

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

In [ ]:

```

```

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 [19]:

```
String=str(input("Enter the string : "))
def split(word):
    return [char for char in word]
out=""
for i in String:
    if i.isdigit():
        out+="#"
    else :
        out+=""

print(out)
```

```
Enter the string : Hake5h kum@r i5 w0rking i4 moengag3
#####
```

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','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','student8','st
t9','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 [18]:

```
import math
dictionary = dict(zip(students, marks))
def students_dashboard(students,marks):
    dictionary = dict(zip(students, marks))
    sortDict=(sorted(dictionary.items(), key=lambda x: x[1]))
    #print(sortDict)
    top5=list(sorted(dictionary.items(), key=lambda x: x[1], reverse=True)[:5])
    least5=list(sorted(dictionary.items(), key=lambda x: x[1], reverse=False)[:5])
    print('a.')
    for i,j in top5:
        print(i,j)
    print('\n')
    print("b.")
    for i,j in least5:
        print(i,j)
    print('\n')
A=students_dashboard(students,marks)
print('c.')
def students_within_25_and_75(marks, percent):
    if not marks:
            return None
    k = len(marks) * percent
    f = math.floor(k)
    c = math.ceil(k)
    if f == c:
        return marks[int(k)]
    d0 = marks[int(c)+1]
    d1 = marks[int(c)]
    return (d0+d1)/2
students=['student1','student2','student3','student4','student5','student6','student7','student8',
'student9','student10']
marks = [45, 78, 12, 14, 48, 43, 47, 98, 35, 80]

B= students_within_25_and_75(marks, 0.25)
C= students_within_25_and_75(marks, 0.75)
count=0
dictionary = dict(sorted(dictionary.items(), key=lambda x: x[1], reverse=False))
for k,v in dictionary.items():
        if v > B and v < C:
            if count<=4:
                print(k,v)
                count+=1
```

```
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 [ ]:

```
```

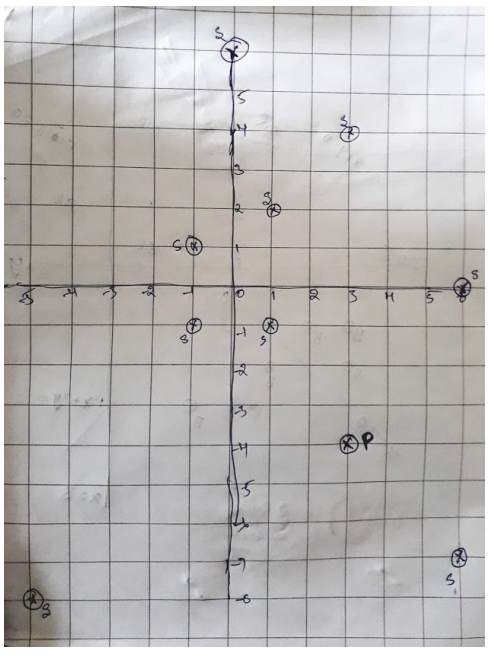
Q5: Find the closest points

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

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 [21]:

```
#Reference : https://stackoverflow.com/questions/3071415/efficient-method-to-calculate-the-rank-ve
```

```
#Reference : https://stackoverflow.com/questions/30771113/efficient-method-to-calculate-the-rank-ve
ctor-of-a-list-in-python


import math #importing math to use a.cos
S= [('1','2'),('3','4'),('-1','1'),('6','-7'),('0','6'),('-5','-8'),('-1','-1'),('6','0'),('1','-1'
)] #points initialization
P= ('3','-4')
Cosine_Distance=[]#Creating empty list to store  the cosine distance as list
for i in range(len(S)):
    x=int(S[i][0]) # Getting first point of first tuple S ( in second iteration its second tuple an
d so on)
    y=int(S[i][1]) #Getting second point of first  tuple S ( in second iteration its second tuple a
nd so on)
    Cosine_Distance.append(math.acos((x*int(P[0])+y*int(P[1]))/math.sqrt((x**2+y**2)*(int(P[0])**2+
int(P[1])**2)))) # using a.cos formula calculated cosine dist

    print("Points are : {0} {1}".format(x,y) )

print("\n")
def Argsort(Cosine_Distance):  # since we cannot use numpy created function to get cosine function
( Reference attached )
    return sorted(range(len(Cosine_Distance)), key=Cosine_Distance.__getitem__)
Result=Argsort(Cosine_Distance) # storing indices  as per the cosine distance list
J=[S[i] for i in Result] # printing the tuple as per the indices (first iteration s[i] is 3  so it
gives (6,-7)and so on)
print("Closest Distance to the given point:  {0}".format(J))
```

```
Points are : 1 2
Points are : 3 4
Points are : -1 1
Points are : 6 -7
Points are : 0 6
Points are : -5 -8
Points are : -1 -1
Points are : 6 0
Points are : 1 -1


Closest Distance to the given point:  [('6', '-7'), ('1', '-1'), ('6', '0'), ('-5', '-8'), ('-1',
'-1'), ('3', '4'), ('1', '2'), ('0', '6'), ('-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

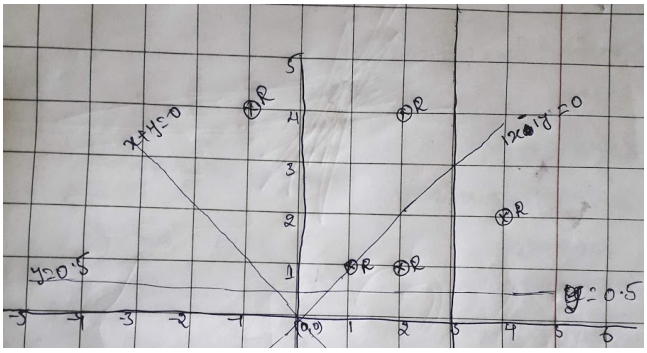
```
Red =[(R11,R12),(R21,R22),(R31,R32),(R41,R42),(R51,R52),..,(Rn1,Rn2)]
Blue=[(B11,B12),(B21,B22),(B31,B32),(B41,B42),(B51,B52),..,(Bm1,Bm2)]
```

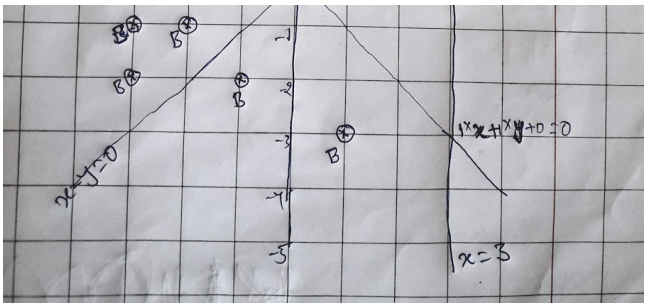
and set of line equations(in the string format, i.e list of strings)

```
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 coefficients 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 [22]:

```
import re
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"]


def i_am_the_one(red,blue,line):


    list_of_coef=[]
    list_of_coef_clean = []
    for x in Lines:
        list_of_coef.append(re.split('x|y', x))
    for i in list_of_coef:
        temp_list=[]
        for k in i:
            if ('+' in k):
                temp_list.append(float(k.split('+')[1]))
            else:
                temp_list.append(float(k))
        list_of_coef_clean.append(temp_list)
    list_of_coef_clean = [i for i in list_of_coef_clean]


    for line in list_of_coef_clean:
        red_flag_list = []
        blue_flag_list = []

        for i in range(len(Red)):
            r = ((line[0]*Red[i][0])+(line[1]*Red[i][1])+line[2])
            if r>0:
                red_flag_list.append(True)
            else:
                red_flag_list.append(False)

        for j in range(len(Blue)):
            bl = ((line[0]*Blue[i][0])+(line[1]*Blue[i][1])+line[2])
            if bl<0:
                blue_flag_list.append(True)
            else:
                blue_flag_list.append(False)


        if (all(red_flag_list) and all(blue_flag_list)):
                 print('Yes')
        else:
                 print('No')


yes_or_no = i_am_the_one(Red, Blue, i)
```

```
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 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 di
stributed 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 [12]:

```
# takes an array x and two indices a,b.
# Replaces all the _'s with (x[a]+x[b])/(b-a+1)
def fun(x, a, b):
    if a == -1:
        v = float(x[b])/(b+1)
        for i in range(a+1,b+1):
            x[i] = v
    elif b == -1:
        v = float(x[a])/(len(x)-a)
        for i in range(a, len(x)):
            x[i] = v
    else:
        v = (float(x[a])+float(x[b]))/(b-a+1)
        for i in range(a,b+1):
            x[i] = v
    return x

def replace(text):
    # Create array from the string
    x = text.replace(" ","").split(",")
    # Get all the pairs of indices having number
    y = [i for i, v in enumerate(x) if v != '_']
    #print(list(enumerate(x)))
    # Starting with _ ?
    if y[0] != 0:
        y = [-1] + y
    # Ending with _ ?
    if y[-1] != len(x)-1:
```

```
        y = y + [-1]
    # run over all the pairs
    for (a, b) in zip(y[:-1], y[1:]):
        fun(x,a,b)
    return x
# Test cases
questions= [
    "_,_,_,24",
    "40,_,_,_,60",
    "80,_,_,_,_",
     "_,_,30,_,_,_,50,_,_"]

for i in questions:
    print (replace(i))
```

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

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,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 [24]:

```
A = [['F1', 'S1'], ['F2', 'S2'], ['F3', 'S3'], ['F1', 'S2'], ['F2', 'S3'], ['F3', 'S2'],
     ['F2', 'S1'], ['F4', 'S1'], ['F4', 'S3'], ['F5', 'S1']]  # Initialize list as per question

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,}
#Take all possible combinations in dictionary

dictionary2 = {'S1': 0,'S2': 0,'S3': 0}



def compute_conditional_probabilites(A):
    for i in range(len(A)):

        k = A[i][0] + A[i][1] # assign the value of F_s_ into k and increment dictionary 1 [F_S_]
        dictionary1[k] += 1
        dictionary2[A[i][1]] += 1

compute_conditional_probabilites(A)

for key,value in dictionary1.items(): # Print the probabilities accordingly
    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)
```

```
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: 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 [25]:

```
S1=str(input("Enter the First String : "))
S2=str(input("Enter the Second String :"))
S1=S1.split()
S2=S2.split()
a=0
for i in S1:
    for j in S2:
        if i==j:
            a+=1
print(a)
b = (set(S1) - set(S2))
c= (set(S2) - set(S1))
print(b)
print(c)
```

```
Enter the First String : hakesh is working in a product based company
Enter the Second String :hakesh likes to work in ML domain
2
{'product', 'company', 'based', 'working', 'a', 'is'}
{'to', 'domain', 'ML', 'work', 'likes'}
```

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

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

In [27]:

```
from math import log10
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]]
b=0
for i in range(len(A)):
    if A[i][0]==1:
        b+=(-(1))*((A[i][0])*(math.log10(A[i][1])))
    else:
        b+=(-(1))*((1-A[i][0])*(math.log10(1-A[i][1])))
round(b/8, ndigits=3)
```

Out[27]:

```
0.424
```

In [ ]:

```
```