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 37 44 51]]

Ex 3: A   = [[1 2]
             [3 4]]
      B   = [[1 4]
             [5 6]
             [7 8]
             [9 6]]
      A*B =Not possible
```

In [6]:

```
# 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
# you can take matrix input from user or you can directly define the matrix and g
# reference for creating input - https://stackoverflow.com/questions/12293208/how

# you can free to change all these codes/structure
# here A and B are list of lists
def matrix_mul(A, B):
    if len(A)!=len(B[0]): #checking if the matrix multiplication is possible
        return print("A*B =Not possible")

    C = [[0 for i in range(len(A[0]))] for j in range(len(B))] # initializing out
    for i in range(len(A)): #iterating rows of A
        for j in range(len(B[0])): # iterating rows of B
            for k in range(len(B)): #iterating columns of B
                C[i][j] += A[i][k]*B[k][j] #multipying the matrices
    return C

#Checking the results
A   = [[1,3,4],
       [2,5,7],
       [5,9,6]]

B   = [[1,0,0],
       [0,1,0],
       [0,0,1]]

matrix_mul(A, B)
```

Out[6]:

```
[[1, 3, 4], [2, 5, 7], [5, 9, 6]]
```

Q2: Proportional Sampling - 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 experiment
s.
f(100) > f(79) > f(45) > f(28) > f(27) > f(13) > f(10) > f(6) > f(5) > f
(0)
```

In [7]:

```
import random

A = [0,5,27,6,13,28,100,45,10,79]
#Calculating the sum of each items in A
def sum_list(A):
    total = 0
    for x in A:
        total += x
    return total

total = sum_list(A)

#Calculating the probabilities of each items in A
prob = [x / total for x in A]

def pick_a_number_from_list(A, prob):

    num = random.choices(A, weights = prob)

    return num[0]

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

sampling_based_on_magnitude(A)
```

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

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

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

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 [8]:

```python
import re

def replace_digits(String):
    new_string = re.sub("\D","",String)#replacing non digits as empty spaces
    output = re.sub("\d","#",new_string)# replacing digits with #
    return(output)

replace_digits('#2a$#b%c%561#')
```

Out[8]:

```
'####'
```

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, 22, 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','student
6','student7','student8','student9','student10']
Marks = [45, 78, 12, 14, 48, 43, 47, 98, 22, 80]
a.
student8  98
student10 80
student2  78
student5  48
student7  47
b.
student3 12
student4 14
student9 22
student6 43
student1 45
c.
student9 22
student6 43
student1 45
student7 47
student5 48
```

In [102]:

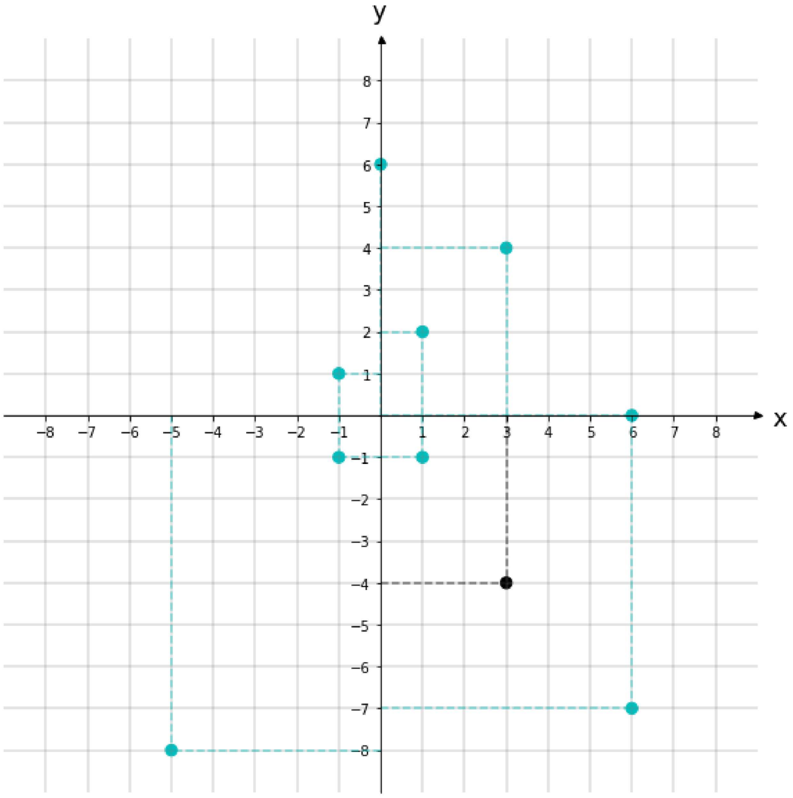
```
students=['student1','student2','student3','student4','student5','student6','stud
marks = [45, 78, 12, 14, 48, 43, 47, 98, 22, 80]


def display_dash_board(students, marks):
    dictionary = dict(zip(students,marks))
    #https://www.tutorialspoint.com/how-to-convert-list-to-dictionary-in-python

    print('top_5_students')
    print(25*"-")
    #iterating over the dictionary after sorting it in ascending order of values
    for key, value in sorted(dictionary.items(), key=lambda item: item[1],reverse
          print("%s: %s" % (key, value))
    #https://www.askpython.com/python/dictionary/sort-a-dictionary-by-value-in-py

    print('least_5_students')
    print(25*"-")
    for key, value in sorted(dictionary.items(), key=lambda item: item[1])[:5]:
          print("%s: %s" % (key, value))

    print('students_within_25_and_75')
    print(25*"-")
    #Calculating the 25th and 75th percentile
    max_mark = max(marks)
    min_mark = min(marks)
    diff_mark = max_mark - min_mark
    per_25 = diff_mark*0.25
    per_75 = diff_mark*0.75
    for key, value in filter(lambda item: int(item[1]) >= per_25 and int(item[1])
        print("%s: %s" % (key, value))


display_dash_board(students, marks)
```

```
top_5_students
-------------------------
student8: 98
student10: 80
student2: 78
student5: 48
student7: 47
least_5_students
-------------------------
student3: 12
student4: 14
student9: 22
student6: 43
student1: 45
students_within_25_and_75
-------------------------
student1: 45
student5: 48
student6: 43
student7: 47
student9: 22
```

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)
```

Hint - If you write the formula correctly you'll get the distance between points (6,-7) and (3,-4) = 0.065

In [65]:

```
import math

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

# here S is list of tuples and P is a tuple ot len=2
def closest_points_to_p(S, P):
    cosine_distance = []
    for i in range (len(S)):
        x = int(S[i][0]) #x cordinates of tuple S
        y = int(S[i][1]) #y coordinates of tuple S
        #Calculating cosine distance using math.acos
        cosine_distance.append(math.acos((x*int(P[0])+y*int(P[1]))/math.sqrt((x**
        #https://www.kite.com/python/answers/how-to-create-a-dictionary-from-two-
        Cosine_distances = zip(S,cosine_distance)
        Cosine_distances_dict = dict(Cosine_distances)

        #Sorting cosine distances in the dictionary using lambda
        sorted_Cosine_distance = dict(sorted(Cosine_distances_dict.items(), key=l

    return sorted_Cosine_distance

sorted_Cosine_distance = closest_points_to_p(S, P)
points = list(sorted_Cosine_distance.items())[:5]
print("Closest points to P")
print(20*"-")
print(points)
```

```
Closest points to P
--------------------
[(('6', '-7'), 0.0651251633343868), (('1', '-1'), 0.14189705460416438), (('6',
'0'), 0.9272952180016123), (('-5', '-8'), 1.202100424136847), (('-1', '-1'), 1.
4288992721907328)]
```

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

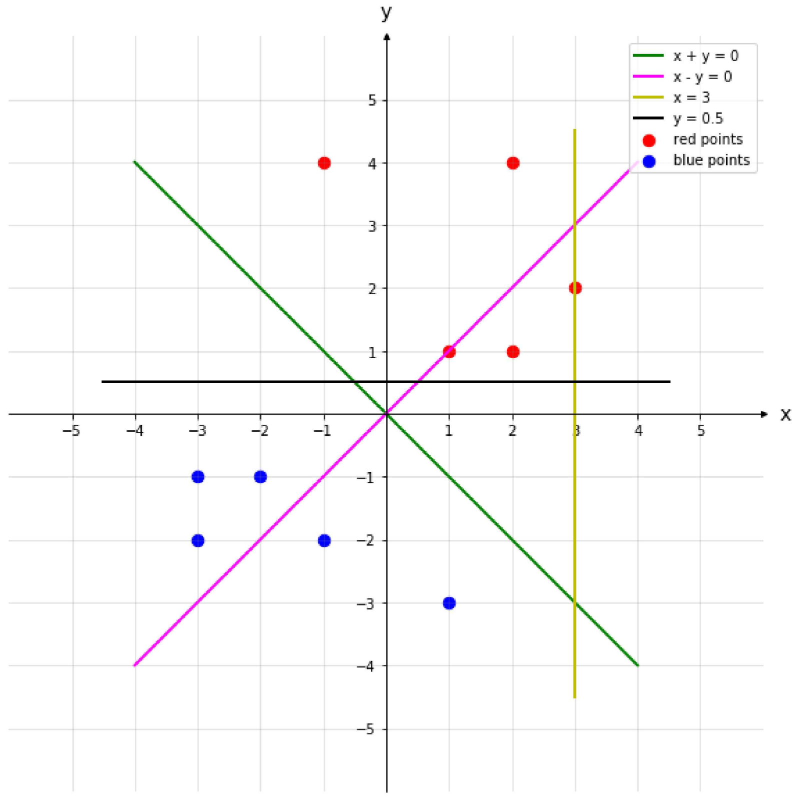
```
Red =[(R11,R12),(R21,R22),(R31,R32),(R41,R42),(R51,R52),..,(Rn1,Rn2)]
Blue=[(B11,B12),(B21,B22),(B31,B32),(B41,B42),(B51,B52),..,(Bm1,Bm2)]
```

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 [75]:

```
import re

#https://stackoverflow.com/questions/57188227/to-find-whether-a-given-line-equati
def i_am_the_one(red,blue,line):

    #Checking if the line completely seperates the Red points
    for i in Red:
        #Replacing coeffient of X with X coefficient of Red point
        Line_1 = line.replace('x','*'+str(i[0]))
        #Replacing coeffient of Y with Y coefficient of Red point
        Line_1 = Line_1.replace('y','*'+str(i[1]))
        #Evaluating the equation
        flag1 = eval(Line_1)
        if flag1 > 0:
            pass
        else:
            return "NO"
     ##Checking if the line completely seperates the Blue points
    for j in Blue:
        #Replacing coeffient of X with X coefficient of Blue point
        Line_2 = line.replace('x','*'+str(i[0]))
        #Replacing coeffient of Y with Y coefficient of Blue point
        Line_2 = Line_2.replace('y','*'+str(i[1]))
        #Evaluating the equation
        flag2 = eval(Line_2)
        if flag2 > 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"]

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

```
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+4
0)/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, 1
0, 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, 1
2, 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 [76]:

```
# refered the code from https://www.kaggle.com/amitalexander/pure-python-exercise
def curve_smoothing(string):
    non_empty_index = []
    #Splitting the string by comma
    S = string.split(',')

    for idx in range(len(S)):
        if S[idx] != "_":
            non_empty_index.append(idx)

    #adding index length
    non_empty_index.append(len(S) - 1)

    #keeping start position as nil and iteraring over the non empty cells to fill

    start = 0
    for ele in non_empty_index:
        #The value to be filled is the sum of values divided by the empty cells i

        cum_sum = int(S[ele]) if S[ele] != "_" else 0
        cum_sum += int(S[start]) if S[start] != "_" and start != ele else 0

        #dividing cum_sum by number of empty cells to get the value to be replace
        #else the numbers to remain the same if the same
        replace_value = cum_sum // (ele - start +1)

        #replacing the empty cells with the replace_value
        S = [replace_value if start <= x <= ele else S[x] for x in range(len(S))]

        #updating start point as the next element in string

        start = ele

    return S


S1 = "_,_,_,24"
S2 = "40,_,_,_,60"
S3 = "80,_,_,_,_"
S4 =  "_,_,30,_,_,_,50,_,_"

smoothed_values1= curve_smoothing(S1)
smoothed_values2= curve_smoothing(S2)
smoothed_values3= curve_smoothing(S3)
smoothed_values4= curve_smoothing(S4)

print("smoothed_values for Input1: _,_,_,24")
print("-"*50)
print(smoothed_values1)

print("smoothed_values for Input2: 40,_,_,_,60")
print("-"*50)
print(smoothed_values2)

print("smoothed_values for Input3: 80,_,_,_,_")
print("-"*50)
print(smoothed_values3)

print("smoothed_values for Input4: _,_,30,_,_,_,50,_,_")
print("-"*50)
print(smoothed_values4)
```

```
smoothed_values for Input1: _,_,_,24
--------------------------------------------------
[6, 6, 6, 6]
smoothed_values for Input2: 40,_,_,_,60
--------------------------------------------------
[20, 20, 20, 20, 20]
smoothed_values for Input3: 80,_,_,_,_
--------------------------------------------------
[16, 16, 16, 16, 16]
smoothed_values for Input4: _,_,30,_,_,_,50,_,_
--------------------------------------------------
[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 [48]:

```
#Initializing numerator and denominator part to calculate conditional probabiliti
num = []
den = []
def compute_conditional_probabilites(A):
    for i in range(len(A)):
        #Making list combining both elements of given list
        k = A[i][0]+A[i][1]
        num.append(k)
        den.append(A[i][1])

A = [['F1','S1'],['F2','S2'],['F3','S3'],['F1','S2'],['F2','S3'],['F3','S2'],['F2
compute_conditional_probabilites(A)
print("P(F=F1|S==S1)=",num.count('F1S1')/den.count('S1'),"P(F=F1|S==S2)=",num.cou
print("P(F=F2|S==S1)=",num.count('F2S1')/den.count('S1'),"P(F=F2|S==S2)=",num.cou
print("P(F=F3|S==S1)=",num.count('F3S1')/den.count('S1'),"P(F=F3|S==S2)=",num.cou
print("P(F=F4|S==S1)=",num.count('F4S1')/den.count('S1'),"P(F=F4|S==S2)=",num.cou
print("P(F=F5|S==S1)=",num.count('F5S1')/den.count('S1'),"P(F=F5|S==S2)=",num.cou
```

```
P(F=F1|S==S1)= 0.25 P(F=F1|S==S2)= 0.3333333333333333 P(F=F1|S==S3)= 0.0
P(F=F2|S==S1)= 0.25 P(F=F2|S==S2)= 0.3333333333333333 P(F=F2|S==S3)= 0.33333333
33333333
P(F=F3|S==S1)= 0.0 P(F=F3|S==S2)= 0.3333333333333333 P(F=F3|S==S3)= 0.333333333
3333333
P(F=F4|S==S1)= 0.25 P(F=F4|S==S2)= 0.0 P(F=F4|S==S3)= 0.3333333333333333
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 [17]:

```
def string_features(S1, S2):
    #Splitting the sentences into words
    #https://www.tutorialspoint.com/common-words-in-two-strings-in-python

    S1_list = S1.split(" ")
    S2_list = S2.split(" ")

    a = len(list(set(S1_list)&set(S2_list)))
    b = list(set(S1_list) - set(S2_list))
    c = list(set(S2_list) - set(S1_list))
    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("Number of common words between S1, S2:",a)
print("Words in S1 but not in S2:",b)
print("Words in S2 but not in S1:",c)
```

```
Number of common words between S1, S2: 7
Words in S1 but not in S2: ['F', 'first', '5']
Words in S2 but not in S1: ['S', '3', 'second']
```

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
```

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

In [21]:

```
import math

def compute_log_loss(A):
    #Number of rows n
    n = len(A)
    sum = 0 # initializing sum as zero
    for i in range (len(A)):
        #using the formula for f(Y,Yscore)
        sum += ((A[i][0] * math.log(A[i][1],10)) + ((1-A[i][0]) * (math.log(1-A[i
    # COmputing log loss
    loss = -(sum/n)
    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.
loss = compute_log_loss(A)

print("log_loss:",loss)
```

```
log_loss: 0.42430993457031635
```

In [ ]: