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 [2]:
```
# 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 examples

# you can free to change all these codes/structure
# here A and B are list of lists


    # write your code
import random
m1=int(input("Enter No. of rows in the first matrix: "))
n1=int(input("Enter No. of columns in the first matrix: "))
a = [[random.random() for col in range(n1)] for row in range(m1)]
for i in range(m1):
    for j in range(n1):
        a[i][j]=int(input())
m2=int(input("Enter No. of rows in the second matrix: "))
n2=int(input("Enter No. of columns in the second matrix: "))
b = [[random.random() for col in range(n2)] for row in range(m2)]
for i in range(m2):
    for j in range(n2):
        b[i][j]=int(input())
c=[[random.random() for col in range(n2)]for row in range(m1)]
if (n1==m2):
    for i in range(m1):
        for j in range(n2):
            c[i][j]=0
            for k in range(n1):
                c[i][j]+=a[i][k]*b[k][j]
            print(c[i][j],'\n',end="")
        print
else:
    print("Multiplication not possible")
```

```
Enter No. of rows in the first matrix: 2
Enter No. of columns in the first matrix: 2
2
2
2
2
Enter No. of rows in the second matrix: 3
Enter No. of columns in the second matrix: 3
2
2
2
2
2
2
2
2
2
Multiplication not possible
```

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 [2]:
```
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 examples
import random
from itertools import accumulate
from bisect import bisect

# you can free to change all these codes/structure
def pick_a_number_from_list(A, n=100):
    # your code here for picking an element from with the probability propotional to its magnitude
    cum_sum = [*accumulate(A)]
    # cum_sum = [0, 5, 32, 38, 51, 79, 179, 224, 234, 313]

    out = []
    for _ in range(n):
        i = random.random()                     # i = [0.0, 1.0)
        idx = bisect(cum_sum, i*cum_sum[-1])    # get index to list A
        out.append(A[idx])

    return out
A = [0,5,27,6,13,28,100,45,10,79]
print(pick_a_number_from_list(A))
```

```
[45, 28, 79, 79, 5, 100, 28, 27, 79, 79, 79, 45, 79, 100, 28, 13, 27, 79, 100, 79, 28, 10, 79, 27, 100, 27, 7
9, 45, 28, 79, 28, 28, 100, 79, 100, 10, 100, 79, 28, 45, 5, 100, 45, 28, 100, 100, 79, 100, 79, 45, 100, 45,
27, 79, 6, 100, 28, 100, 10, 45, 100, 10, 79, 100, 100, 100, 79, 100, 79, 5, 27, 45, 45, 100, 79, 100, 79, 2
7, 45, 79, 28, 100, 28, 79, 79, 28, 27, 27, 79, 45, 79, 100, 79, 27, 13, 27, 27, 45, 45, 100]
```

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 [3]:
```
# 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 examples
import re
# 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
    new=""
    for i in String:
        for x in i:
            if x.isdigit():
                new+='#'*(len(x))
            else:
                new+=""*(len(x))
    return(new) # modified string which is after replacing the # with digits
String=input("Enter the string:")
A=replace_digits(String)
print(A)
```

```
Enter the string:25edrftgyun3456789)
#########
```

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','student9','stud
ent10']
Marks = [45, 78, 12, 14, 48, 43, 47, 98, 35, 80]

a
```

In [200]:
```
# 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 examples
import operator
import math
import functools
# you can free to change all these codes/structure
def display_dash_board(students, marks):
    dictionary = dict(zip(students, marks))
    top_5_students = dict( sorted(dictionary.items(), key=operator.itemgetter(1), reverse=True)[:5])
    least_5_students = dict( sorted(dictionary.items(), key=operator.itemgetter(1), reverse=False)[:5])
    #students_within_25_and_75 = dict( sorted(dictionary.items(), key=operator.itemgetter(1), reverse=False)
[2:7])

    print("a.")
    for k,v in top_5_students.items():
        print(k,v)
    print("------------------------")
    print("b.")
    for k,v in least_5_students.items():
        print(k,v)
    print("------------------------")
    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]
A = display_dash_board(students, marks)
B= students_within_25_and_75(marks, 0.25)
C= students_within_25_and_75(marks, 0.75)
#print(B)
```

```
count=0
dictionary = dict(sorted(dictionary.items(), key=operator.itemgetter(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
```

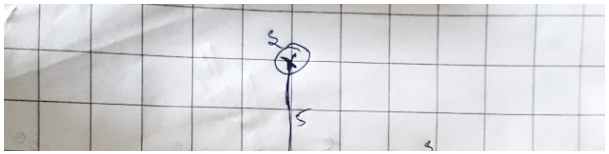
Q5: Find the closest points

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)



In [201]:
```
import math
import numpy as np

# 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 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(set_ofpoints, point):
    # write your code here
    Cosine_Distance=[]
    for i in range(len(set_ofpoints)):
        x = int(set_ofpoints[i][0])
        y = int(set_ofpoints[i][1])
        Cosine_Distance.append(math.acos((x*int(point[0])+y*int(point[1]))/math.sqrt((x**2+y**2)*(int(point[0
])**2+int(point[1])**2))))
        np.argsort(Cosine_Distance[::-1])
    return np.argsort(Cosine_Distance)  # its list of tuples

X= [('1','2'),('3','4'),('-1','1'),('6','-7'),('0','6'),('-5','-8'),('-1','-1'),('6','0'),('1','-1')]
P= ('3','-4')
M=closest_points_to_p(X, P)
[X[i] for i in M][:5] #print the returned values
```

Out[201]:
```
[('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 coefficients of x,y and intercept.
```

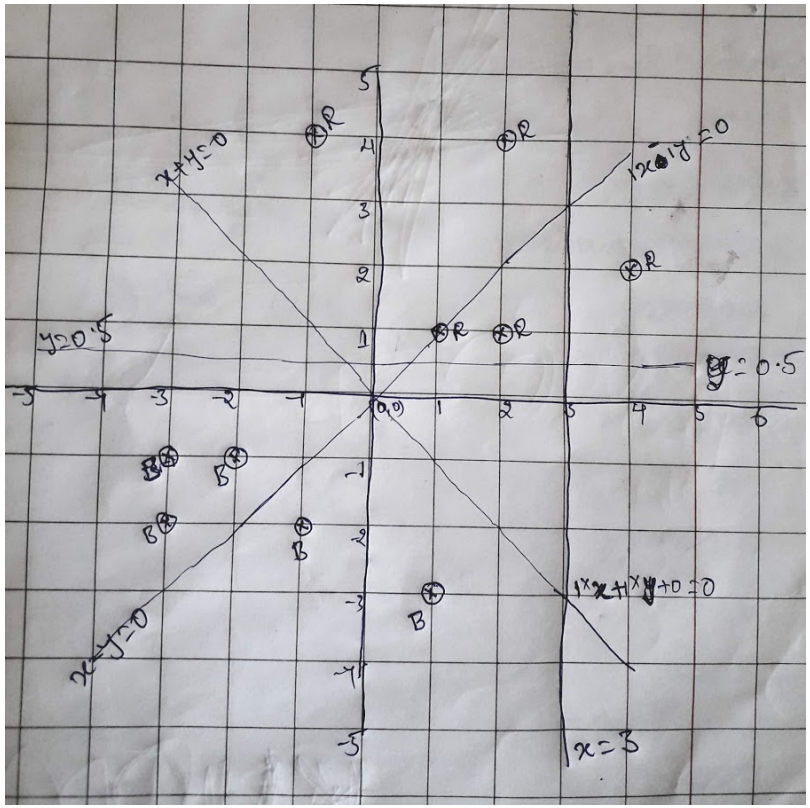
Your task here is to print "YES"/"NO" for each line given. You should print YES, if all the red points are one side of the line and blue points are on other side of the line, otherwise you should print NO.

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 [202]:
```
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 strings


# you can free to change all these codes/structure
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)
```

```
#         print(red_flag_list)
#         print(blue_flag_list)
        if (all(red_flag_list) and all(blue_flag_list)):
                print('Yes')
        else:
                print('No')
        #return #yes/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"]

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

```
Yes
No
No
Yes
```

In [153]:

```
Lines=["1x+1y+0","1x-1y+0","1x+0y-3","0x+1y-0.5"]

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]
print(list_of_coef_clean)
```

```
[[1.0, 1.0, 0.0], [1.0, -1.0, 0.0], [1.0, 0.0, -3.0], [0.0, 1.0, -0.5]]
```

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 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 [3]:
```
# https://stackoverflow.com/questions/57179618/filling-the-missing-values-in-the-specified-format-python/57180882
def curve_smoothing(string):
    a = string.split(',')
    count = 0
    middle_store = 0
    first = 0
    lst = []
    start = 0

    for i in range(len(a)):
        if a[i] == '_':
            count = count + 1  # find number of blanks to the left of a number
        elif count>0:
            middle_store = int(a[i])
            middle_store = (first+middle_store)//(count+1)
            for j in range(start,start+count+1):
                a[j]=middle_store
            start = start +count
            count = 0
            i = i-1
        else:
            first = int(a[i])
            count = count+1
        if i ==len(a)-1:
            if middle_store==0:
                middle_store = (first+middle_store)//(count)
                for j in range(start,len(a)):
                    a[j]=middle_store
            else:
                middle_store=(first+middle_store)//(count+1)
                for j in range(start+1, len(a)):
                    a[j] = middle_store
    return a
```

In [4]:
```
A = "_,_,_,24"
B = "40,_,_,_,60"
C = "80,_,_,_,_"
D = "_,_,30,_,_,_,50,_,_"

a = curve_smoothing(A)
b = curve_smoothing(B)
c = curve_smoothing(C)
d = curve_smoothing(D)

print(a)
print(b)
print(c)
print(d)
```

```
[6, 6, 6, 6]
[20, 20, 20, 20, 20]
[16, 16, 16, 16, 16]
[10, 10, 12, 12, 12, 12, 4, 4, 4]
```

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 [45]:
```
# 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 strings

import math as mt

# you can free to change all these codes/structure
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,
}

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

def compute_conditional_probabilites(A):
    for i in range(len(A)):
        for j in range(i+1,len(A)):
            c=1
            k = A[i][0] + A[i][1]
            dictionary1[k] = c

            if A[i][0]+A[i][1] == A[j][0]+A[j][1]:
                k = A[i][0] + A[i][1]
                c+=1
                dictionary1[k] = c
```

```
        dictionary2[A[i][1]] += 1
    k = A[len(A)-1][0] + A[len(A)-1][1]
    dictionary1[k] = 1

    return dictionary1,dictionary2

    # print the output as per the instructions
A = [['F1','S1'],['F2','S2'],['F3','S3'],['F1','S2'],['F2','S3'],['F3','S2'],['F2','S1'],['F4','S1'],['F4','S
3'],['F5','S1']]
d1,d2 = compute_conditional_probabilites(A)
print(d1,d2)

for key,value in dictionary1.items():
    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)
```

```
{'F1S1': 1, 'F2S1': 1, 'F3S1': 0, 'F4S1': 1, 'F5S1': 1, 'F1S2': 1, 'F2S2': 1, 'F3S2': 1, 'F4S2': 0, 'F5S2':
0, 'F1S3': 0, 'F2S3': 1, 'F3S3': 1, 'F4S3': 1, 'F5S3': 0} {'S1': 4, 'S2': 3, 'S3': 3}
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 [125]:
```
# 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 strings

# you can free to change all these codes/structure
def string_features(S1, S2):
    # your code
    count=0
    res1=0
    for ch in S1.split():
        for ch1 in S2.split():
            if ch == ch1:
                count+=1
    print(count)
def words(S1,S2):
        F= S1.split()
        H = S2.split()
        x = list(set(F) - set(H))
        c = list(set(H) - set(F))
        print(x)
        print(c)
S1= "the first column F will contain only 5 uniques values"
S2= "the second column S will contain only 3 uniques values"
a = string_features(S1, S2)
b = words(S1,S2)
```

```
7
['first', '5', 'F']
['second', '3', 'S']
```

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 $f(Y, Y_{score}) = -1 * \frac{1}{n}\Sigma_{foreach Y, Y_{score} pair}(Y log10(Y_{score}) + (1-Y) log10(1-Y_{score}))$ 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
```

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

In [66]:
```
# 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 strings

import math as mt
# you can free to change all these codes/structure
def compute_log_loss(A):
    # your code
    N=len(A)
    sum=0
    for i in range(len(A)):
        if A[i][0]==1:
            sum+=(-1)*((A[i][0])*(mt.log(A[i][1],10)))
        else:
            sum+=(-1)*((1-A[i][0])*(mt.log(1-A[i][1],10)))
    return sum/N
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
```

In [6]:
```
import random
r1=int(input("ënter no of rows in the 1st matrix:"))
c1=int(input("enter no of columns in the 1st matrix:"))
a=[[random.random() for col in range(c1)] for row in range(r1)]
for i in range (r1):
    for j in range (c1):
        a[i][j]=int (input())


r2=int(input("enter no of rows in the 2nd matrix:"))
c2=int(input("enter no of columns in the 2nd matrix:"))
b=[[random.random() for col in range(c2) ] for row in range(r2)]
for i in range (r2):
    for j in range (c2):
        b[i][j]=int(input())


c=[[random.random() for col in range(c2)] for row in range(r1)]
if (c1==r2):
    for i in range (r1):
        for j in range(c2):
            c[i][j]=0
            print("-----------")
            for k in range (c1):
                c[i][j]+=a[i][k]*b[k][j]

            print(c[i][j],'\n',end="")

        print
else:
    print("multiplication not possible")
```

```
ënter no of rows in the 1st matrix:2
enter no of columns in the 1st matrix:2
1
2
3
4
enter no of rows in the 2nd matrix:2
enter no of columns in the 2nd matrix:2
1
0
0
1
-----------
1
-----------
2
-----------
3
-----------
4
```

In [ ]: