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 [30]:

```
#Matrix multipliaction function
def matrix_mul(A, B):
    #Iterate through row of matrix A
    for i in range(len(A)):
        #Iterate through column of matrix B
        for j in range(len(B[0])):
            #Iterate through row of matrix B
            for k in range(len(B)):
                C[i][j] += A[i][k] * B[k][j]
    #returns the value of C = A*B
    return(C)


#Input Matrix A from user
A_row=int(input("Enter no. of rows of matrix A "))
A_col=int(input("Enter no. of col of matrix A "))

# Initialize matrix A
A = []
print("Enter matrix A entries rowwise:")

# For loop user input matrix A
for i in range(A_row):
    a =[]
    for j in range(A_col):
         a.append(int(input()))
    A.append(a)

B_row=int(input("Enter no. of rows of matrix B "))
B_col=int(input("Enter no. of col of matrix B "))

# Initialize matrix B
B = []
print("Enter matrix B entries rowwise:")

# For user input input matrix B
for i in range(B_row):
    b =[]
    for j in range(B_col):
         b.append(int(input()))
    B.append(b)
#print the Matrix A and B input by the user
print("Matrix A {1}X{2} : {0}" .format(A,len(A),len(A[0])))
print("Matrix B {1}X{2} : {0}" .format(B,len(B),len(B[0])))

# Initialize resultant matrix C
C = [[0 for col in range(len(B[0]))] for row in range(len(A))]

#Matrix multiplication possible only if number of column in matrix A is equal to number
rows in matrix B
if len(A[0]) == len(B):
    print("Matrix A*B {1}X{2} :{0}".format(matrix_mul(A, B),len(C),len(C[0])))
else:
    print("Matrix multiplication A*B not possible")
```

```
Enter no. of rows of matrix A 3
Enter no. of col of matrix A 3
Enter matrix A entries rowwise:
1
3
4
2
5
7
5
9
6
Enter no. of rows of matrix B 3
Enter no. of col of matrix B 3
Enter matrix B entries rowwise:
1
0
0
0
1
0
0
0
1
Matrix A 3X3 : [[1, 3, 4], [2, 5, 7], [5, 9, 6]]
Matrix B 3X3 : [[1, 0, 0], [0, 1, 0], [0, 0, 1]]
Matrix A*B 3X3 :[[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 experiments.
f(100) > f(79) > f(45) > f(28) > f(27) > f(13) > f(10) > f(6) > f(5) > f(0)
```

In [3]:

```
import random

def pick_a_number_from_list(A):
    #calculate the magnitude of each element in list
    sum_num=0
    num_magnitude=[]
    for i in range(len(A)):
        sum_num+=A[i]
    for i in range(len(A)):
        num_magnitude.append(A[i]/sum_num)
    num=random.choices(A,num_magnitude,k=100)
    return num

def sampling_based_on_magnitude():
    number = pick_a_number_from_list(A)
    print("Output : {0}".format(number))
A=[]
n=int(input("enter the size of list A "))
for i in range(n):
    a=input("enter the list of Y : ")
    A.append(int(a))

sampling_based_on_magnitude()
```

```
enter the size of list A 10
enter the list of Y : 0
enter the list of Y : 5
enter the list of Y : 27
enter the list of Y : 6
enter the list of Y : 13
enter the list of Y : 28
enter the list of Y : 100
enter the list of Y : 45
enter the list of Y : 10
enter the list of Y : 79
Output : [5, 100, 100, 100, 79, 27, 100, 28, 79, 79, 100, 100, 27, 28, 10
0, 27, 79, 100, 45, 100, 45, 100, 45, 79, 45, 27, 79, 100, 27, 100, 45, 7
9, 79, 100, 45, 45, 28, 100, 45, 100, 100, 27, 100, 100, 100, 100, 100, 4
5, 27, 79, 45, 27, 100, 100, 79, 100, 100, 100, 100, 100, 13, 27, 79, 100,
100, 100, 79, 100, 28, 27, 27, 100, 10, 28, 100, 45, 28, 79, 45, 13, 45, 2
7, 100, 100, 100, 100, 79, 79, 28, 28, 79, 100, 100, 28, 79, 45, 13, 10, 2
8, 79]
```

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]:

```
import re
#Replace digit fuction which replaces all digits with #
def replace_digits(list):
    pattern = '[0-9]'
    list = [re.sub(pattern, '#', i) for i in list]
    result=' '
    return (result.join(list))

#get input string from user
s=str(input("Enter a string "))
slist=list(s)
#this function replaces all characters which are not digits with spaces
s1 =[re.sub("\D","",s)]
#calls replace digit function which replaces all digits with #
result=replace_digits(s1)
print("The modified string after replacing the digits with # :  {0}".format(result))
```

```
Enter a string #2a$#b%c%561#
The modified string after replacing the digits with # :  ####
```

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','stu
dent7','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]:

```
def display_dash_board(students, marks):
    #create name and marlist dictionary
    stud_mark_dict = dict(zip(name_list,mark_list))
    #print("Student and mark list : ",stud_mark_dict)
    #sort list with desc order of mark
    desc_stud_mark=sorted(stud_mark_dict, key=stud_mark_dict.get, reverse=True)[:5]

    print("Students who got top 5 ranks: ")
    for i in desc_stud_mark:
        print("{0}\t{1}".format(i,stud_mark_dict[i]))

    #sort list with asc order of mark
    asc_stud_mark=sorted(stud_mark_dict, key=stud_mark_dict.get)[:5]

    print("Students who got least 5 ranks: ")
    for i in asc_stud_mark:
        print("{0}\t{1}".format(i,stud_mark_dict[i]))

    #calculate percentile of mark list
    percentiles=[float(int(value) - min_value) / (max_value - min_value) * 100
                 for value in mark_list]

    #dictionary with student name and percentiles
    stud_percentile_dict = dict(zip(name_list,percentiles))
    #print("Student percentile list: ",stud_percentile_dict)

    #sort asc order of student percentile
    asc_stud_percentile=sorted(stud_percentile_dict, key=stud_percentile_dict.get)
    #print("asc student percentile:",asc_stud_percentile)

    print("Students with percentile >25 and <75: ")
    for i in asc_stud_percentile:
        if float(stud_percentile_dict[i]) > 25 and float(stud_percentile_dict[i]) < 75:
            print("{0}\t{1}".format(i,stud_mark_dict[i]))
    return

#Initialize name list and mark list
name_list = []
mark_list = []

#get input of size of name list and mark list
name_list_size = int(input("Enter total elements(greater than 5) for the name list : "
))
mark_list_size = int(input("Enter total elements for the mark list(equal to elements in
name list) : "))

#get input name list from user
for i in range(name_list_size):
    name_list.append(input("Enter value for the name list : "))
#get input mark list from user
for i in range(mark_list_size):
    mark_list.append(input("Enter value for the mark list : "))
#print obtained mark list and name list
print("list of names : ",name_list)
print("list of marks : ",mark_list)

#calculate min and max marks
min_value = int(min(mark_list))
max_value = int(max(mark_list))
```

```
##top_5_students, least_5_students, students_within_25_and_75,call function display_das
h_board(name_list, mark_list)
display_dash_board(name_list, mark_list)
```

```
Enter total elements(greater than 5) for the name list : 10
Enter total elements for the mark list(equal to elements in name list) : 1
0
Enter value for the name list : s1
Enter value for the name list : s2
Enter value for the name list : s3
Enter value for the name list : s4
Enter value for the name list : s5
Enter value for the name list : s6
Enter value for the name list : s7
Enter value for the name list : s8
Enter value for the name list : s9
Enter value for the name list : s10
Enter value for the mark list : 45
Enter value for the mark list : 78
Enter value for the mark list : 12
Enter value for the mark list : 14
Enter value for the mark list : 48
Enter value for the mark list : 43
Enter value for the mark list : 47
Enter value for the mark list : 98
Enter value for the mark list : 35
Enter value for the mark list : 80
list of names :  ['s1', 's2', 's3', 's4', 's5', 's6', 's7', 's8', 's9', 's
10']
list of marks :  ['45', '78', '12', '14', '48', '43', '47', '98', '35', '8
0']
Students who got top 5 ranks:
s8      98
s10     80
s2      78
s5      48
s7      47
Students who got least 5 ranks:
s3      12
s4      14
s9      35
s6      43
s1      45
Students with percentile >25 and <75:
s9      35
s6      43
s1      45
s7      47
s5      48
```

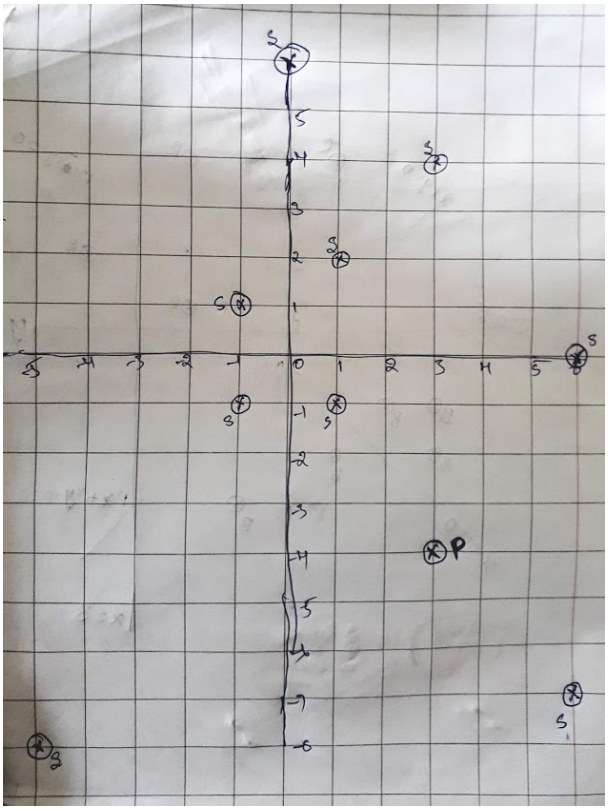
Q5: Find the closest points

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

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 [20]:

```
import math

def closest_points_to_p(s, p):
    for i in range(len(s)):
        j=1
        dis=s[i][0]-p1[0][0]+s[i][j]-p1[0][1]
        cal1=(math.sqrt(math.pow(s[i][0],2)+math.pow(s[i][j],2)))*(math.sqrt((math.pow(
p1[0][0],2)+math.pow(p1[0][1],2))))
        cal=dis/cal1
        dis1.append(abs(round(cal,4)))

        dis1_dict = dict(zip(s,dis1))
        desc_dis1_dict=sorted(dis1_dict, key=dis1_dict.get)[:5]
    return desc_dis1_dict

s=[]
p1=[]
dis1=[]

n=int(input("Enter the number of coordinate points : "))

for i in range(n):
    t= tuple([eval(x) for x in input("enter the values: ").split(',')])
    s.append(t)
print(s)

p= tuple([eval(x) for x in input("enter the coordinate point : ").split(',')])
p1.append(p)
print(p1)

points = closest_points_to_p(s, p)
print("Five closest points :")
for ele in points:
    print(ele)
```

```
Enter the number of coordinate points : 9
enter the values: 1,2
enter the values: 3,4
enter the values: -1,1
enter the values: 6,-7
enter the values: 0,6
enter the values: -5,-8
enter the values: -1,-1
enter the values: 6,0
enter the values: 1,-1
[(1, 2), (3, 4), (-1, 1), (6, -7), (0, 6), (-5, -8), (-1, -1), (6, 0), (1,
-1)]
enter the coordinate point : 3,-4
[(3, -4)]
Five closest points :
(6, -7)
(-1, 1)
(-1, -1)
(1, -1)
(0, 6)
```

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

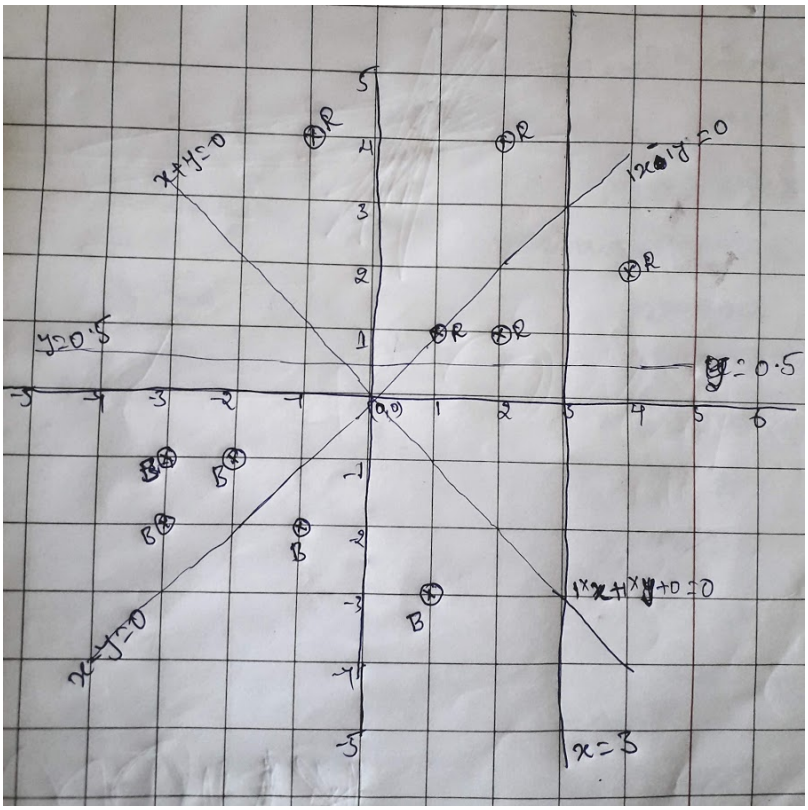
```
Red =[(R11,R12),(R21,R22),(R31,R32),(R41,R42),(R51,R52),..,(Rn1,Rn2)]
Blue=[(B11,B12),(B21,B22),(B31,B32),(B41,B42),(B51,B52),..,(Bm1,Bm2)]
```

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 int
ercept.
```

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 [2]:

```
import re

def i_am_the_one(p1,q1,Eq):
    R=[]
    #Reg ex to extract the coef of line equation
    for i in Eq:
        r=re.findall(r'[\d\.\-\+]+', i)
        R.append(r)

        R_points=[]
        B_points=[]
     #distance from a point on a line check if positive or negative,distance formula a
x'+by'+c/sqrt(a2+b2)
    for i in range(len(R)):
        R_points=[]
        B_points=[]
        for j in range(len(p1)):
            r_points=float(R[i][0])*float(p1[j][0])+float(R[i][1])*float(p1[j][1])+floa
t(R[i][2])
            if float(r_points)>0.0:
                calc="P"
                R_points.append(calc)
            else:
                calc="N"
                R_points.append(calc)
        for j in range(len(q1)):
            b_points=float(R[i][0])*float(q1[j][0])+float(R[i][1])*float(q1[j][1])+floa
t(R[i][2])
            if float(b_points)>0.0:
                calc="P"
                B_points.append(calc)
            else:
                calc="N"
                B_points.append(calc)
        R_P_count=0
        R_N_count=0
        B_P_count=0
        B_N_count=0
        for i in R_points:
            if i=='P':
                R_P_count+=1
            else:
                R_N_count+=1
        for i in B_points:
            if i=='P':
                B_P_count+=1
            else:
                B_N_count+=1
        print("Output : ")
        if R_P_count==len(p1) and B_N_count == len(q1):
            print("YES")
        else:
            print("NO")
    return

Eq=[]
p1=[]
q1=[]
n=int(input("enter the size of line equation list Eq "))
```

```
for i in range(n):
    eq=input("enter the list of Eq : ")
    Eq.append(eq)
print(Eq)

m=int(input("enter no of points in list RED "))
for i in range(m):
    p= tuple([eval(x) for x in input("enter the coordinate point for list RED : ").spli
t(',')])
    p1.append(p)
print(p1)

o=int(input("enter no of points in list blue "))
for i in range(o):
    q= tuple([eval(x) for x in input("enter the coordinate point for List BLUE : ").spl
it(',')])
    q1.append(q)
print(q1)
#function call p1-red points,q1-blue points,Eq-equation of line
i_am_the_one(p1, q1, Eq)
```

```
enter the size of line equation list Eq 4
enter the list of Eq : 1x+1y+0
enter the list of Eq : 1x-1y+0
enter the list of Eq : 1x+0y-3
enter the list of Eq : 0x+1y-0.5
['1x+1y+0', '1x-1y+0', '1x+0y-3', '0x+1y-0.5']
enter no of points in list RED 5
enter the coordinate point for list RED : 1,1
enter the coordinate point for list RED : 2,1
enter the coordinate point for list RED : 4,2
enter the coordinate point for list RED : 2,4
enter the coordinate point for list RED : -1,4
[(1, 1), (2, 1), (4, 2), (2, 4), (-1, 4)]
enter no of points in list blue 5
enter the coordinate point for List BLUE : -2,-1
enter the coordinate point for List BLUE : -1,-2
enter the coordinate point for List BLUE : -3,-2
enter the coordinate point for List BLUE : -3,-1
enter the coordinate point for List BLUE : 1,-3
[(-2, -1), (-1, -2), (-3, -2), (-3, -1), (1, -3)]
Output :
YES
Output :
NO
Output :
NO
Output :
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 equ
ally 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 place
s

Ex 3: 80, _, _, _, _  ==> 80/5,80/5,80/5,80/5,80/5 ==> 16, 16, 16, 16, 16 i.e. t
he 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, 1
2, 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]:

```
#functon to fill the series
def curve_smoothing(string):
    count=0
    split=0
    scount=0
    value=0
    spos=1
    # for loop to iterate the entire list of s1 elements
    for ele in s1:
        #check if element in s1 is '-' or has a value
        if ele == '_':
            count+=1
            scount+=1
            #when list s1 ends with '-'
            if count==len(s1):
                count1=0
                #iterate s1 in reverse and populate the series
                for i in range(len(s1),0,-1):
                    j=i-1
                    if s1[j] == '_':
                        count1+=1
                    else:
                        count1+=1
                        split1=int(value)/count1
                        for i in range(j,len(s1),1):
                            s1[i]=int(split1)
                        break
        else:
            # when element in s1 is not '-'
            count+=1
            scount+=1
            value=int(value)+int(ele)
            split=int(value)/scount
            for i in range(spos-1,count):
                s1[i]=int(split)
            value=int(split)
            scount=1
            spos=count
    str1=','.join(map(str,s1))
    return  str1


s=input("Enter the string sequence : ")
s1=s.split(',')
filled_str=smoothed_values= curve_smoothing(s1)
print("output : {0}".format(filled_str))
```

```
Enter the string sequence : _,_,30,_,_,_,50,_,_
output : 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,S
1]]

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 [1]:

```
def compute_conditional_probabilites(A):
    for i in range(len(A)):
        k = A[i][0] + A[i][1]
        #increments the count and updates the dict list1 and list2
        list1_dict[k] += 1
        list2_dict[A[i][1]] += 1
    print('a. P(F=F1|S==S1)= {0}/{1},P(F=F1|S==S2)= {2}/{3},P(F=F1|S==S3)= {4}/{5}'
          .format(list1_dict['F1S1'],list2_dict['S1'],list1_dict['F1S2'],list2_dict['S
2'],
                  list1_dict['F1S3'],list2_dict['S3']))
    print('b. P(F=F2|S==S1)= {0}/{1},P(F=F2|S==S2)= {2}/{3},P(F=F2|S==S3)= {4}/{5}'
          .format(list1_dict['F2S1'],list2_dict['S1'],list1_dict['F2S2'],list2_dict['S
2'],
                  list1_dict['F2S3'],list2_dict['S3']))
    print('c. P(F=F3|S==S1)= {0}/{1},P(F=F3|S==S2)= {2}/{3},P(F=F3|S==S3)= {4}/{5}'
          .format(list1_dict['F3S1'],list2_dict['S1'],list1_dict['F3S2'],list2_dict['S
2'],
                  list1_dict['F3S3'],list2_dict['S3']))
    print('d. P(F=F4|S==S1)= {0}/{1},P(F=F4|S==S2)= {2}/{3},P(F=F4|S==S3)= {4}/{5}'
          .format(list1_dict['F4S1'],list2_dict['S1'],list1_dict['F4S2'],list2_dict['S
2'],
                  list1_dict['F4S3'],list2_dict['S3']))
    print('e. P(F=F5|S==S1)= {0}/{1},P(F=F5|S==S2)= {2}/{3},P(F=F5|S==S3)= {4}/{5}'
          .format(list1_dict['F5S1'],list2_dict['S1'],list1_dict['F5S2'],list2_dict['S
2'],
                  list1_dict['F5S3'],list2_dict['S3']))
A_row=10
A_col=2
A=[]

print("Enter the values of list A rowwise:")

# For loop user input list A
for i in range(A_row):
    a =[]
    for j in range(A_col):
        b=str(input())
        a.append(b)
    A.append(a)


list1_dict = {'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
}

list2_dict= {'S1':0,'S2':0,'S3':0
}

#calling the function
compute_conditional_probabilites(A)
#used some code references from stackoverflow
```

```
Enter the values of list A rowwise:
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
```

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 [1]:

```
def string_features(s1, s2):
    s1_list=s1.split()
    s2_list=s2.split()
    s3=[]
    count=0
    for i in s1_list:
        for j in s2_list:
            if i==j:
                count+=1
                s3.append(i)
    for i in s3:
        s1_list.remove(i)
        s2_list.remove(i)

    return count,s1_list,s2_list

s1= input("Enter the first sentence :\n")
s2= input("Enter the second sentence :\n")

a,b,c= string_features(s1,s2)
print("a. {0}".format(a))
print("b. {0}".format(b))
print("c. {0}".format(c))
```

```
Enter the first sentence :
the first column F will contain only 5 unique values
Enter the second sentence :
the second column S will contain only 3 unique values
a. 7
b. ['first', 'F', '5']
c. ['second', 'S', '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
$f(Y, Y_{score}) = -1 * \frac{1}{n}\Sigma_{foreachY,Y_{score}pair}(Ylog10(Y_{score}) + (1-Y)log10(1-Y_{score}))$ 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
```

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

In [1]:

```
import math

def compute_log_loss(Input):
    calc=0.0
    output=0.0
    a=0.0
    b=0.0
    for i in range(n):
        a=0.0
        b=0.0
        a=(int(Input[i][0]))*(round(math.log10(Input[i][1]),5))
        b=(1-int(Input[i][0]))*(round(math.log10(1-Input[i][1]),5))
        calc+=(a+b)
    output=-1*(calc/8)
    return output

Y=[]
Yscore=[]
Input=[]
ip=[]
#size of the input list
#get input list of Y and corresponding Yscore from user
n=int(input("enter the size of list Y "))
for i in range(n):
    y=input("enter the list of Y : ")
    Y.append(int(y))
a=n
for i in range(a):
    ys=input("enter the list of corresponding Y_Score :")
    Yscore.append(float(ys))

for j in range(int(a)):
    ip=[]
    ip.append(Y[j])
    ip.append(Yscore[j])
    Input.append(ip)
print(Input)
#call the function log loss
loss=0.0
loss = compute_log_loss(Input)
print("Output: {0}".format(loss))
```

```
enter the size of list Y 8
enter the list of Y : 1
enter the list of Y : 0
enter the list of Y : 0
enter the list of Y : 0
enter the list of Y : 0
enter the list of Y : 1
enter the list of Y : 1
enter the list of Y : 1
enter the list of corresponding Y_Score :0.4
enter the list of corresponding Y_Score :0.5
enter the list of corresponding Y_Score :0.9
enter the list of corresponding Y_Score :0.3
enter the list of corresponding Y_Score :0.6
enter the list of corresponding Y_Score :0.1
enter the list of corresponding Y_Score :0.9
enter the list of corresponding Y_Score :0.8
[[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: 0.42431
```

In [ ]: