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
```

In [12]:
```
def readMatrix(rows,cols,matrix1):
    print("enter the matrix row by row:")

    for row in range(rows):
        rowVals = [int(marks) for marks in input().split()]
        if len(rowVals) != cols:
            print("Wrong column length,quitting the program")
            return False
        matrix1.append(rowVals)

    return True




m1rows = int(input("Enter the number of rows for matrix1:"))
m1cols = int(input("Enter the number of columns for maxtrix1:"))

m2rows = int(input("Enter the number of rows for matrix2:"))
m2cols = int(input("Enter the number of columns for maxtrix2:"))

if m1cols != m2rows:
    print('''Error:number of columns in first matrix should be
             equal to rows in second matrix''')
    quit()

matrix1 = []
matrix2 = []

if readMatrix(m1rows,m1cols,matrix1) == False or readMatrix(m2rows,m2cols,matr
ix2) == False:
    quit()


product=[]

for i in range(m1rows):
    row=[]
    res=0
    for j in range(m2cols):
        res=0
        for k in range(m2rows):
            res += matrix1[i][k]*matrix2[k][j]
            if k == m2rows-1:
                row.append(res)
                #print(res)
                #print(row)
                #res=0
    product.append(row)



print(product)
```

```
Enter the number of rows for matrix1:2
Enter the number of columns for maxtrix1:2
Enter the number of rows for matrix2:2
Enter the number of columns for maxtrix2:2
enter the matrix row by row:
1 2
3 4
enter the matrix row by row:
1 2
3 4
[[7, 10], [15, 22]]
```

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 [11]:
```
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 inp
ut examples


# you can free to change all these codes/structure
def pick_a_number_from_list(A):
    s= sum(A)
    weights=[]
    for val in A:
        weights.append(val/s)
    #print(weights)

    #the below logic is taken from stackoverflow
    cummulativeSum=[0 for i in range(len(weights))]
    cummulativeSum[0]=0
    for val in range(len(weights)-1):
        cummulativeSum[val+1]=cummulativeSum[val]+weights[val+1]

    rnum = uniform(0.0,1.0)
    number =0
    for i in range(len(cummulativeSum)):
        if(rnum<=cummulativeSum[i]):
            number=A[i]
            break

    return number

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

A=[int(val) for val in input().split()]
sampling_based_on_magnitued(A)
```

```
0 5 27 6 13 28 100 45 10 79
79
79
100
79
79
27
28
100
45
100
100
45
100
45
79
100
100
100
100
79
45
100
27
45
100
27
79
79
6
45
100
100
28
79
27
100
79
5
100
45
5
45
28
79
100
28
28
27
28
100
28
28
28
79
79
13
```

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

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 [10]:
```
# Online Python compiler (interpreter) to run Python online.
# Write Python 3 code in this online editor and run it.
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 inp
ut examples

# you can free to change all these codes/structure
# String: it will be the input to your program
ip =input ("Enter String :")
String  = str(ip)

def replace_digits(String):
    s=''
    for c in String:

        if c.isdigit():
            s+='#'

    return s # modified string which is after replacing the # with digits

print(replace_digits(String))
```

```
Enter String :#2a$#b%c%561#
####
```

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','studen
t7','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]:
```
import operator



# you can free to change all these codes/structure
def display_dash_board(students, marks):
    #print(len(students))
    #print(len(marks))

    markList = dict()


    mLen = len(students)

    if mLen < 5:
        print("Warning: Number of students is less than 5 so only top "
              +str(mLen)+" and bottom "+str(mLen)+" students marks will be dis
played")



    #build the dictionary with students and marks data
    for i in range(len(students)):
        markList[students[i]]=marks[i]
    mlPairs = dict(sorted(markList.items(),key=operator.itemgetter(1)));
    mlPairsRev = dict(sorted(markList.items(),key=operator.itemgetter(1),rever
se=True));


   #write code for computing top top 5 and least 5 students
    top_5_students = dict()
    least_5_students = dict()

    index=0
    leastMark=0
    for pair in mlPairs.items():
        if index==0:
            leastMark=pair[1]
        if  index  <  5:
            least_5_students[pair[0]]=pair[1]
        index +=1

    index =0
    topMark=0
    for pair in mlPairsRev.items():
        if index==0:
            topMark=pair[1]
        if  index  <  5:
            top_5_students[pair[0]]=pair[1]
        index +=1

    diff = topMark-leastMark
    pre_25=diff*0.25
    pre_75=diff*0.75

    # write code for computing students between 25 and 75
```

```
    #filter out students marks > 25th percentile and then filter out
    #students with marks < 75th percentile
    students_gt_25 = {student:score for student,score in markList.items() if s
core>pre_25}
    students_within_25_and_75 = {student:score for student,score in students_g
t_25.items() if score<pre_75}



    return top_5_students, least_5_students, students_within_25_and_75



#driver code below
print("enter student name seperated by space:")

#input reading logic taken from stackoverflow
#https://stackoverflow.com/questions/4663306/get-a-list-of-numbers-as-input-fr
om-the-user
students = [str(student) for student in input().split()]

print("enter marks name seperated by space:")

Marks = [int(marks) for marks in input().split()]

if len(students) != len(Marks):
        print("Error:count of studends and marks should be equal")
        quit()

top_5_students, least_5_students, students_within_25_and_75 = display_dash_boa
rd(students, Marks)
print("top 5 students:")
print(top_5_students)
print("least 5 students:")
print(least_5_students)
print("students who scored between 25th percentile and 75th percentile:")
print(students_within_25_and_75)
```

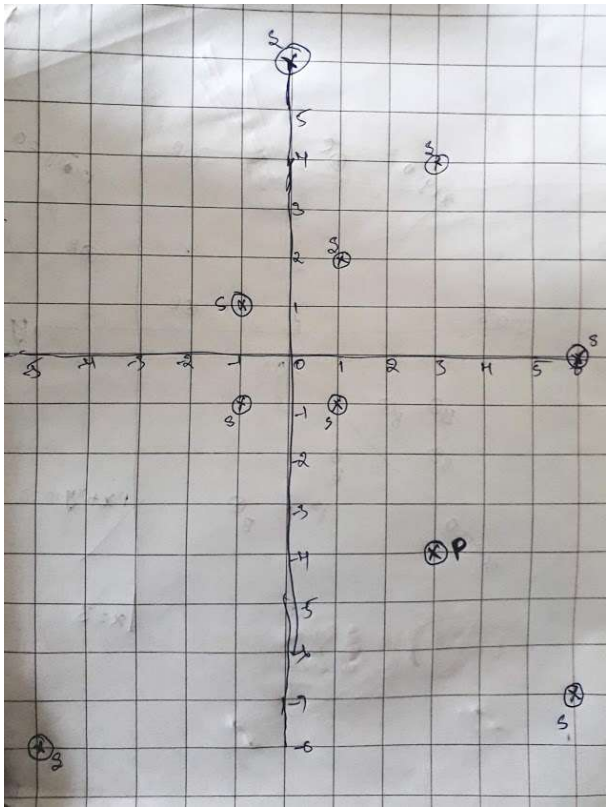
```
enter student name seperated by space:
s1 s2 s3 s4 s5 s6
enter marks name seperated by space:
45 78 12 23 25 80
top 5 students:
{'s6': 80, 's2': 78, 's1': 45, 's5': 25, 's4': 23}
least 5 students:
{'s3': 12, 's4': 23, 's5': 25, 's1': 45, 's2': 78}
students who scored between 25th percentile and 75th percentile:
{'s1': 45, 's4': 23, 's5': 25}
```

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)

In [8]:
```
import math
import operator

# 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 inp
ut 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(S, P):
    distances =dict()
    # write your code here
    for i in range(len(S)):
        tup = S[i]
        numerator = P[0]*tup[0]+P[1]*tup[1]
        denominator= math.sqrt(tup[0]**2+tup[1]**2)*math.sqrt(P[0]**2+P[1]**2)
        distances[S[i]]=math.acos(numerator/denominator)


    sortedDistances=dict(sorted( distances.items(),key=operator.itemgetter(1
)));

    closestPoints = list(sortedDistances.keys())
    return closestPoints[0:5]

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

S=[]

nps = int(input("enter number of points in s:"))

if nps<5:
    print("Number of points should be atleast 5")
    quit()

for i in range(nps):
    print("enter coordinates for point "+str(i+1))
    x=int(input("enter x coordinate:"))
    y=int(input("enter y coordinate:"))
    S.append((x,y))


px=int(input("enter x coordinate for point p:"))
py=int(input("enter y coordinate for point p:"))

P =(px,py)


points = closest_points_to_p(S, P)
print(points) #print the returned values
```

```
enter number of points in s:9
enter coordinates for point 1
enter x coordinate:1
enter y coordinate:2
enter coordinates for point 2
enter x coordinate:3
enter y coordinate:4
enter coordinates for point 3
enter x coordinate:-1
enter y coordinate:1
enter coordinates for point 4
enter x coordinate:6
enter y coordinate:-7
enter coordinates for point 5
enter x coordinate:0
enter y coordinate:6
enter coordinates for point 6
enter x coordinate:-5
enter y coordinate:-8
enter coordinates for point 7
enter x coordinate:-1
enter y coordinate:-1
enter coordinates for point 8
enter x coordinate:6
enter y coordinate:0
enter coordinates for point 9
enter x coordinate:1
enter y coordinate:-1
enter x coordinate for point p:3
enter y coordinate for point p:-4
[(6, -7), (1, -1), (6, 0), (-5, -8), (-1, -1)]
```

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

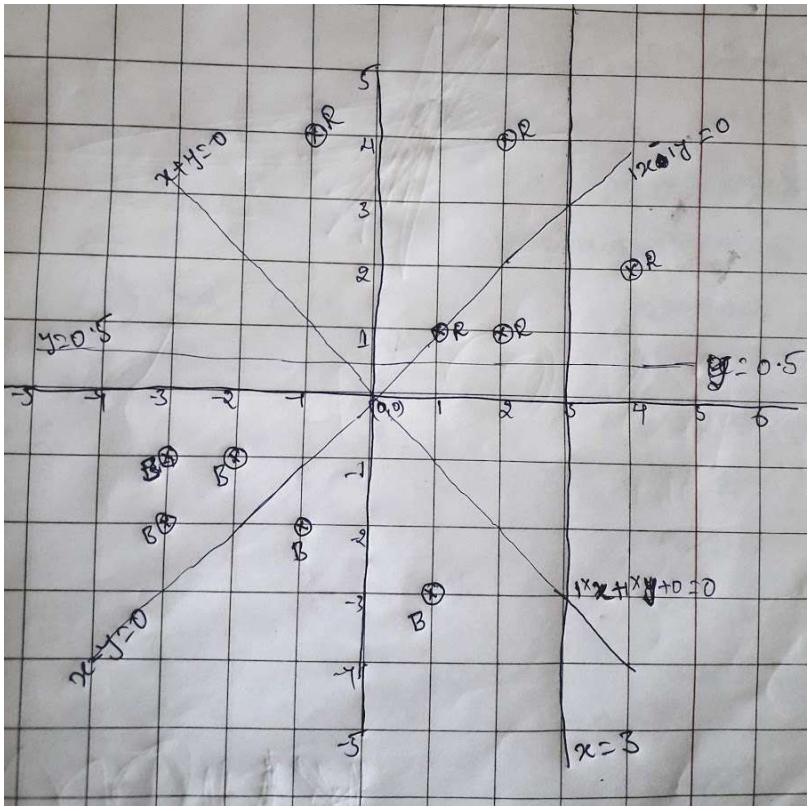
```
Red =[(R11,R12),(R21,R22),(R31,R32),(R41,R42),(R51,R52),..,(Rn1,Rn2)]
Blue=[(B11,B12),(B21,B22),(B31,B32),(B41,B42),(B51,B52),..,(Bm1,Bm2)]
```

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 [7]:
```
import math
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 inp
ut strings

def getCoeffs(line):
    #taken below line from stackoverflow
    #https://stackoverflow.com/questions/56948506/how-to-extract-coefficients-
from-a-line-equation-in-python-without-using-numpy
    coeffs= [float(i) for i in re.split('[xy]', line)]
    return coeffs[0],coeffs[1],coeffs[2]


def isAboveLine(x,y,line):
    a,b,c=getCoeffs(line)
    if ((x*a)+(b*y)+(c)) > 0 and b > 0:
        return True
    if ((x*a)+(b*y)+(c)) < 0 and b < 0:
        return True

    return False
# you can free to change all these codes/structure
def i_am_the_one(red,blue,line):

    rabove=0
    rbelow=0
    for r in red:
        if isAboveLine(r[0],r[1],line):
            rabove += 1
        else:
            rbelow += 1

    babove=0
    bbelow=0
    for b in blue:
        if isAboveLine(b[0],b[1],line):
            babove += 1
        else:
            bbelow += 1

    if bbelow > 0 and babove > 0:
        return "NO"
    if rbelow > 0 and rbelow > 0:
        return "NO"

    if rabove == len(red) and bbelow == len(blue):
        return "YES"
    if rbelow == len(red) and babove == len(blue):
        return "YES"

    # your code
    return "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"]

nr=int(input("enter number of points for red:"))
Red=[]
for i in range(nr):
    x=float(input("enter x coordinate for point "+str(i+1)+":"))
    y=float(input("enter y coordinate for point "+str(i+1)+":"))
    Red.append((x,y))

nb=int(input("enter number of points for blue:"))
Blue=[]
for i in range(nb):
    x=float(input("enter x coordinate for point "+str(i+1)+":"))
    y=float(input("enter y coordinate for point "+str(i+1)+":"))
    Blue.append((x,y))

Lines=[]
nl=int(input("enter number of lines:"))
for i in range(nl):
    line=input("enter equation for line"+str(i+1)+":")
    Lines.append(line)


for i in Lines:
    yes_or_no = i_am_the_one(Red, Blue, i)
    print(yes_or_no) # the returned value
```

```
enter number of points for red:5
enter x coordinate for point 1:1
enter y coordinate for point 1:1
enter x coordinate for point 2:2
enter y coordinate for point 2:1
enter x coordinate for point 3:4
enter y coordinate for point 3:2
enter x coordinate for point 4:2
enter y coordinate for point 4:4
enter x coordinate for point 5:-1
enter y coordinate for point 5:4
enter number of points for blue:5
enter x coordinate for point 1:-2
enter y coordinate for point 1:-1
enter x coordinate for point 2:-1
enter y coordinate for point 2:-2
enter x coordinate for point 3:-3
enter y coordinate for point 3:-2
enter x coordinate for point 4:-3
enter y coordinate for point 4:-1
enter x coordinate for point 5:1
enter y coordinate for point 5:-3
enter number of lines:4
enter equation for line1:1x+1y+0
enter equation for line2:1x-1y+0
enter equation for line3:1x+0y-3
enter equation for line4:0x+1y-0.5
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 equall
y 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, 1
2, 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 [1]:
```
#from collections import deque
# 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 inp
ut strings-

# you can free to change all these codes/structure
def curve_smoothing(string):
    #print(string)
    #take the elements into list 'delimited'
    delimited=string.split(',')
    #print(delimited)
    i=0
    k=0
    avg=0
    #stack = deque()
    #start and end for replacing the underscores with right value
    start=0
    end=0
    while i < len(delimited):
        start=end
        end=0
        count=0
        #detect first '_' and count subsequent underscores
        if delimited[i] == '_':
            k=i
            while k < len(delimited):
                if delimited[k]!='_':
                    break
                #stack.append(delimited[k])
                count +=1
                k += 1
        else:
            count=0
            #stack.clear()
            i +=1
            continue

        #if contigous underscores are found
        #find the start and end element to calculate the average
        if count > 0:
            if delimited[i-1] !='_':
                if start==0 and i!=0:
                    start=int(delimited[i-1])
            if k < len(delimited) and delimited[k] !='_' :
                end=int(delimited[k])

        #calculate the average value to replace with '_'
        tc=count+(1 if start !=0 else 0)+(1 if end!=0 else 0)
        avg = 0
        if tc !=0:
            avg=(start+end)/tc
        #print(stack)
        #print(avg)
        #print(start)
        #print(end)
```

```
        #print(tc)
        #print(i)
        #print(k)
        #adjust and low and high
        #Low and high are used to replace the
        #values correctly for underscores
        low=i-1
        high=k
        if i==0:
            low=i
        if k==len(delimited):
            high=k-1
        ri=low
        #print(low)
        #print(how)
        #replace the underscores,
        #ri =>replace Index
        while ri <= high:
            delimited[ri]=avg
            ri +=1
        #stack.clear()
        #print(delimited)
        if k !=0:
            i=k
        else:
            i +=1

    return delimited

#S=  "_,_,30,_,_,_,50,_,_"
#S="_,_,_,24"
#S="80,_,_,_,_"
#S="40,_,_,_,60"
#S="_"
S=input("enter the string to be smoothed:")
smoothed_values= curve_smoothing(S)
string=""
i=0
while i<len(smoothed_values):
    string +=str(int(smoothed_values[i]))
    i +=1
    if i < len(smoothed_values):
        string +=","


print(string)
```

```
enter the string to be smoothed:_,_,_,24
6,6,6,6
```

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 [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 inp
ut strings
def printProbablity(index,f1,s1,prob1,f2,s2,prob2,f3,s3,prob3):
    p1str="{0}. P(F=={1}|S=={2})={3}), "
    p2str="P(F=={4}|S=={5})={6}), "
    p3str="P(F=={7}|S=={8})={9})"
    finalstr=p1str+p2str+p3str
    print(finalstr.format(index,f1,s1,prob1,
                                  f2,s2,prob2,
                                   f3,s3,prob3 ))

def Probability(f,s,A,flist,slist):
    f1count=flist.count(f)
    s1count=slist.count(s)
    f1s1count= countFS(f,s,A)
    op=str(f1s1count)+'/'+str(s1count)
    return op

def countFS(f,s,A):
    count=0
    for i in A:
        if i[0]==f and i[1]==s:
            count +=1

    return count
# you can free to change all these codes/structure
def compute_conditional_probabilites(A):
    f=[]
    s=[]
    #print(A)
    nr=len(A)
    for i in A:
        f.append(str(i[0]))
        s.append(str(i[1]))

    #print(f)
    #print(s)
    #logic: p(a/b)=p(a&b)/p(b)
    #p(F==F1|S==S1)
    printProbablity('a','F1','S1',Probability('F1','S1',A,f,s),
                        'F1','S2',Probability('F1','S2',A,f,s),
                        'F1','S3',Probability('F1','S3',A,f,s))

    printProbablity('b','F2','S1',Probability('F2','S1',A,f,s),
                        'F2','S2',Probability('F2','S2',A,f,s),
                        'F2','S3',Probability('F2','S3',A,f,s))

    printProbablity('c','F3','S1',Probability('F3','S1',A,f,s),
                        'F3','S2',Probability('F3','S2',A,f,s),
                        'F3','S3',Probability('F3','S3',A,f,s))

    printProbablity('d','F4','S1',Probability('F4','S1',A,f,s),
                        'F4','S2',Probability('F4','S2',A,f,s),
                        'F4','S3',Probability('F4','S3',A,f,s))
```

```
    printProbablity('e','F5','S1',Probability('F5','S1',A,f,s),
                          'F5','S2',Probability('F5','S2',A,f,s),
                          'F5','S3',Probability('F5','S3',A,f,s))




    #p(F==F1|S==S2)


    # your code
    # print the output as per the instructions
    return

def readInput():
    rows=[]
    nr=int(input("enter number of rows:"))
    for i in range(nr):
        rowVals=[vals.upper() for vals in input().split() ]
        if len(rowVals)!=2:
            print("Error: input only two elements per row")
            quit()
        rows.append(rowVals)
    return rows


#A = [['F1','S1'],['F2','S2'],['F3','S3'],['F1','S2'],['F2','S3'],['F3','S2'],
['F2','S1'],['F4','S1'],['F4','S3'],['F5','S1']]
A= readInput()
compute_conditional_probabilites(A)
```

```
enter number of rows:10
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: 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 [2]:
```
def string_features(s1,s2):
    s1list=  set(s1.split(' '))
    s2list=  set(s2.split(' '))


    return len(s1list & s2list), list(s1list-s2list), list(s2list-s1list)




s1=input("enter string1:")
s2=input("enter string2:")

a,b,c = string_features(s1,s2)

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

```
enter string1:the first column F will contain only 5 unique values
enter string2:the second column S will contain only 3 unique values
7
['first', '5', 'F']
['3', 'S', '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
```

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

In [1]:
```
import math
# 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 inp
ut strings


# you can free to change all these codes/structure
def compute_log_loss(A):
    # your code
    sum = 0.0
    for row in A:
        try:
          sum += (abs(row[0])*math.log10(row[1]))+(abs(1-row[0])*math.log10(ab
s(1-row[1])))
        except:
            pass

    loss = (-1*sum)/len(A)
    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, 0.8]]
nrows=int(input("enter number of rows:"))
A=[]
for row in range(nrows):
    rowvals=[float(vals) for vals in input().split()]
    if(len(rowvals)) !=2:
        print("only 2 values allowed in a row")
        quit()
    A.append(rowvals)


print(A)
loss = compute_log_loss(A)
print(loss)
```

```
enter number of rows:8
1 0.4
0 0.5
0 0.9
0 0.3
0 0.6
1 0.1
1 0.8
1 0.9
[[1.0, 0.4], [0.0, 0.5], [0.0, 0.9], [0.0, 0.3], [0.0, 0.6], [1.0, 0.1], [1.
0, 0.8], [1.0, 0.9]]
0.42430993457031635
```