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 [69]:

```python
def matrix_mul(A, B):
    """This function return the matrix multiplication of two given matrix.
       It takes input from the user to create a matrix."""

    #crate a zero matrix first with name Z
    Z = [[0 for j in range(len(B[0]))] for i in range(len(A))]

    #Multiplication of two matrices

    for i in range(len(A)):              #iterate over rows of A
        for j in range(len(B[0])):       #iterate over columns of B
            for k in range(len(B)):      #iterate over each element in rows of B
                Z[i][j] += A[i][k] * B[k][j]
    return Z



#Create matrix A
print("Matrix A :")
r1 = int(input("Rows of matrix A:"))
c1 = int(input("Column of matrix A:"))

A = []
for i in range(r1):
    rows = []
    for j in range(c1):
        rows.append(int(input()))

    A.append(rows)
print(A)

#Create matrix B
print("Matrix B :")
r2 = int(input("Rows of matrix B:"))
c2 = int(input("Column of matrix B:"))

B = []
for i in range(r2):
    rows = []
    for j in range(c2):
        rows.append(int(input()))

    B.append(rows)
print(B)
#Check the condition for matrix multiplication i.e column of matrix 1 = rows of matrix 2
if c1 != r2:
    print("A*B = Not Possible")

else:
    Z = matrix_mul(A,B)
    print("A*B = ")
    for row in Z:
        print(row)
```

```
Matrix A :
Rows of matrix A:2
Column of matrix A:3
2
3
4
5
6
7
[[2, 3, 4], [5, 6, 7]]
Matrix B :
Rows of matrix B:3
Column of matrix B:3
7
6
5
8
9
4
5
6
7
[[7, 6, 5], [8, 9, 4], [5, 6, 7]]
A*B =
[58, 63, 50]
[118, 126, 98]
```

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 [12]:

```python
import random

def pick_a_number_from_list(A):
    sum = 0
    cum_sum = []
    for i in range(len(A)):
        sum+=A[i]
        cum_sum.append(sum)    #cumulative sum of the list
    #print(cum_sum)             #print cum_sum
    rand_num = int(random.uniform(0,sum))    #selecting a random number between 0 and sum
    print(rand_num)

    number = 0
    for index in range(len(cum_sum)):
        #Accumulate the values starting from the beginning of the array until you are >= to the random value.
        if (rand_num >= cum_sum[index] and rand_num < cum_sum[index+1]):
            return A[index+1]
    return number
def sampling_based_on_magnitude():
    A = [0, 5, 27, 6, 13, 28, 100, 45, 10, 79]
    x = dict()
    for i in range(1,100):
        number = pick_a_number_from_list(A)
        #print(number)
        if number not in x:
            x[number] = 1
        else:
            x[number]+=1
    print(x)


sampling_based_on_magnitude()
```

```
169
283
113
88
12
191
196
190
104
72
293
105
219
245
207
0
17
88
26
293
165
12
111
90
166
112
62
191
155
102
18
105
287
150
44
273
218
37
278
279
287
281
143
224
295
282
281
120
190
21
74
91
43
74
306
311
47
215
96
234
117
200
184
108
84
238
268
99
261
44
155
97
242
92
127
15
178
83
97
101
166
106
189
32
280
227
291
97
112
179
243
52
242
195
4
286
48
224
164
{100: 36, 79: 25, 27: 7, 45: 14, 28: 5, 5: 2, 13: 5, 6: 2, 10: 3}
```

In [ ]:

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 [73]:

```python
import re

def replace_digits(string):
    #Find all the digits in the string and add them to a list.
    num_str = "".join(re.findall('\d',string))

    #Replace the digits with '#'
    final_str = re.sub('\d','#',num_str)
    return(final_str) # modified string which is after replacing the # with digits

string = input()
replace_digits(string)
```

```
#2a$#b%c%561#
```

Out[73]: `'####'`

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','student7','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 [50]:

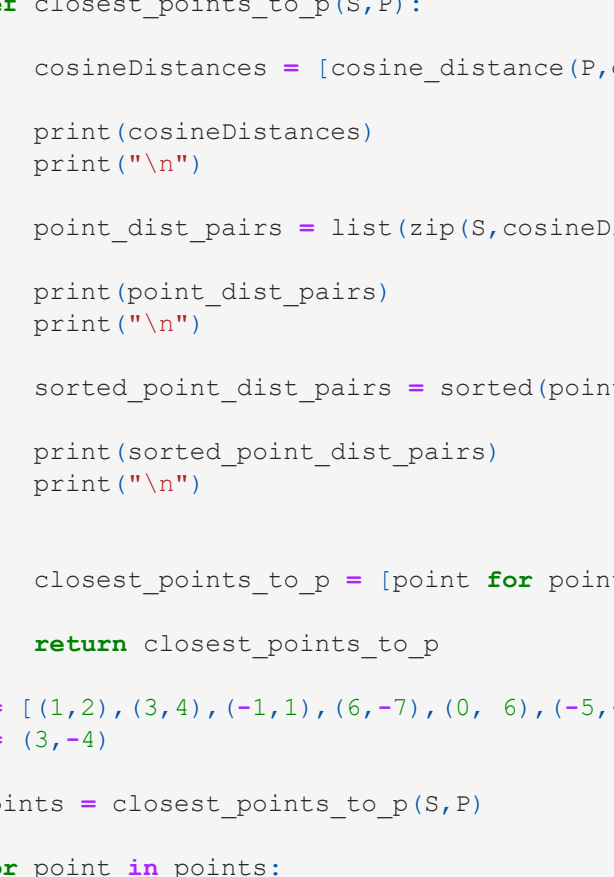
```python
def display_dash_board(students, marks):

    dic = dict(zip(students,marks))
    print('Students who got top 5 ranks')
    first_five = (sorted(dic.values())[5:])[::-1]

    for m in first_five:
        print('Name :',list(dic.keys())[list(dic.values()).index(m)])
        print('Marks :',m)
    print('\n')

    print('Students who got least 5 ranks')
    last_five = (sorted(dic.values())[:5])

    for n in last_five:
        print('Name :',list(dic.keys())[list(dic.values()).index(n)])
        print('Marks :',n)

    print('\n')

    print('Students who got marks between IQR')
    for x in sorted(dic.values()):
        if (x>=25 and x<=75):
            print('Name:',list(dic.keys())[list(dic.values()).index(x)])
            print('Marks :',x)
        else:
            pass

Students=['student1','student2','student3','student4','student5','student6','student7','student8','student9','s
Marks = [45, 78, 12, 14, 48, 43, 47, 98, 35, 80]

display_dash_board(Students,Marks)
```

```
Students who got top 5 ranks
Name : student8
Marks : 98
Name : student10
Marks : 80
Name : student2
Marks : 78
Name : student5
Marks : 48
Name : student7
Marks : 47


Students who got least 5 ranks
Name : student3
Marks : 12
Name : student4
Marks : 14
Name : student9
Marks : 35
Name : student6
Marks : 43
Name : student1
Marks : 45


Students who got marks between IQR
Name: student9
Marks : 35
Name: student6
Marks : 43
Name: student1
Marks : 45
Name: student7
Marks : 47
Name: student5
Marks : 48
```

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 [51]:

```python
import math

def cosine_distance(p,q):

    numerator = (p[0]*q[0]) + (p[1]*q[1])
    denominator = math.sqrt(p[0]**2+p[1]**2) + math.sqrt(q[0]**2+q[1]**2)

    return numerator/denominator


def closest_points_to_p(S,P):

    cosineDistances = [cosine_distance(P,q) for q in S]

    print(cosineDistances)
    print("\n")

    point_dist_pairs = list(zip(S,cosineDistances))

    print(point_dist_pairs)
    print("\n")

    sorted_point_dist_pairs = sorted(point_dist_pairs,reverse=True,key=lambda x:x[1])

    print(sorted_point_dist_pairs)
    print("\n")


    closest_points_to_p = [point for point,_ in sorted_point_dist_pairs[:5]]

    return closest_points_to_p

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

points = closest_points_to_p(S,P)

for point in points:
    print(point)
```

```
[-0.6909830056250525, -0.7, -1.0913263071038406, 3.234984083924547, -2.1818181818181817, 1.1777762382025354, 0.
15590375815769153, 1.6363636363636365, 1.0913263071038406]


[((1, 2), -0.6909830056250525), ((3, 4), -0.7), ((-1, 1), -1.0913263071038406), ((6, -7), 3.234984083924547),
((0, 6), -2.1818181818181817), ((-5, -8), 1.1777762382025354), ((-1, -1), 0.15590375815769153), ((6, 0), 1.6363
636363636365), ((1, -1), 1.0913263071038406)]


[((6, -7), 3.234984083924547), ((6, 0), 1.6363636363636365), ((-5, -8), 1.1777762382025354), ((1, -1), 1.091326
3071038406), ((-1, -1), 0.15590375815769153), ((1, 2), -0.6909830056250525), ((3, 4), -0.7), ((-1, 1), -1.09132
63071038406), ((0, 6), -2.1818181818181817)]


(6, -7)
(6, 0)
(-5, -8)
(1, -1)
(-1, -1)
```

In [26]:

```python
import math

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

def close_points(point,point_list):
    for i in range(len(point_list)):
        d = math.acos(((point_list[i][0]*point[0])+(point_list[i][1]*point[1]))
                      /((math.sqrt((point_list[i][0]**2)+(point_list[i][1]**2))) *(math.sqrt((point[0]**2)+(poi
        distance[point_list[i]] = d
    print(distance)

    dis = sorted(distance.values())[:5]
    print("\n")
    print(dis)

    print("Closes point from P :")
    for m in dis:
            print("Distance from P(3,4) point to the {}".format(list(distance.keys())[list(distance.values()).i
#           print(m)


close_points(P, S)
```

```
{(1, 2): 2.0344439357957027, (3, 4): 1.8545904360032246, (-1, 1): 2.9996955989856287, (6, -7): 0.06512516333438
509, (0, 6): 2.498091544796509, (-5, -8): 1.2021004241368467, (-1, -1): 1.4288992721907328, (6, 0): 0.927295218
0016123, (1, -1): 0.14189705460416438}


[0.06512516333438509, 0.14189705460416438, 0.9272952180016123, 1.2021004241368467, 1.4288992721907328]
Closes point from P :
Distance from P(3,4) point to the (6, -7)
Distance from P(3,4) point to the (1, -1)
Distance from P(3,4) point to the (6, 0)
Distance from P(3,4) point to the (-5, -8)
Distance from P(3,4) point to the (-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

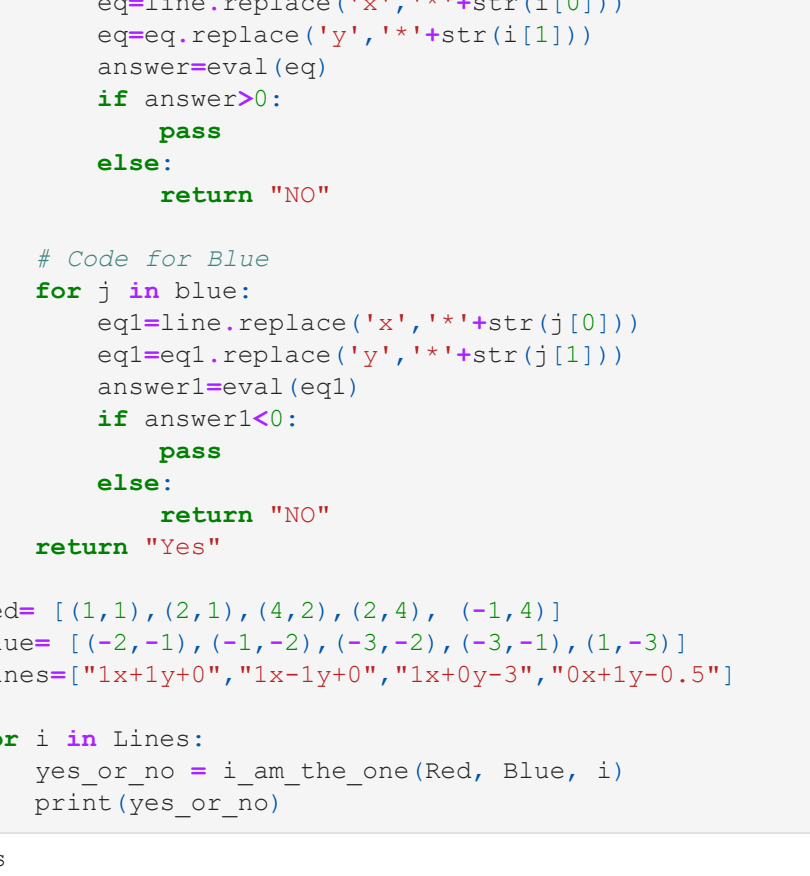
```
Red =[(R11,R12),(R21,R22),(R31,R32),(R41,R42),(R51,R52),..,(Rn1,Rn2)]
Blue=[(B11,B12),(B21,B22),(B31,B32),(B41,B42),(B51,B52),..,(Bm1,Bm2)]
```

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 [27]:

```python
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 input strings
# https://stackoverflow.com/questions/57188227/to-find-whether-a-given-line-equation-is-able-to-separate-the-t


# you can free to change all these codes/structure
def i_am_the_one(red,blue,line):
    # your code
    for i in red:
        eq=line.replace('x','*'+str(i[0]))
        eq=eq.replace('y','*'+str(i[1]))
        answer=eval(eq)
        if answer>0:
            pass
        else:
            return "NO"

    # Code for Blue
    for j in blue:
        eq1=line.replace('x','*'+str(j[0]))
        eq1=eq1.replace('y','*'+str(j[1]))
        answer1=eval(eq1)
        if answer1<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+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 [15]:

```python
# 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 curve_smoothing(string):
    index_of_non_empty_cells_list = []

    split_string = string.split(',')
    for i in range(len(split_string)):
        if split_string[i] != '_':
            index_of_non_empty_cells_list.append(i)

    # add the length of the original list to index_of_non_empty_cells_list
    index_of_non_empty_cells_list.append(len(split_string) - 1)

    #print(index_of_non_empty_cells_list)

    # Loop over the index_of_non_empty_cells_list to modify the original list by filling up empty slots

    start = 0

    for ele in index_of_non_empty_cells_list:
        # Whenever there is a missing value, add the proceeding value and the immediately next available value
        # and divide it by the number of missing values.
        # Calculate the cumulative_sum of available non-empty cells.
        cumulative_sum_prev_and_next_value = int(split_string[ele]) if split_string[ele] != '_' else 0
        cumulative_sum_prev_and_next_value += int(split_string[start]) if split_string[start] != '_' and start
        #print(cumulative_sum_prev_and_next_value)

        # Divide the cumulative_sum of prev and next value by the number of missing values
        integer_to_replace = cumulative_sum_prev_and_next_value // (ele - start + 1)
        split_string = [integer_to_replace if start <= x <= ele else split_string[x] for x in range(len(split_s
        start = ele

    return split_string#list of values

# S =  "_,_,30,_,_,_,50,_,_"
# smoothed_values= curve_smoothing(S)
# print(smoothed_values)

# Test cases
tests = [
    "_,_,_,24",
    "40,_,_,_,60",
    "80,_,_,_,_",
     "_,_,30,_,_,_,50,_,_"]

for i in tests:
    print (curve_smoothing(i))
```

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

In [12]:

```python
string = "_,_,30,_,_,_,50,_,_"
split_string = string.split(',')
print(split_string)
print(len(split_string))
```

```
['_', '_', '30', '_', '_', '_', '50', '_', '_']
9
```

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 [2]:

```python
# 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 compute_conditional_probabilites(F,S):
    # your code
    numerator = 0
    denominator = 0
    for i in range(len(A)):
        if(A[i][1] == S):
            denominator += 1
            if(A[i][0] == F):
                numerator += 1

    print('P(F = {} | S == {}) = {}/{}'.format(F,S, str(numerator), str(denominator)))

    # 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','S3'

for i in ['F1', 'F2', 'F3', 'F4', 'F5']:
    for j in ['S1', 'S2', 'S3']:
        compute_conditional_probabilites(i, j)
```

```
P(F = F1 | S == S1) = 1/4
P(F = F1 | S == S2) = 1/3
P(F = F1 | S == S3) = 0/3
P(F = F2 | S == S1) = 1/4
P(F = F2 | S == S2) = 1/3
P(F = F2 | S == S3) = 1/3
P(F = F3 | S == S1) = 0/4
P(F = F3 | S == S2) = 1/3
P(F = F3 | S == S3) = 1/3
P(F = F4 | S == S1) = 1/4
P(F = F4 | S == S2) = 0/3
P(F = F4 | S == S3) = 1/3
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 [44]:

```python
def string_features(S1, S2):
    """This function return :
       1.Number of common words between two provided Strings.
       2.Words in String1 but not in String2.
       3.Words in String2 but not in String1."""

    #convert the string into a list
    lst1 = S1.split()
    print(lst1)
    lst2 = S2.split()
    print(lst2)
    print("\n")
    #Number of common words btw both strings using forloop
    common_ele = 0
    for ele in lst1:
        if ele in lst2:
            common_ele+=1
    print("Number of common words between S1, S2: ",common_ele)

    #Words in S1 but not in S2
    a = []
    for ele in lst1:
        if ele in lst2:
            a = list(set(lst1) - set(lst2))
    print("Words in S1 but not in S2 : ",a)

    b = []
    for ele in lst2:
        if ele in lst1:
            b = list(set(lst2) - set(lst1))
    print("Words in S2 but not in S1 : ",b)

S1= "the first column F will  contain only 5 uniques values"
S2= "the second column S will  contain only 3  uniques values"
string_features(S1,S2)
```

```
['the', 'first', 'column', 'F', 'will', 'contain', 'only', '5', 'uniques', 'values']
['the', 'second', 'column', 'S', 'will', 'contain', 'only', '3', 'uniques', 'values']


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

In [38]:
```python
#anotherway
def string_features(S1,S2):

    #create a list of unique words in S1
    words_S1 = [item.lower() for item in set(S1.split())]
    print(words_S1)


    a = 0

    for word in words_S1:
        if word in S2:
            a+=1

    set_S1 = set(S1.split())
    set_S2 = set(S2.split())

    b = set_S1 - set_S2
    print(b)

    c = set_S2 - set_S1
    print(c)

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

```
['f', 'only', 'the', 'column', '5', 'contain', 'values', 'will', 'uniques', 'first']
{'F', '5', 'first'}
{'S', '3', 'second'}
7
{'F', '5', 'first'}
{'S', '3', 'second'}
```

IF WE CONSIDER THE BOUNDARY CASE FOR WORDS UPPER AND LOWER CASE

In [48]:
```python
#IF WE CONSIDER THE BOUNDARY CASE FOR WORDS UPPER AND LOWER CASE
def string_features(S1, S2):
    """This function return :
       1.Number of common words between two provided Strings.
       2.Words in String1 but not in String2.
       3.Words in String2 but not in String1."""

    #convert the string into a list
    lst1 = S1.lower().split()
    print(lst1)
    lst2 = S2.lower().split()
    print(lst2)
    print("\n")
    #Number of common words btw both strings using forloop
    common_ele = 0
    for ele in lst1:
        if ele in lst2:
            common_ele+=1
    print("Number of common words between S1, S2: ",common_ele)

    #Words in S1 but not in S2
    a = []
    for ele in lst1:
        if ele in lst2:
            a = list(set(lst1) - set(lst2))
    print("Words in S1 but not in S2 : ",a)

    b = []
    for ele in lst2:
        if ele in lst1:
            b = list(set(lst2) - set(lst1))
    print("Words in S2 but not in S1 : ",b)

S1= "The first column F will  contain only 5 uniques values"
S2= "the second Column S will  contain only 3  uniQues values"
string_features(S1,S2)
```

```
['the', 'first', 'column', 'f', 'will', 'contain', 'only', '5', 'uniques', 'values']
['the', 'second', 'column', 's', 'will', 'contain', 'only', '3', 'uniques', 'values']


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

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
```

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

In [34]:
```python
import math

def compute_log_loss(A):
    """This function return the loss of a given matrix A."""

    #Initializing value for log
    log_val = 0
    for item in A:

        y,y_score = item
        #Using formula given above calculating loss
        log_val += (y * math.log10(y_score)) + ((1-y) * math.log10(1-y_score))

    loss = (-1/len(A)) * log_val
    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]]
loss = compute_log_loss(A)
print("%.7f"%loss)
```

```
0.4243099
```

In [ ]: