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 [82]:

```
# DEFINE A FUNCTION TO MULTIPLY TWO MATRICES:

def matrix_mul(A, B):
    if (cols_A != rows_B):  # boundary case
        return "Matrix Multiplication is not possible"
    else:
        n = cols_A
        Multiplied_output = [[0 for i in range(cols_B)] for j in range(
rows_A)]

        for p in range(rows_A):
            for q in range(cols_B):
                for r in range(n):
                    Multiplied_output[p][q] += A[p][r] * B[r][q]

        print("The resultant matrix is: \n")
        for p1 in range(rows_A):
            for q1 in range(cols_B):
                print(Multiplied_output[p1][q1],end=" ")
            print("\n")


# INPUT DESIRED ORDER TO CREATE MATRICES:

rows_A = int(input("Enter the number of rows of matrix A: "))


cols_A = int(input("Enter the number of columns of matrix A: "))


rows_B = int(input("Enter the number of rows of matrix B: "))


cols_B = int(input("Enter the number of columns of matrix B: "))
```

```python
# CREATE MATRIX A:
print("\nCreate matrix A :")
A = [[int(input()) for i in range(cols_A)] for j in range(rows_A)]

print("\nThe created Matrix A is: \n")
for i in range(rows_A):
    for j in range(cols_A):
        print(A[i][j],end=" ")
    print("\n")

# CREATE MATRIX B:
print("\nCreate matrix B :")
B = [[int(input()) for i in range(cols_B)] for j in range(rows_B)]

print("\nThe created Matrix B is: \n")
for i in range(rows_B):
    for j in range(cols_B):
        print(B[i][j],end=" ")
    print("\n")

# TO MULTIPLY THE MATRICES call the function--> matrix_mul():
matrix_mul(A, B)
```

```
Enter the number of rows of matrix A: 2
Enter the number of columns of matrix A: 2
Enter the number of rows of matrix B: 2
Enter the number of columns of matrix B: 5

Create matrix A :
1
2
3
4

The created Matrix A is:

1 2

3 4
```

```
Create matrix B :
1
2
3
4
5
5
6
7
8
9

The created Matrix B is:

1 2 3 4 5

5 6 7 8 9

The resultant matrix is:

11 14 17 20 23

23 30 37 44 51
```

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 ex
periments.
```

```
f(100) > f(79) > f(45) > f(28) > f(27) > f(13) > f(10) > f(6) >
f(5) > f(0)
```

In [79]:
```
from random import uniform  # necessary package for the task

# define the function:
def pick_a_number_from_list(A):

    # SORT THE LIST AND NORMALIZE THE VALUES:
    A = sorted(A)
    normalized_A = [(A[i]/sum(A)) for i in range(len(A))]

    # CREATING CUMULATIVE SUM OF NORMALIZED LIST:
    cumulative_list = []
    element_list = 0
    for j in normalized_A:
        element_list += j
        cumulative_list.append(element_list)

    # GENERATING RANDOM NUMBER AND SELECTING AN ELEMENT FROM LIST:
    random_number = uniform(0,1)
    selected_random_number = 0
    for i in range(len(cumulative_list)-1):
        if (random_number >= cumulative_list[i]) & (random_number <= cu
mulative_list[i+1]):
            selected_random_number = A[i+1]
            break

    return selected_random_number

def sampling_based_on_magnitude():
    for i in range(1,50):
        number = pick_a_number_from_list(A)
        print(number,end=" ")

# TEST CASE: input (A = [0,5,27,6,13,28,100,45,10,79])
A = [0,5,27,6,13,28,100,45,10,79]
sampling_based_on_magnitude()
```

```
45 100 28 100 79 79 45 100 79 100 100 79 79 100 100 100 79 13 28 28 45
79 27 100 45 5 27 79 27 100 79 13 10 13 5 45 100 5 79 79 45 79 79 45 5
79 79 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 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 [76]:

```python
import re       # necessary package for the task

# define the function:
def replace_digits(String):

    # CREATE DIGIT PATTERN TO MATCH THE DIGITS IN THE GIVEN STRING:

    digit_pattern = re.compile(r"\d")
    digits_match = digit_pattern.finditer(string)

    # EXTRACT NUMBER OF DIGITS IN THE GIVEN STRING:

    noof_digits = 0
    for j in digits_match:
        noof_digits += 1
        print(j)

    # REPLACE THE NO OF DIGITS WITH "#" :

    print("No of digits in the given string :",noof_digits)
    output_string = "#"*(noof_digits)
    return output_string

# TEST CASE: input (string = "#2a$#b%c%561#" or "abc" )
string = "#2a$#b%c%561#"
replace_digits(string)
```

```
<re.Match object; span=(1, 2), match='2'>
<re.Match object; span=(9, 10), match='5'>
<re.Match object; span=(10, 11), match='6'>
<re.Match object; span=(11, 12), match='1'>
No of digits in the given string : 4
```

Out[76]: '####'

Q4: Students marks dashboard

consider the marks list of class students given two lists
Students =
['student1','student2','student3','student4','student5','student6','student7','student8','student9','studer
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','student
5','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 [4]:

```
# define the function:
def display_dash_board(students, marks):

    # Lets map the two lists using dictionary and zip()

    dict_create = dict(zip(students,marks))
    list_mapping = list(dict_create.items())

    # Top 5 students:

    students_marks_descending = []
    students_marks_descending = sorted(list_mapping,key = lambda value
: value[1],reverse = True)
    top_5_students = [students_marks_descending[i] for i in range(5)]


    # Least 5 students:

    students_marks_ascending = []
    students_marks_ascending = sorted(list_mapping,key = lambda value :
 value[1],reverse = False)
    least_5_students = [students_marks_ascending[i] for i in range(5)]

    # Students in interquartile range:

    list_mapping1 = sorted(list_mapping,key = lambda value : value[1],r
everse = False)

    import math

    max_mark = max(list_mapping1,key = lambda value: value[1])
    min_mark = min(list_mapping1,key = lambda value: value[1])
    diff = max_mark[1] - min_mark[1]
    value_25th_percentile = diff*(0.25)
    value_75th_percentile = diff*(0.75)

    #print(value_25th_percentile)
    #print(value_75th_percentile)

    lst = []
    for i in range(len(list_mapping1)):
        if (list_mapping1[i][1] > value_25th_percentile) and (list_mapp
ing1[i][1] < value_75th_percentile):
            lst.append(list_mapping1[i])

    students_within_25_and_75 = lst

    # Return the results of our task:

    return top_5_students, least_5_students, students_within_25_and_75



students = ['student1','student2','student3','student4','student5','stu
dent6','student7','student8','student9','student10']
marks = [45, 78, 12, 14, 48, 43, 47, 98, 35, 80]
```

```python
top_5_students,least_5_students, students_within_25_and_75 = display_da
sh_board(students, marks)

# Our task is to print the following:

print("a.")
for i in top_5_students:
    print(*i,end = "\n\n")

print("b.")
for j in least_5_students:
    print(*j,end = "\n\n")


print("c.")
for j in students_within_25_and_75:
    print(*j,end = "\n\n")
```

```
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
```

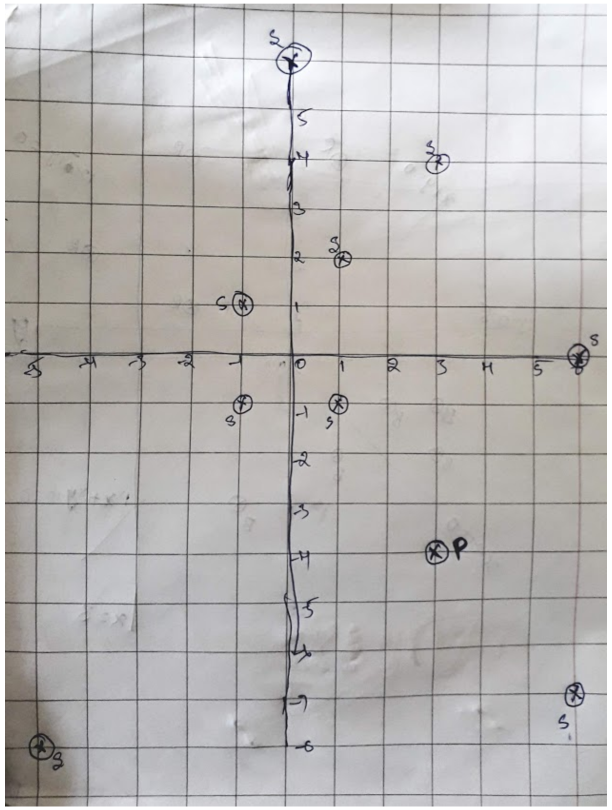
Q5: Find the closest points

consider you 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 [71]:
```
import math # necessary package for our task

# define the function:
def closest_points_to_p(S, P):
    # convert list of tuples to list of lists:

    S_list = [list(member) for member in S]
    P_list = list(P)

    # compute the cosine distance between the respective point and othe
r points in the list:

    distance_list = []
    for i in range(len(S_list)):
        x = S_list[i][0]
        y = S_list[i][1]
        p = P_list[0]
        q = P_list[1]
        n = (x*p) + (y*q)
        d = (math.sqrt(x*x + y*y)) * (math.sqrt(p*p + q*q))
        distance_list.append(math.acos(n/d))

    # Map the list of points and the distance values in a dictionary an
d convert to a list:

    dict_fromlists = dict(zip(S,distance_list))
    mapped_list = list(dict_fromlists.items())

    # Sort the distance according to the closest and farthest point:

    points_ascending = []
    points_ascending = sorted(mapped_list,key = lambda value : value[1
],reverse = False)

    # Return the 5 closest points from the list:

    closest_points = []
    for i in range(5):
        closest_points.append(points_ascending[i][0])

    return closest_points

# Test cases
```

```
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)

print("The 5 closest points are: ")
for i in points:
    print(i,end="\n")
```

```
The 5 closest points are:
(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)]
```

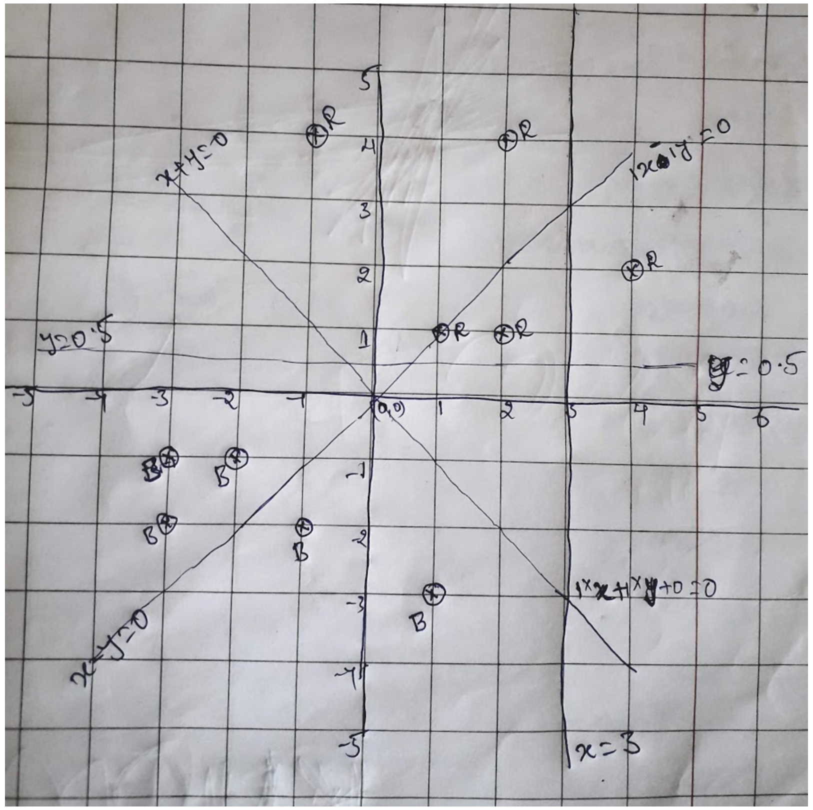
and set of line equations(in the string formate, i.e list of strings)

```
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 o
f 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 [80]:
```python
import math # necessary package for our task

# define the function:
def i_am_the_one(Red,Blue,Lines):
    # define an empty list and modify the given list elements for our d
esired task:
    final_lines = []
    for lin in Lines:
        list1 = list(lin)
        index1 = list1.index("x")
        index2 = list1.index("y")
        list1.insert(index1,"*")
        list1.insert(index2+1,"*")
        final_lines.append("".join([str(char) for char in list1]))

    print("The line equations are:",final_lines)
    tred = []
    tblue = []

    # evaluate the lines using the given points:

    for line1 in final_lines:
        for i in Red:
            x,y = (*i,)
            tred.append(eval(line1))

    for line2 in final_lines:
        for i in Blue:
            x,y = (*i,)
            tblue.append(eval(line2))

    tred1 = [tred[k1:k1 + len(Red)] for k1 in range(0,len(tred),len(Red
))]
    tblue1 = [tblue[k2:k2 + len(Blue)] for k2 in range(0,len(tblue),len
(Blue))]

    # check whether the red points and blue points are one side of the
 line :
    print("The required output is: ")
    j = 0
    while(j<len(tred1)):
        if all(j1>0 for j1 in tred1[j]) and all(l1<0 for l1 in tblue1[j
]):
```

```
            print("YES")
        else:
            print("NO")
        j = j+1

# Test set:
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"]

# call the function to check whether each line seperates red and blue p
oints
i_am_the_one(Red,Blue,Lines)
```

```
The line equations are: ['1*x+1*y+0', '1*x-1*y+0', '1*x+0*y-3', '0*x+1*
y-0.5']
The required output is:
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 distri
buted 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 di
stributed qually to all 5 places

Ex 3: 80, _, _, _, _  ==> 80/5,80/5,80/5,80/5,80/5 ==> 16, 16, 1
6, 16, 16 i.e. the 80 is distributed qually to all 5 missing val
ues 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 value
s (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 (1
0, 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 [9]:
```python
# reference: https://stackoverflow.com/questions/57179618/filling-the-m
issing-values-in-the-specified-format-python

# define the function:
def myfunc(S):
    # convert the given string into list
    print(S)
    k = S.split(",")
    #print(k)

    # According to the following conditions missing values are filled:

    if (k[-1].isdigit()) and all(k[i]=="_" for i in range(0,len(k)-1)):
        for j in range(len(k)):
            if j==(len(k)-1):
                print(int(int(k[-1])//len(k)),end="")
            else:
                print(int(int(k[-1])//len(k)),end=",")
        print("\n")



    elif (k[0].isdigit() and k[-1].isdigit()) and all(k[i]=="_" for i i
n range(1,len(k)-1)):
        for l in range(len(k)):
            if l==(len(k)-1):
                print(int((int(k[0])+int(k[-1]))//len(k)),end="")
            else:
                print(int((int(k[0])+int(k[-1]))//len(k)),end=",")
        print("\n")


    elif (k[0].isdigit()) and all(k[i]=="_" for i in range(1,len(k))):
        for i in range(len(k)):
            if i==(len(k)-1):
                print(int(int(k[0])//len(k)),end="")
            else:
                print(int(int(k[0])//len(k)),end=",")
        print("\n")
    else:
        curvalue = 0;prev_value = 0;prev_pos = 0;p = 0
        while p < len(k) :
            if k[p] != "_" or (p+1 == len(k)):
                if k[p] != "_":
                    curvalue = int(k[p])
                else:
                    curvalue = 0
                Length = (p-prev_pos+1)
                filled_value = (curvalue+prev_value)//(Length)
                for j in range(prev_pos,p+1):
                    k[j] = filled_value
                prev_value = filled_value
                prev_pos = p
            p+=1
        for i in range(len(k)):
            if i == (len(k)-1):
                print(k[i],end="")
            else:
                print(k[i],end=",")

# test case:
test_list = ["_,_,_,24","40,_,_,_,60","80,_,_,_,_","_,_,30,_,_,_,50,_,
_"]

print("The required output for the test inputs are: \n")

for y in test_list:
    myfunc(y)       # call the function
```

```
The required output for the test inputs are:

_,_,_,24
6,6,6,6

40,_,_,_,60
20,20,20,20,20

80,_,_,_,_
16,16,16,16,16

_,_,30,_,_,_,50,_,_
10,10,12,12,12,12,4,4,4
```

Q8: Filling the missing values in the specified formate

You will be given a list of lists, each sublist will be of length 2 i.e. [[x,y],[p,q],[l,m]..[r,s]] consider its like a martrix of n rows and two columns

1. the first column F will contain only 5 uniques values (F1, F2, F3, F4, F5)
2. the second column S will contain only 3 uniques values (S1, S2, S3)

```
    your task is to find
    a. Probability of P(F=F1|S==S1), P(F=F1|S==S2), P(F=F1|S==S3)
    b. Probability of P(F=F2|S==S1), P(F=F2|S==S2), P(F=F2|S==S3)
    c. Probability of P(F=F3|S==S1), P(F=F3|S==S2), P(F=F3|S==S3)
    d. Probability of P(F=F4|S==S1), P(F=F4|S==S2), P(F=F4|S==S3)
    e. Probability of P(F=F5|S==S1), P(F=F5|S==S2), P(F=F5|S==S3)
```

Ex:

```
[[F1,S1],[F2,S2],[F3,S3],[F1,S2],[F2,S3],[F3,S2],[F2,S1],[F4,S
1],[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 [28]:
```
# reference:https://stackoverflow.com/questions/57160252/find-conditional-probabilities-using-python
# define the function:
def compute_conditional_probabilites(A):
    # define the unique values of each column of the list:
    F = ("F1","F2","F3","F4","F5")
    S = ("S1","S2","S3")
    for k in F:
        for j in S:
            n = 0
            m = 0
            # count the number of outcomes for the given condition:
            for i in range(len(A)):
                if A[i][1] == j:
                    m+=1
                    if A[i][0] == k:
                        n+=1
            # print the conditional probability values:
            print("P(F={}|S=={})=".format(k,j),"{}/{}".format(n,m),end=
" ,")
        print("\n")

# test case
A = [['F1','S1'],['F2','S2'],['F3','S3'],['F1','S2'],['F2','S3'],['F3',
'S2'],['F2','S1'],['F4','S1'],['F4','S3'],['F5','S1']]

compute_conditional_probabilites(A) # call the function
```

```
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 [42]:

```
# define the function:
def string_features(S1, S2):
    # used set() for my task
    set1 = set(S1.split(" "))
    set2 = set(S2.split(" "))

    #Number of common words between S1, S2
    a = len(set1.intersection(set2))

    #Words in S1 but not in S2
    b = set1.difference(set2)

    #Words in S2 but not in S1
    c = set2.difference(set1)

    return a, b, c

# test cases:
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) # call the function

# print the desired output:
print("a.",a)
print("b.",list(b))
print("c.",list(c))
```

```
a. 7
b. ['5', 'F', 'first']
```

```
c. ['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
$f(Y, Y_{score}) = -1 * \frac{1}{n}\Sigma_{foreachY,Y_{score}pair}(Ylog10(Y_{score}) + (1-Y)log10(1-Y_{sco}$
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 [68]:
```
import math # necessary package for computing log value

# define the function:
def compute_log_loss(A):
    sum = 0

    # use the logloss formula(iteratively) to obtain the final result:
    for i in range(len(A)):
        sum = sum + (((A[i][0])*(math.log(A[i][1],10))) + ((1-A[i][0])*
(math.log(1-A[i][1],10))))
```

```
    loss = -1*(sum/len(A))

    return round(loss,7) # final value is returned


# test case:
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) # call the function

print(loss) # print the required output
```

```
0.4243099
```