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 [5]:

```
# 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 exampl
es


# you can free to change all these codes/structure
# here A and B are list of lists
def matrix_mul(A, B):
    """
    This function return list of matrix multipication
    """
    if (int(len(A[0]))!=int(len(B))): #Validating 1st Matrix column and 2nd Matrix Row
        print("Not possible")
    else :
        result = [[0 for j in range(len(B[0]))] for i in range(len(A))] #initalizing re
sult is '0'
        for i in range(len(A)):  # Iterating rows of A
            for j in range(len(B[0])): #Iterating columns of B
                for k in range(len(B)): #Iterating rows of B
                    result[i][j] +=A[i][k]*B[k][j]
        return result


#Input 1
A = [[1,3,4],[2,5,7],[5,9,6]]
B=  [[1,0,0],[0,1,0],[0,0,1]]
#Input 2
#A  = [[1,2],[3,4]]
#B   = [[1,2,3,4,5],[5,6,7,8,9]]
#Input 3
#A   = [[1,2],[3,4]]
#B   = [[1,4],[5,6],[7,8],[9,6]]
p = matrix_mul(A, B)
for r in p:
        print(r)
```

```
[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 [6]:

```
    from random import uniform
    import random
    # 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 ex
amples


    # you can free to change all these codes/structure
    def pick_a_number_from_list(A):
        # your code here for picking an element from with the probability propotional t
o its magnitude
        """
        This function return Probablity Proporational Value
        """
        norms = []
        for i in range(len(A)):
            norms.append(A[i]/sum(A))  #Performed Normalization technique N values in r
ange of [0,1]
        cusum = []
        total = 0.0
        for x in range(len(norms)):
            total +=norms[x]
            cusum.append(total)   #Performed Cumulative Sum for calculated  normalized
 data
        #print(cusum)
        rand_uniform = random.uniform(0,1)  #Pick a uniform random variable in range of
[0,1]
        #print(rand_uniform)
        for i in range(len(cusum)):
            if rand_uniform < cusum[i]:  #Performing  proporational operation
                return A[i]


    def sampling_based_on_magnitued():
        A = [0,5,27,6,13,28,100,45,10,79]    #Given list of elements
        number = []
        for i in range(1,100):              #Loop iterating 100 times
            number.append(pick_a_number_from_list(A))
        differ = []
        for i in range(len(A)):
            differ.append(number.count(A[i]))  #Stored COUNT values  in list
        listmerged = {A[i]:differ[i] for i in range(0,len(A))}  #Merged values with the
ir count
        dictbased = {ele[0]:str(ele[1]) +" times" for ele in sorted(listmerged.items()
, reverse=True, key=lambda x: x[1])} #Performed dict operation in descending order
        print(dictbased)
        for x in dictbased.keys():
            print("f({}) >".format(x),end= " ")

    sampling_based_on_magnitued()
```

```
{100: '33 times', 79: '21 times', 45: '14 times', 13: '9 times', 27: '8 ti
mes', 28: '8 times', 10: '3 times', 5: '2 times', 6: '1 times', 0: '0 time
s'}
f(100) > f(79) > f(45) > f(13) > f(27) > f(28) > f(10) > f(5) > f(6) > f
(0) >
```

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 [7]:

```
import re
# write your python code here
# you can take the above example as sample input for your program to test
# it should work for any general input try not to hard code for only given input exampl
es

# you can free to change all these codes/structure
# String: it will be the input to your program
def replace_digits(S):
    alphabet = re.sub('\D',"",str(S))   #Replaced characters with white space
    number = re.sub("\d","#",str(alphabet)) #Replaced Digits with '#'
    return number # modified string which is after replacing the # with digits


lstrin = [234,'a2b3c4','abc','#2a$#b%c%561#']
#String = 234
#String = 'a2b3c4'
#String = 'abc'
#String = '#2a$#b%c%561#'
for i in lstrin:
    p = replace_digits(i)
    print("Output of {} is ".format(i), str(p))
```

```
Output of 234 is  ###
Output of a2b3c4 is  ###
Output of abc is
Output of #2a$#b%c%561# is  ####
```

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 [8]:

```
# 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 exampl
es
import math
def display_dash_board(students, marks):

    """
    This function return student details with respective ranks
    """
    listmerged = {students[i]:marks[i] for i in range(0,len(students))} #Merging two li
sts as dictinoary
    top_5_students = {ele[0]:ele[1] for ele in sorted(listmerged.items() , reverse=True
, key=lambda x: x[1])[:5]} #Sorted elements in decreasing order
    least_5_students ={ele[0]:ele[1] for ele in sorted(listmerged.items() , key=lambda
x: x[1])[:5]} #Sorted elements in increasing order
    s = len(listmerged.values()) #calculate no of elements in dictionary
    increaseorder =  {ele[0]:ele[1] for ele in sorted(listmerged.items() , key=lambda x
: x[1])} #Arranging all elements in increasing order
    xo = [values for i,values in enumerate(increaseorder.values()) if i == ((math.ceil(
s*0.25))-1)] #Calculated 25 th percentile
    yo = [values for i,values in enumerate(increaseorder.values()) if i == ((math.ceil(
s*0.75))-1)] #Calculated 75th percentile

    students_within_25_and_75 = {ele[0]:ele[1] for ele in sorted(listmerged.items() , k
ey=lambda x: x[1]) if ele[1]>=xo[0] and ele[1]<yo[0]} #Print that elements between 25th
and 75 th percentile

    return top_5_students, least_5_students, students_within_25_and_75

students = ['student1','student2','student3','student4','student5','student6','student
7','student8','student9','student10']
marks = [45, 78, 12, 14, 48, 43, 47, 98, 35, 80]
top_5_students, least_5_students, students_within_25_and_75 = display_dash_board(studen
ts, marks)
print("top_5_students :")
print(top_5_students)
print("least_5_students :")
print(least_5_students)
print("students_within_25_and_75 :")
print(students_within_25_and_75)
```

```
top_5_students :
{'student8': 98, 'student10': 80, 'student2': 78, 'student5': 48, 'student
7': 47}
least_5_students :
{'student3': 12, 'student4': 14, 'student9': 35, 'student6': 43, 'student
1': 45}
students_within_25_and_75 :
{'student9': 35, 'student6': 43, 'student1': 45, 'student7': 47, 'student
5': 48}
```

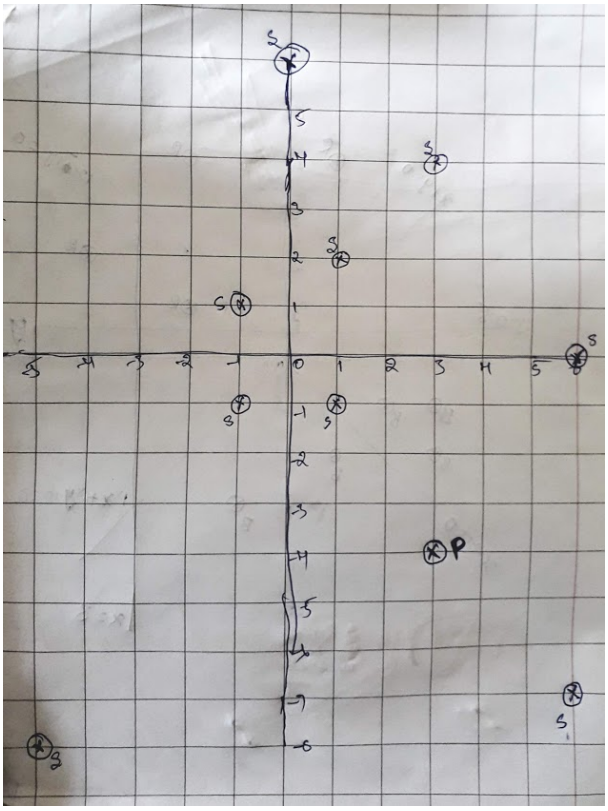
Q5: Find the closest points

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 [9]:

```
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 input exampl
es
# you can free to change all these codes/structure

import math
# here S is list of tuples and P is a tuple ot len=2
def closest_points_to_p(s, p):
    # write your code here
    output = ()
    z = math.sqrt(p[0]*p[0] + p[1]*p[1])  #Calculating sqrt(√p2+q2)
    #print(z)
    for i in range(len(s)):
        ss = math.sqrt(s[i][0]*s[i][0] + (s[i][1]*s[i][1]))  #Calculating sqrt(√x2+y2)
        sp = s[i][0]*p[0] + s[i][1]*p[1]  #Calculating (x·p+y·q)
        output+=((sp/(ss*z)),)   #Stored all values in tuple
    merged = {s[i]:output[i] for i in range(0,len(s))}   #Merged calculated values in i
nitial values
    closest_points_to_p = []
    for key,value in sorted(merged.items(), key=operator.itemgetter(1), reverse =True)
[:5]: #Finding 5 closest points to Point p
        closest_points_to_p.append(key)

    return closest_points_to_p  # its list of tuples

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(points) #print the returned values
```

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

Q6: Find which line separates oranges and apples

Consider you are given two set of data points in the form of list of tuples like

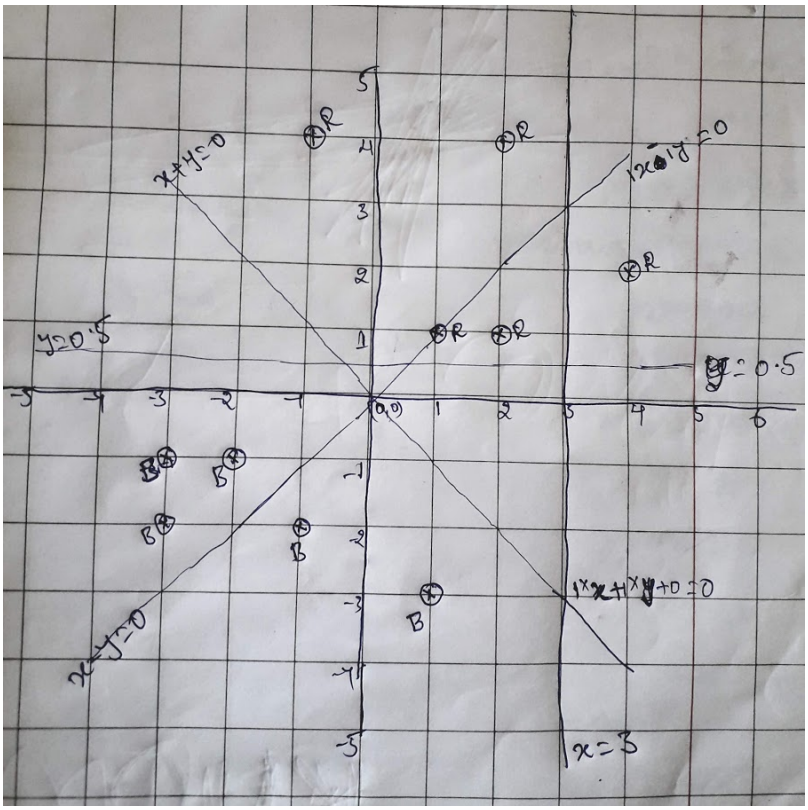
```
Red =[(R11,R12),(R21,R22),(R31,R32),(R41,R42),(R51,R52),..,(Rn1,Rn2)]
Blue=[(B11,B12),(B21,B22),(B31,B32),(B41,B42),(B51,B52),..,(Bm1,Bm2)]
```

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

```
Lines = [a1x+b1y+c1,a2x+b2y+c2,a3x+b3y+c3,a4x+b4y+c4,..,K lines]
Note: You need to do string parsing here and get the coefficients of x,y and 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 [10]:

```
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 string
s
import re

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

    a, b, c = [float(ele.strip()) for ele in re.split('x|y', line)]  #Spliting line int
o co-ordinates
    all_ = []
    new1 = []
    new2 = []
    all_.extend(red)  #Extending tuples into list
    all_.extend(blue)
    for i in range(len(all_)):
        di = a * all_[i][0] + b * all_[i][1] + c   #Passing all points through line
        if di > 0 :                                #If values is greater than 0,store i
t into new list
            new1.append(all_[i])
        else:
            new2.append(all_[i])                   #Else store it into new list
    if((sorted(new1)==sorted(red)) or (sorted(new1)==sorted(blue))):  #Comparing two li
sts equal or not
         return "YES"
    else :
         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"]

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

```
YES
NO
NO
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 [11]:

```
# 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 string
s


# you can free to change all these codes/structure
def curve_smoothing(string):
    """
    This function returns missing value in string
    """
    splited = string.split(',')

    inc = 0
    nextc = 0
    fp = 0
    lv = 0
    while inc < len(splited):
        if splited[inc] != '_' or (inc + 1 == len(splited)): #Check element is availabl
e or  if count is final value )
            if splited[inc] != '_':              #if element is available
                nextc = int(splited[inc])       #store  it to next value
            else:
                nextc = 0                        #Initiate the next value is zero
            latestvalue = (nextc + lv) / (inc - fp + 1)   #Find the new value
            for i in range(fp, inc + 1):              #Replace the "_" with new value
                splited[i] = latestvalue              #Changing existing string with ne
w values
            lv = latestvalue                          #Assigned to current value
            fp = inc                                  #Assigned to cuurent position
        inc += 1                                    #Loop iterates till all elements ar
e covered

    return splited

#S=  "_,_,30,_,_,_,50,_,_"
lint = ["_,_,_,24","40,_,_,_,60","80,_,_,_,_","_,_,30,_,_,_,50,_,_"]
for i in  lint:
    smoothed_values= curve_smoothing(i)
    print("Output of {} is ".format(i),smoothed_values)
```

```
Output of _,_,_,24 is  [6.0, 6.0, 6.0, 6.0]
Output of 40,_,_,_,60 is  [20.0, 20.0, 20.0, 20.0, 20.0]
Output of 80,_,_,_,_ is  [16.0, 16.0, 16.0, 16.0, 16.0]
Output of _,_,30,_,_,_,50,_,_ is  [10.0, 10.0, 12.0, 12.0, 12.0, 12.0, 4.
0, 4.0, 4.0]
```

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 [12]:

```
# 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 string
s



# you can free to change all these codes/structure
def compute_conditional_probabilites(A):
    for i in range(len(A)):
        m = A[i][0] + A[i][1]  #Performing merging with existing values in A
        test1[m] += 1      #Increment the dict1 value if exist in A
        test2[A[i][1]] += 1 #Increment the dict2 value if exist in A
    for key,value in test3.items():
        print('P(F='+ key.replace(value," ") +'| S=='+ value +') =' + str(test1[key])+
"/"+str(test2[value])) #Calucalating Probablity of all values in dict



A = [['F1', 'S1'], ['F2', 'S2'], ['F3', 'S3'], ['F1', 'S2'], ['F2', 'S3'], ['F3', 'S2'
], ['F2', 'S1'], ['F4', 'S1'], ['F4', 'S3'], ['F5', 'S1']]
res1, res2 = map(list, zip(*A))  #Splited List A into two lists
test1 = { str(i) + str(j) : 0 for i in set(res1) for j in set(res2)}   #Created dict wi
th key and value:0 with two lists
test3 = { str(i) + str(j) : str(j) for i in set(res1) for j in set(res2)}    #Created d
ict with two lists
test2 = { str(j) : 0 for j in set(res2) } #Created dict  with singe list
compute_conditional_probabilites(A)
```

```
P(F=F5 | S==S2) =0/3
P(F=F5 | S==S3) =0/3
P(F=F5 | S==S1) =1/4
P(F=F2 | S==S2) =1/3
P(F=F2 | S==S3) =1/3
P(F=F2 | S==S1) =1/4
P(F=F3 | S==S2) =1/3
P(F=F3 | S==S3) =1/3
P(F=F3 | S==S1) =0/4
P(F=F4 | S==S2) =0/3
P(F=F4 | S==S3) =1/3
P(F=F4 | S==S1) =1/4
P(F=F1 | S==S2) =1/3
P(F=F1 | S==S3) =0/3
P(F=F1 | S==S1) =1/4
```

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 [13]:

```
# 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 string
s

# you can free to change all these codes/structure
def string_features(S1, S2):
    """
    This function return count of common words or non common words in two sentences
    """
    ps = set(S1.split(" ")) #Split S1 into words (no duplicates are allowed)
    qs = set(S2.split(" ")) #Split S2 into words (no duplicates are allowed)
    p = len(list(ps&qs)) #Print Count common words both the sentences
    q = list(ps-qs) #Print only count words first sentence except Common words
    r = list(qs-ps) #Print only count words second sentence except Common words
    return p, q, r

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. " + str(a))
print("b. " + str(b))
print("c. " + str(c))
```

```
a. 7
b. ['5', 'F', 'first']
c. ['3', 'second', 'S']
```

Q10: Error Function

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

a. the first column Y will contain interger values
b. the second column Y_{score} will be having float values
Your task is to find the value of
$f(Y,Y_{score}) = -1 * \frac{1}{n}\Sigma_{foreachY,Y_{score}pair}(Ylog10(Y_{score}) + (1-Y)log10(1-Y_{score}))$ 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 [14]:

```
# 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 string
s

import math
# you can free to change all these codes/structure
def compute_log_loss(A):
    """
    This function return log loss value
    """
    sum =0
    for i in range(len(A)):
        p = (int(A[i][0])*math.log10(float(A[i][1])))+ ((1-int(A[i][0]))*math.log10(1-f
loat(A[i][1]))) #Calulating each iteration value of List A
        sum =  sum + p                  #Total sum of all iterationns of log loss
    lossaverage  = sum * -1 /(len(A))  #Calculating Log-loss value
    return lossaverage

A = [[1, 0.4], [0, 0.5], [0, 0.9], [0, 0.3], [0, 0.6], [1, 0.1], [1, 0.9], [1, 0.8]]
loss = compute_log_loss(A)
print(loss)
```

```
0.42430993457031635
```