# 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 [78]:

```python
# write your python code here
# you can take the above example as sample input for your program to te
st
# it should work for any general input try not to hard code for only gi
ven input examples

# Take two matrix A and B
A = [[1,2,3],[3,4,5],[3,4,5]]
B = [[1,2,3],[5,6,7],[7,8,9]]

# you can free to change all these codes/structure
# here A and B are list of lists
def matrix_mul(A,B):
    # To check number whether matrix multiplication is possible or not
    if(len(A[0])!=len(B)):
        return("Multiplication of matrix A and B is Not possible")
    else:
        # Define a Zero matrix of Size Row(A) * Column(B)
        lst = [[0 for i in range(len(B[0]))] for j in range(len(A))]

        # Iterate over Each row of matrix A
        for i in range(len(A)):

            # Iterate over Column of Matrix B
            for j in range(len(B[0])):
                for k in range(len(B)):
                    lst[i][j]+=A[i][k] * B[k][j]
        print("Multiplication of Matrix A and B is "+str(lst))

matrix_mul(A,B)
```

```
Multiplication of Matrix A and B is [[32, 38, 44], [58, 70, 82], [58, 7
0, 82]]
```

## 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 [34]:

```
import random
from random import uniform
from collections import Counter
import numpy as np
# write your python code here
# you can take the above example as sample input for your program to te
st
# it should work for any general input try not to hard code for only gi
ven input examples

A = [0,5,27,6,13,28,100,45,10,79]
# you can free to change all these codes/structure
def propotional_sampling(A):
    #create empty list for normalizae values
    lst=[]
    #create empty list for cum_sum
    lst1=[]
    # Sorting the list A
    A=sorted(A)

    # Sum of A
    Sum_of_A=sum(A)

    #calculating Normalize value
```

```
    for i in range(len(A)):
        lst.append(A[i]/Sum_of_A)

    #clculating Cumulative_Sum of each normalize value
    for i in lst:
        if len(lst1)==0:
            lst1.append(i)
        else:
            lst1.append(i+lst1[-1])

    #generating random variable
    r=random.uniform(0.0,1.0)

    #getting Propotional sampling with respect to that number
    for i in range(len(lst1)):
        if (r<=lst1[i]):
            number=A[i]
            break
    return number

def sampling_based_on_magnitued():
    lst2=[]
    for i in range(1,100):
        number = propotional_sampling(A)
        lst2.append(number)
    return Counter(lst2)

sampling_based_on_magnitued()
#propotional_sampling(A)
```

Out[34]:
```
Counter({10: 2, 28: 11, 100: 32, 79: 19, 45: 21, 13: 4, 5: 2, 27: 7, 6:
1})
```

### 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 [77]:
```
import re
# write your python code here
# you can take the above example as sample input for your program to te
st
# it should work for any general input try not to hard code for only gi
ven input examples
String= "#2a$#b%c%561#"

# you can free to change all these codes/structure
# String: it will be the input to your program
def replace_digits(String):
    # write your code
    # Convert a string to list
    li = list(String)
    # Create an empty string
    s = ""
    # Iterate over each item of list
    for ele in li:
        # To check whether a item is digit or not
        if ele.isdigit():
            # Append the '#' to the String
            s+=str('#')
    return(s) # modified string which is after replacing the # with dig
its

replace_digits(String)
```

Out[77]:
```
'####'
```

### 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','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 [76]:

```
# write your python code here
# you can take the above example as sample input for your program to te
st
# it should work for any general input try not to hard code for only gi
ven input examples
students = ['student1','student2','student3','student4','student5','stu
dent6','student7','student8','student9','student10']
marks = [45, 78, 12, 14, 48, 43, 45, 98, 35, 80]

def students_marks(Students,marks):

    # create a dictionary using these two lists
    d = dict(zip(students, marks))

# Sort dictionary in desc order to get Top 5 students
    Top_5_Students = [(k, d[k]) for k in sorted(d, key=d.get, reverse=T
rue)][:5]
    print('Top 5 Students:\n')
    for k, v in Top_5_Students:
        print(k,v)
    print('**********************************************************')

    # Sort dictionary in Asc order to get least 5 students
    Last_5_Students = [(k, d[k]) for k in sorted(d, key=d.get)][:5]
    print('last 5 Students:\n')
    for k, v in Last_5_Students:
        print(k,v)
    print('**********************************************************')

    # function to get 25th and 75th percentile to calculate IQR
```

```
    def in_percentile(marks, perc):
        myList=sorted(marks)
        l=len(myList)
        return (myList[int(l*perc)])

    percentile_25th = in_percentile(marks,0.25)

    percentile_75th = in_percentile(marks,0.75)

    percentile = [(k, d[k]) for k in sorted(d, key=d.get)]
    print('Students got marks between >25th percentile <75th percentil
e:\n')
    for k,v in percentile:
        if (v>=percentile_25th and v<percentile_75th):
            print(k,v)

students_marks(students,marks)
```

```
Top 5 Students:

student8 98
student10 80
student2 78
student5 48
student1 45
**********************************************************
last 5 Students:

student3 12
student4 14
student9 35
student6 43
student1 45
**********************************************************
Students got marks between >25th percentile <75th percentile:

student9 35
student6 43
student1 45
student7 45
```

```
student5 48
```

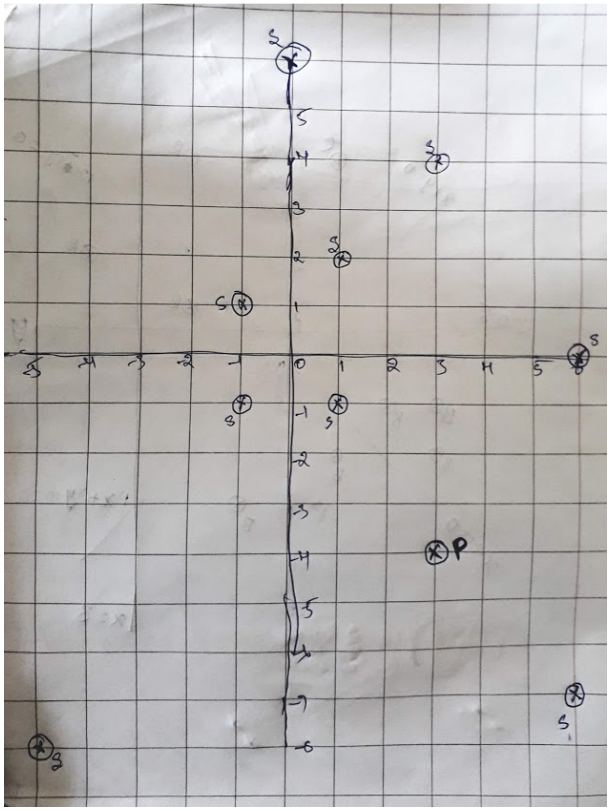
## Q5: Find the closest points

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

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 [75]:
```
import math

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

```
def shortest_distance(S,P):
    #create empty list
    c=[]
    index_S=[]
    top_5_closest_points=[]
    # calculate cosine distance
    for ele in S:
        sumxx = math.sqrt(ele[0]**2+ele[1]**2)
        sumyy= math.sqrt(P[0]**2+P[1]**2)
        sumxy=sumxx*sumyy
        d=math.acos((ele[0]*P[0]+ele[1]*P[1])/(sumxy))
        c.append(d)

    #calculate index of top_5 elements and append it to empty list
    index_S=sorted(range(len(c)), key=lambda i: c[i])[:5]

    # append value correpond to index into new list
    for i in index_S:
        top_5_closest_points.append(S[i])
    return ("Closest points from {0} is :" .format(P)+ str(top_5_closes
t_points))

shortest_distance(S,P)
```

Out[75]:
```
'Closest points from (3, -4) is :[(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

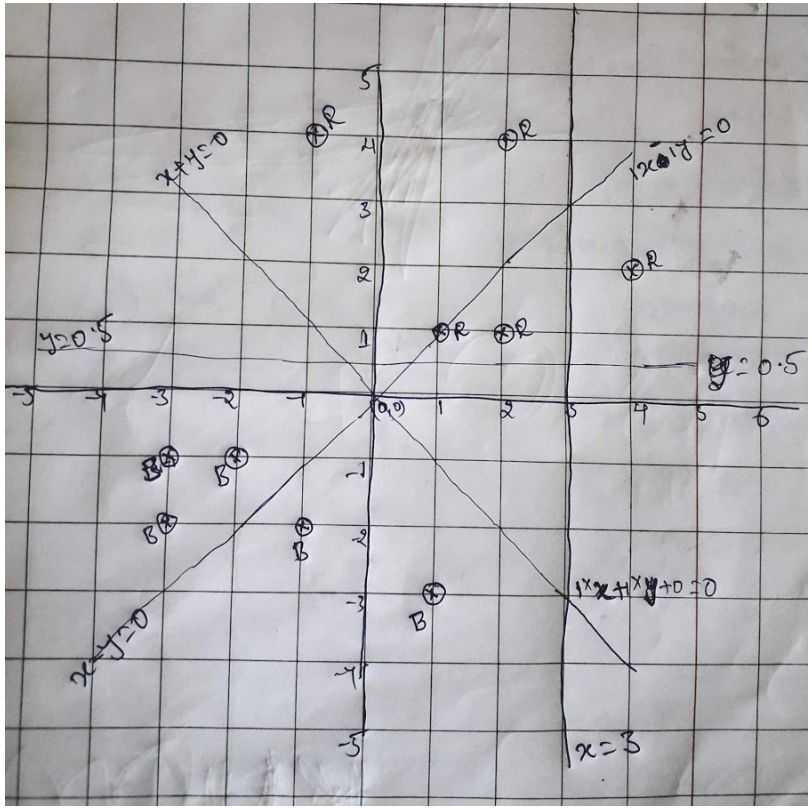
```
Red =[(R11,R12),(R21,R22),(R31,R32),(R41,R42),(R51,R52),..,(Rn1,
Rn2)]
Blue=[(B11,B12),(B21,B22),(B31,B32),(B41,B42),(B51,B52),..,(Bm1,
Bm2)]
```

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 coefficient
s of x,y and intercept.
```

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 [74]:

```python
import math
import re
# write your python code here
# you can take the above example as sample input for your program to te
st
# it should work for any general input try not to hard code for only gi
ven input strings


# you can free to change all these codes/structure
def i_am_the_one(Red,Blue,Lines):
    # your code
    #create empty lists
    Coeffiecient_remove=[]
    z=[]
    lst_Red=[]
    lst_Blue=[]

    for i in Lines:
        z = re.findall(r'[\d\.\-\+]+', i)
        Coeffiecient_remove.append(z)

    #covert lists of list to float value
    z=[[float(i) for i in val] for val in Coeffiecient_remove]

    #put all points in each line and getting sum
    for j in z:
        for i in Red:
            lst_Red.append(i[0]*j[0]+i[1]*j[1]+j[2])
        for i in Blue:
            lst_Blue.append(i[0]*j[0]+i[1]*j[1]+j[2])
        pos_count = len(list(filter(lambda x: (x >0), lst_Red)))
```

```
        neg_count = len(list(filter(lambda x: (x <0), lst_Blue)))

        #print yes if all red points and blue points on different side
 of a line
        if (neg_count==pos_count) and ((neg_count + pos_count) % 5==0):
            print("Yes")
        else:
            print("No")
        lst_Red=[]
        lst_Blue=[]


    #return #yes/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"]

i_am_the_one(Red,Blue,Lines)
```

```
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 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 [73]:
```
# write your python code here
# you can take the above example as sample input for your program to te
st
# it should work for any general input try not to hard code for only gi
ven input strings
```

```
def curve_smoothing(string):
    position =0
    next_item=0
    last_position=0
    last_item=0
    while position < len(string):
        if string[position] != '_' or (position + 1 == len(string)):
            if string[position] != '_':
                next_item = int(string[position])
            else:
                next_item = 0
            new_value = (next_item + last_item) / (position - last_posi
tion + 1)
            for i in range(last_position, position + 1):
                string[i] = new_value
            last_item = new_value
            last_position = position
        position += 1
    return string

S=  "_,_,30,_,8"
S=S.split(',')
curve_smoothing(S)

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

Out[73]: [10.0, 10.0, 6.0, 6.0, 6.0]

## 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,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 [72]:
```
# 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(P):
    # your code
    #getting sum of each list who have S1, S2, S3 respectively
```

```
    S1=sum('S1' in item for item in P)
    S2=sum('S2' in item for item in P)
    S3=sum('S3' in item for item in P)

    #Creating lists  od list who have second value is 'S1','S2','S3' re
spectively
    list1=[item for item in P if item[1]=='S1']

    list2=[item for item in P if item[1]=='S2']

    list3=[item for item in P if item[1]=='S3']

    P_F1_S1=P_F1_S2=P_F1_S3=P_F2_S1=P_F2_S2=P_F2_S3=P_F3_S1=P_F3_S2=P_F
3_S3=P_F4_S1=P_F4_S2=P_F4_S3=P_F5_S1=P_F5_S2=P_F5_S3=0

    #conditional probablilty of each item
    P_F1_S1=sum('F1' in item for item in list1 )/S1

    P_F1_S2=sum('F1' in item for item in list2 )/S2

    P_F1_S3=sum('F1' in item for item in list3 )/S3

    P_F2_S1=sum('F2' in item for item in list1 )/S1

    P_F2_S2=sum('F2' in item for item in list2 )/S2

    P_F2_S3=sum('F2' in item for item in list3 )/S3

    P_F3_S1=sum('F3' in item for item in list1 )/S1

    P_F3_S2=sum('F3' in item for item in list2 )/S2

    P_F3_S3=sum('F3' in item for item in list3 )/S3

    P_F4_S1=sum('F4' in item for item in list1 )/S1

    P_F4_S2=sum('F4' in item for item in list2 )/S2

    P_F4_S3=sum('F4' in item for item in list3 )/S3
```

```
    P_F5_S1=sum('F5' in item for item in list1 )/S1

    P_F5_S2=sum('F5' in item for item in list2 )/S2

    P_F5_S3=sum('F5' in item for item in list3 )/S3

    PF1=[]
    PF1.extend([P_F1_S1,P_F1_S2,P_F1_S3])
    PF2=[]
    PF2.extend([P_F2_S1,P_F2_S2,P_F2_S3])
    PF3=[]
    PF3.extend([P_F3_S1,P_F3_S2,P_F3_S3])
    PF4=[]
    PF4.extend([P_F4_S1,P_F4_S2,P_F4_S3])
    PF5=[]
    PF5.extend([P_F5_S1,P_F5_S2,P_F5_S3])

    print(' probability of P(F=F1|S==S1), P(F=F1|S==S2), P(F=F1|S==S3)
 is: {0} \n Probability of P(F=F2|S==S1), P(F=F2|S==S2), P(F=F2|S==S3)
 is: {1} \n Probability of P(F=F3|S==S1), P(F=F3|S==S2), P(F=F3|S==S3)
 is: {2} \n Probability of P(F=F4|S==S1), P(F=F4|S==S2), P(F=F4|S==S3)
 is: {3} \n Probability of P(F=F5|S==S1), P(F=F5|S==S2), P(F=F5|S==S3)
 is :{4}'.format(PF1,PF2,PF3,PF4,PF5))
    # 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'],['F5','S1']]
compute_conditional_probabilites(A)
```

```
 probability of P(F=F1|S==S1), P(F=F1|S==S2), P(F=F1|S==S3) is: [0.25,
0.3333333333333333, 0.0]
 Probability of P(F=F2|S==S1), P(F=F2|S==S2), P(F=F2|S==S3) is: [0.25,
0.3333333333333333, 0.3333333333333333]
 Probability of P(F=F3|S==S1), P(F=F3|S==S2), P(F=F3|S==S3) is: [0.0,
0.3333333333333333, 0.3333333333333333]
 Probability of P(F=F4|S==S1), P(F=F4|S==S2), P(F=F4|S==S3) is: [0.25,
0.0, 0.3333333333333333]
 Probability of P(F=F5|S==S1), P(F=F5|S==S2), P(F=F5|S==S3) is :[0.25,
0.0, 0.0]
```

## 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 [71]:

```
# write your python code here
# you can take the above example as sample input for your program to te
st
# it should work for any general input try not to hard code for only gi
ven input strings

# you can free to change all these codes/structure
def string_features(S1, S2):

    #split the string S1,S2 and covert it to list
    list1=S1.split(" ")

    list2=S2.split(" ")
    #create two empty lists
    a=[]

    b=[]
```

```
    #intialize the counter
    count=0
    # create two lists whose items are not present in one another
    a=[element for element in list1 if element not in list2]

    b=[element for element in list2 if element not in list1]

    #get total number of common items
    for i in list1:

        if i in list2:

            count+=1

    return a, b, count

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,b,c)
```

```
['first', 'F', '5'] ['second', 'S', '3'] 7
```

## 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
$f(Y, Y_{score}) = -1 * \frac{1}{n}\Sigma_{foreachY,Y_{score}pair}(Ylog10(Y_{score}) + (1-Y)log10(1-Y_{sco}$
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:
0.44982

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

In [70]:
```
# write your python code here
# you can take the above example as sample input for your program to te
st
# it should work for any general input try not to hard code for only gi
ven input strings
import math

# you can free to change all these codes/structure
def compute_log_loss(A):
    #creation of empty list
    s=[]
    # your code

    #iterating over loop to computh log function using math module
    for list in A:
        p=((list[0]*(math.log10(list[1])))+((1-list[0])*(math.log10(1-l
ist[1]))))
        s.append(p)

    #divide the sum of all item with total number of item to get final
 output
    loss=-sum(s)/len(s)
    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(loss)
```

0.42430993457031635