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 [45]:

```
# 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 examples


# you can free to change all these codes/structure
# here A and B are list of lists
def matrix_mul(A, B):
    # write your code
    if(len(A[0])!=len(B)):
      print("Not Possible")
    else:
      lst=[]
      for i in range(0,len(A)):
        lst1=[]
        for k in range(0,len(B[0])):
          sum=0
          for j in range(0,len(B)):
            sum+=A[i][j]*B[j][k]
          lst1.append(sum)
        lst.append(lst1)
      print(lst)

    #return(#multiplication_of_A_and_B)
A=[[1,2],
    [3,4]]
```

```
B=[[1,4],
    [5,6],
    [7,8],
    [9,6]]
matrix_mul(A, B)
```

```
Not Possible
```

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 [46]:

```
"""
Referred Analytics vidhya for the syntax and working of uniform function
"""
from random import uniform
# write your python code here
# you can take the above example as sample input for your program to test
# it should work for any general input try not to hard code for only given input examples


# you can free to change all these codes/structure
def pick_a_number_from_list(B,n2):
    x=int(uniform(0,n2))
    return B[x]

def sampling_based_on_magnitued(A):
    n=len(A)
    B=[]
    for i in range(n):
      n1=A[i]
      for j in range(n1):
        B.append(A[i])
    n2=len(B)
    for i in range(1,100):
        number = pick_a_number_from_list(B,n2)
        print(number)

A = [0,5,27,6,13,28,100,45,10,79]
sampling_based_on_magnitued(A)
```

```
27
100
27
79
79
100
28
79
27
100
100
45
79
79
45
27
79
100
100
```

45
100
100
100
27
100
45
45
45
100
28
28
27
100
100
27
100
27
100
27
79
13
28
100
45
79
100
79
100
100
28
45
100
100
45
6
45
28
79
100
100
28
45
45
45
100
79
100
100
79
79
28
100
6
100
100
13
28
45
100
28
45
100
45
13
27
45
100
45
79
100
27

```
28
100
100
27
100
28
28
79
```

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 [47]:

```
"""
Referred w3 schools for seeing the functions of re module
"""
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 examples

# you can free to change all these codes/structure
# String: it will be the input to your program
def replace_digits(String):
    s=""
    x=re.findall("[0-9]",String)
    for ele in x:
      s+="#"
    return(s) # modified string which is after replacing the # with digits
String="#2a$#b%c%561#"
replace_digits(String)
```

Out[47]:

```
'####'
```

Q4: Students marks dashboard

consider the marks list of class students given two lists
Students =
['student1','student2','student3','student4','student5','student6','student7','student8','student9','student10']
Marks = [45, 78, 12, 14, 48, 43, 45, 98, 35, 80]
from the above two lists the Student[0] got Marks[0], Student[1] got Marks[1] and so on

your task is to print the name of students a. Who got top 5 ranks, in the descending order of marks
b. Who got least 5 ranks, in the increasing order of marks
d. Who got marks between >25th percentile <75th percentile, in the increasing order of marks

```
Ex 1:
Students=['student1','student2','student3','student4','student5','student6','studen
t7','student8','student9','student10']
Marks = [45, 78, 12, 14, 48, 43, 47, 98, 35, 80]
a.
student8  98
student10 80
student2  78
student5  48
student7  47
```

```
b.
student3 12
student4 14
student9 35
student6 43
student1 45
c.
student9 35
student6 43
student1 45
student7 47
student5 48
```

In [48]:

```
# 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 examples

# you can free to change all these codes/structure
def display_dash_board(students, marks):
    # write code for computing top top 5 students
    n=len(marks)
    marks=sorted(marks,reverse=True)
    top_5_students = marks[0:5]
    # write code for computing top least 5 students
    marks=sorted(marks)
    least_5_students = marks[0:5]
    # write code for computing top least 5 students
    n1=n//4
    n2=3*n//4
    students_within_25_and_75 = marks[n1:n2]

    return top_5_students, least_5_students, students_within_25_and_75



Students=['student1','student2','student3','student4','student5','student6','student7','student8','student9','student10']
Marks = [45, 78, 12, 14, 48, 43, 47, 98, 35, 80]
d={}
n=len(Marks)
for i in range(n):
  d[Marks[i]]=Students[i];
top_5_students, least_5_students, students_within_25_and_75 = display_dash_board(Students, Marks)

print("a.")
for i in top_5_students:
  x=d.get(i)
  print(str(x)+" "+str(i))

print("b.")
for i in least_5_students:
  x=d.get(i)
  print(str(x)+" "+str(i))

print("c.")
for i in students_within_25_and_75:
  x=d.get(i)
  print(str(x)+" "+str(i))
```

```
a.
student8 98
student10 80
student2 78
student5 48
student7 47
b.
```

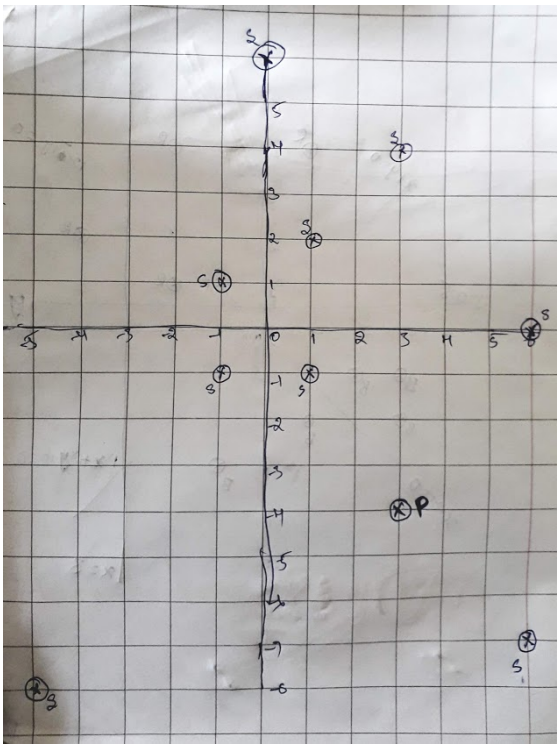
```
student3 12
student4 14
student9 35
student6 43
student1 45
c.
student9 35
student6 43
student1 45
student7 47
student5 48
```

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

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 [49]:

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

```
# here S is list of tuples and P is a tuple ot len=2
def closest_points_to_p(S, P):
    # write your code here
    n=len(S)
    lst=[]
    d={}
    for i in range(n):
      x=math.acos((S[i][0]*P[0]+S[i][1]*P[1])/(math.sqrt(S[i][0]**2+S[i][1]**2)*math.sq
rt(P[0]**2+P[1]**2)))
      lst.append(x)
      d[x]=S[i]
    lst=sorted(lst)
    closest_points_to_p=lst[0:5]
    return closest_points_to_p,d  # 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,d = closest_points_to_p(S, P)
for i in points:
  x=d.get(i)
  print(x)
```

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

Q6: Find Which line separates oranges and apples

consider you have given two set of data points in the form of list of tuples like

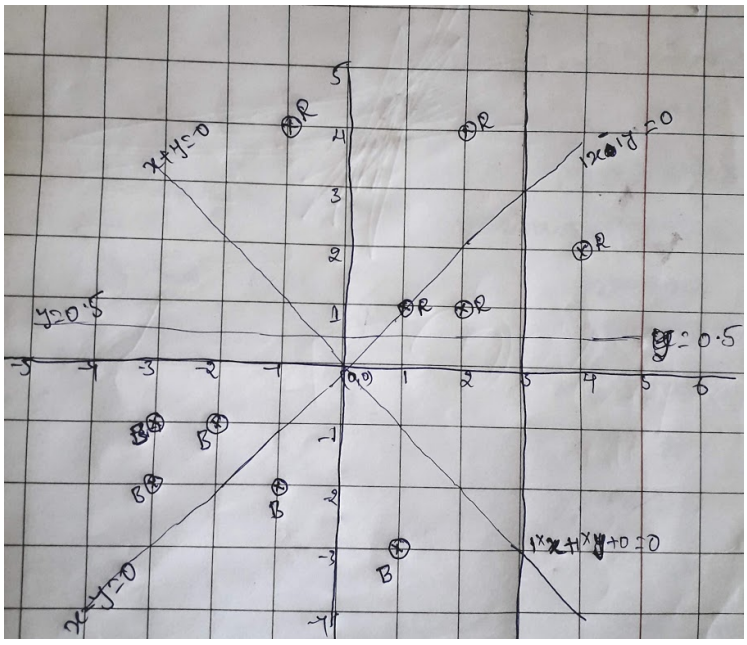
```
Red =[(R11,R12),(R21,R22),(R31,R32),(R41,R42),(R51,R52),..,(Rn1,Rn2)]
Blue=[(B11,B12),(B21,B22),(B31,B32),(B41,B42),(B51,B52),..,(Bm1,Bm2)]
```

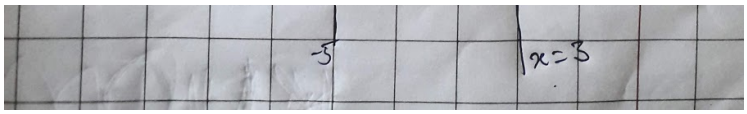
and set of line equations(in the string formate, i.e list of strings)

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

your task is to for each line that is given print "YES"/"NO", you will print yes, if all the red points are one side of the line and blue points are other side of the line, otherwise no

```
Ex:
Red= [(1,1),(2,1),(4,2),(2,4), (-1,4)]
Blue= [(-2,-1),(-1,-2),(-3,-2),(-3,-1),(1,-3)]
Lines=["1x+1y+0","1x-1y+0","1x+0y-3","0x+1y-0.5"]
```





Output:
YES
NO
NO
YES

In [50]:

```
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


# you can free to change all these codes/structure
def i_am_the_one(red,blue,line):
    n=len(red)
    res=[]
    n1=len(line)
    for i in range(n1):
      if(line[i].isdigit()):
        if(line[i-1]=="-"):
          x=int(line[i])-2*int(line[i])
          res.append(x)
        else:
          x=int(line[i])
          res.append(x)
    prod1=red[0][0]*res[0]+red[0][1]*res[1]+res[2]
    prod2=blue[0][0]*res[0]+blue[0][1]*res[1]+res[2]
    if(prod1>0 and prod2<0):
      for i in range(n):
        x=red[i][0]*res[0]+red[i][1]*res[1]+res[2]
        y=blue[0][0]*res[0]+blue[0][1]*res[1]+res[2]
        if(x<0 or y>0):
          return "No"
    elif(prod1<0 and prod2>0):
      for i in range(n):
        x=red[i][0]*res[0]+red[i][1]*res[1]+res[2]
        y=blue[0][0]*res[0]+blue[0][1]*res[1]+res[2]
        if(x>0 or y<0):
          return "No"
    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) # the returned value
```

```
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 [51]:

```
# 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):
    n=len(string)
    j=1
    m=1
    l=0
    y=0
    c=0
    lst=[]
    s=string.split(',')
    st=""
    n1=len(s)
    for i in range(n1):
       if(s[i].isdigit()):
         lst.append(i)
    n2=len(lst)
    if(n2==1):
      x=int(s[lst[0]])//n1
      for k in range(n1):
        st+=str(x)
        c+=1
        if(c!=(n1)):
          st+=','
      return st
```

```
    for i in lst:
      x=(int(s[i])+y)//(i+1-l)
      for k in range(l,i):
        st+=str(x)
        c+=1
        if(c!=n1):
          st+=','
      l=i
      y=x
    if(lst[n2-1]!=n1):
      x=y//(n1-l)
      for k in range(l,n1):
        st+=str(x)
        c+=1
        if(c!=n1):
          st+=','
    return st

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

```
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,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 [52]:

```
# 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(A):
    n=len(A)
    c1=0
    c2=0
    c3=0
    c4=0
    c5=0
    c6=0
    for i in range(n):
```

```
    if(A[i][1]=="S1"):
      c1+=1
      if(A[i][0]=="F1"):
        c2+=1
    if(A[i][1]=="S2"):
      c3+=1
      if(A[i][0]=="F1"):
        c4+=1
    if(A[i][1]=="S3"):
      c5+=1
      if(A[i][0]=="F1"):
        c6+=1
  s1=str(c2)+'/'+str(c1)
  s2=str(c4)+'/'+str(c3)
  s3=str(c6)+'/'+str(c5)
  print("a. P(F=F1|S==S1)="+s1+", P(F=F1|S==S2)="+s2+", P(F=F1|S==S3)="+s3)

  c1=0
  c2=0
  c3=0
  c4=0
  c5=0
  c6=0
  for i in range(n):
    if(A[i][1]=="S1"):
      c1+=1
      if(A[i][0]=="F2"):
        c2+=1
    if(A[i][1]=="S2"):
      c3+=1
      if(A[i][0]=="F2"):
        c4+=1
    if(A[i][1]=="S3"):
      c5+=1
      if(A[i][0]=="F2"):
        c6+=1
  s4=str(c2)+'/'+str(c1)
  s5=str(c4)+'/'+str(c3)
  s6=str(c6)+'/'+str(c5)
  print("b. P(F=F2|S==S1)="+s4+", P(F=F2|S==S2)="+s5+", P(F=F2|S==S3)="+s6)

  c1=0
  c2=0
  c3=0
  c4=0
  c5=0
  c6=0
  for i in range(n):
    if(A[i][1]=="S1"):
      c1+=1
      if(A[i][0]=="F3"):
        c2+=1
    if(A[i][1]=="S2"):
      c3+=1
      if(A[i][0]=="F3"):
        c4+=1
    if(A[i][1]=="S3"):
      c5+=1
      if(A[i][0]=="F3"):
        c6+=1
  s7=str(c2)+'/'+str(c1)
  s8=str(c4)+'/'+str(c3)
  s9=str(c6)+'/'+str(c5)
  print("c. P(F=F3|S==S1)="+s7+", P(F=F3|S==S2)="+s8+", P(F=F3|S==S3)="+s9)

  c1=0
  c2=0
  c3=0
  c4=0
  c5=0
  c6=0
  for i in range(n):
```

```
      if(A[i][1]=="S1"):
        c1+=1
        if(A[i][0]=="F4"):
          c2+=1
      if(A[i][1]=="S2"):
        c3+=1
        if(A[i][0]=="F4"):
          c4+=1
      if(A[i][1]=="S3"):
        c5+=1
        if(A[i][0]=="F4"):
          c6+=1
    s10=str(c2)+'/'+str(c1)
    s11=str(c4)+'/'+str(c3)
    s12=str(c6)+'/'+str(c5)
    print("d. P(F=F4|S==S1)="+s10+", P(F=F4|S==S2)="+s11+", P(F=F4|S==S3)="+s12)

    c1=0
    c2=0
    c3=0
    c4=0
    c5=0
    c6=0
    for i in range(n):
      if(A[i][1]=="S1"):
        c1+=1
        if(A[i][0]=="F5"):
          c2+=1
      if(A[i][1]=="S2"):
        c3+=1
        if(A[i][0]=="F5"):
          c4+=1
      if(A[i][1]=="S3"):
        c5+=1
        if(A[i][0]=="F5"):
          c6+=1
    s13=str(c2)+'/'+str(c1)
    s14=str(c4)+'/'+str(c3)
    s15=str(c6)+'/'+str(c5)
    print("e. P(F=F5|S==S1)="+s13+", P(F=F5|S==S2)="+s14+", P(F=F5|S==S3)="+s15)

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)
```

```
a. P(F=F1|S==S1)=1/4, P(F=F1|S==S2)=1/3, P(F=F1|S==S3)=0/3
b. P(F=F2|S==S1)=1/4, P(F=F2|S==S2)=1/3, P(F=F2|S==S3)=1/3
c. P(F=F3|S==S1)=0/4, P(F=F3|S==S2)=1/3, P(F=F3|S==S3)=1/3
d. P(F=F4|S==S1)=1/4, P(F=F4|S==S2)=0/3, P(F=F4|S==S3)=1/3
e. P(F=F5|S==S1)=1/4, P(F=F5|S==S2)=0/3, P(F=F5|S==S3)=0/3
```

Q9: Given two sentances S1, S2

You will be given two sentances S1, S2 your task is to find

a. Number of common words between S1, S2
b. Words in S1 but not in S2
c. Words in S2 but not in S1

Ex:

S1= "the first column F will contain only 5 uniques values"
S2= "the second column S will contain only 3 uniques values"
Output:
a. 7
b. ['first','F','5']
c. ['second','S','3']

In [53]:

```
# 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 Number_of_words_in_common(S1,S2):
  st1=S1.split(' ')
  st2=S2.split(' ')
  c=0
  lst=list(set(st1)&set(st2))
  return len(lst)

def S1_Minus_S2(S1,S2):
  st1=S1.split(' ')
  st2=S2.split(' ')
  c=0
  lst=list(set(st1)-set(st2))
  return lst

def S2_Minus_S1(S1,S2):
  st1=S1.split(' ')
  st2=S2.split(' ')
  c=0
  lst=list(set(st2)-set(st1))
  return lst


def string_features(S1, S2):
    a=Number_of_words_in_common(S1,S2)
    b=S1_Minus_S2(S1,S2)
    c=S2_Minus_S1(S1,S2)
    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. "+str(a))
print("b. "+str(b))
print("c. "+str(c))
```

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

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 $f(Y, Y_{score})$ here n is the number of rows in the matrix

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

Ex:

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 [54]:

```
# 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

"""
Referred stackoverflow for getting the limits formula for computing log of a value
"""

def log(x):
  n=100000.0
  y=n*((x**(1/n))-1)
  z=n*((10**(1/n))-1)
  return y/z
def compute_log_loss(A):
  n=len(A)
  loss=0
  for i in range(n):
    loss+=((A[i][0])*log(A[i][1]))+((1-A[i][0])*log(1-A[i][1]))
  loss*=(-1/n)
  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)
```

```
0.42430153411900057
```

In [ ]: