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
```

```
# 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

# This function returns the matrix created based on the rank
```

```
def formMatrix(rnklst, ellst):
  matrix=[]
  for i in range(int(rnklst[0])):
    row=[]
    for j in range(i*int(rnklst[1]), (i+1)*int(rnklst[1])):
      row.append(int(ellst[j]))
    matrix.append(row)
  return matrix

# This function returns the multipied matrix
def prodmatrix(matrixA, matrixB):
  finallst=[]
  for i in range(len(matrixA)):
    lst=[]
    for j in range(len(matrixB[0])):
      count=0
      for k in range(len(matrixA[0])):
        #print(matrixA[i][k],matrixB[k][j])
        count=count+matrixA[i][k]*matrixB[k][j]
      lst.append(count)
    finallst.append(lst)
  return finallst

A=input("Enter the rank of Matrix 'A':")
B=input("Enter the rank of Matrix 'B':")
rnkA=str.split(A,',')
rnkB=str.split(B,',')
if rnkA[1]==rnkB[0]:
  eleA = input("Enter "+str(int(rnkA[0])*int(rnkA[1]))+" elements of Matrix 'A':")
  eleB = input("Enter "+str(int(rnkB[0])*int(rnkB[1]))+" elements of Matrix 'B':")
  strlstA=str.split(eleA,',')
  strlstB=str.split(eleB,',')
  finalMatrix=prodmatrix(formMatrix(rnkA, strlstA), formMatrix(rnkB, strlstB))
  print("Multiplied Matrix is: ")
  print(finalMatrix)
else:
  print("Matrix multiplication is not possible for given ranks")
```

```
Enter the rank of Matrix 'A':1,1
Enter the rank of Matrix 'B':1,1
Enter 1 elements of Matrix 'A':2
Enter 1 elements of Matrix 'B':3
Multiplied Matrix is:
[[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)
```

```
import random
nmber = input("Enter the numbers:")
nmbrlst = str.split(nmber,',')
nmbrlstint = [int(x) for x in nmbrlst]
weighnumlst=[]
for i in nmbrlstint:
  weighnumlst.append(i/sum(nmbrlstint))
cumweighnumlst=[]
tempsum=0
for j in range(len(weighnumlst)):
  tempsum+=weighnumlst[j]
  cumweighnumlst.append(tempsum)
for k in range(100):
  ranNum=random.uniform(0.0,1.0)
  for l in range(len(cumweighnumlst)):
    if ranNum<=cumweighnumlst[l]:
      break
  print(nmbrlstint[l])
```

```
Enter the numbers:0,5,27,6,13,28,100,45,10,79
45
45
28
10
28
100
100
28
28
100
79
45
79
6
10
28
79
100
79
45
45
100
79
45
79
13
27
```

```
13
28
27
```

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: ####
```

```
sa
```

```
import re
strng=input("Enter a string:")
rstrng=re.sub('[^0-9]', '', strng)  # removes all the non numeric characters and keeps onl
re.sub('\d', '#', rstrng)           # replaces all numerics with '#'
```

```
Enter a string: #2a$#b%c%561#
'####'
```

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','
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
```

```
studName = input("Enter Student Names:")
studMarks = input("Enter Student Marks:")
sNamelst=str.split(studName,',')
sMarkslst=str.split(studMarks, ',')
sMarkslstint = [int(x) for x in sMarkslst]
studdetails = {}
for i in range(len(sNamelst)):
  studdetails[sNamelst[i]] = sMarkslstint[i]
descstudlst = sorted(studdetails.items(), key=lambda x:x[1], reverse=True)
ascstudlst = sorted(studdetails.items(), key=lambda x:x[1])
print("Top five students are:")
for j in range(5):
  print(descstudlst[j][0])
print("Least five students are:")
for k in range(5):
  print(ascstudlst[k][0])
percent_25 = 25*len(ascstudlst)//100
percent_75 = 75*len(ascstudlst)//100
print("25th Percentile Value:"+str(ascstudlst[percent_25][1]))
print("75th Percentile Value:"+str(ascstudlst[percent_75][1]))
print("Students between 25% and 75% values are:")
for l in range(len(ascstudlst)):
  if ascstudlst[percent_25][1]<ascstudlst[l][1]<ascstudlst[percent_75][1]:
    print(ascstudlst[l][0])
```

```
Enter Student Names:'student1','student2','student3','student4','student5','student6
Enter Student Marks:45, 78, 12, 14, 48, 43, 47, 98, 35, 1000
```

```
Top five students are:
'student10'
'student8'
'student2'
'student5'
'student7'
Least five students are:
'student3'
'student4'
'student9'
'student6'
'student1'
25th Percentile Value:35
75th Percentile Value:78
Students between 25% and 75% values are:
'student6'
'student1'
'student7'
'student5'
```

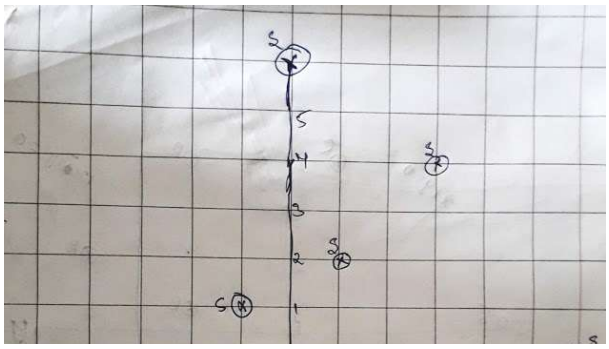
Q5: Find the closest points

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)



```
def closest_points(data_pts_lst,mesure_pt):
  dist_dict={}
  for i in range(0,len(data_pts_lst),2):
    dist = math.acos((data_pts_lst[i]*mesure_pt[0]+data_pts_lst[i+1]*mesure_pt[1])/((data_
    dist_dict[dist]=(data_pts_lst[i],data_pts_lst[i+1])
  print(dist_dict)
  five_closepts=sorted(dist_dict.items(), key=lambda x:x[0])[:5]
  return five_closepts

data_pts=input("Enter n datapoints:")
mesure_pt=input("Enter the point which distance to be measured:")
data_pts=data_pts.replace("(","")
data_pts=data_pts.replace(")","")
data_pts=data_pts.replace("[","")
data_pts=data_pts.replace("]","")
mesure_pt=mesure_pt.replace("(","")
mesure_pt=mesure_pt.replace(")","")
data_pts = str.split(data_pts, ',')
mesure_pt = str.split(mesure_pt, ',')
data_pts_lst=[int(x) for x in data_pts]
mesure_pt=[int(x) for x in mesure_pt]
five_closepts = closest_points(data_pts_lst,mesure_pt)
print("The 5 closest points are:")
for i in range(len(five_closepts)):
  print(five_closepts[i][1])
```

```
Enter n datapoints:(1,2),(3,4),(-1,1),(6,-7),(0, 6),(-5,-8),(-1,-1),(6,0),(1,-1)
Enter the point which distance to be measured:(3,-4)
{2.0344439357957027: (1, 2), 1.8545904360032246: (3, 4), 2.9996955989856287: (-1, 1),
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

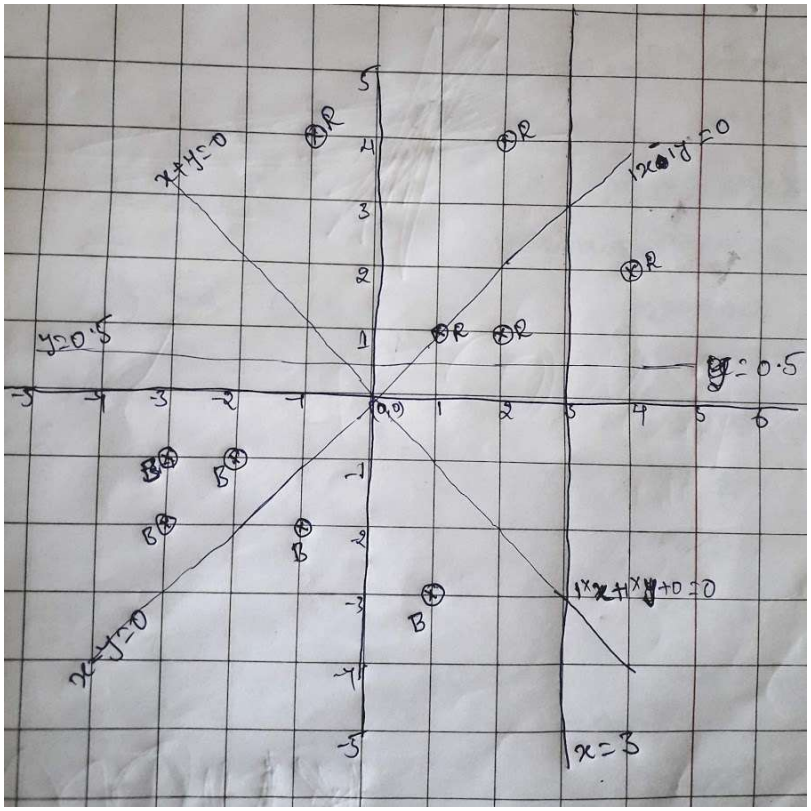
```
Red =[(R11,R12),(R21,R22),(R31,R32),(R41,R42),(R51,R52),..,(Rn1,Rn2)]
Blue=[(B11,B12),(B21,B22),(B31,B32),(B41,B42),(B51,B52),..,(Bm1,Bm2)]
```

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 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
```

```
# Executes multiplication for given string equation and retuns product of each point as a
# Addition or subtraction symbols should be eliminated before calling this function
def Multiply(pts_lst,eqn):
  product_lst = []
  for pt in pts_lst:
```

```
      num_product = 1
      if len(eqn) > 0:
        if eqn.find("x") > -1 and eqn.find("y") > -1:
          eqn1 = eqn.replace("x","*"+str(pt[0]))
          eqn1 = eqn1.replace("y","*"+str(pt[1]))
        elif eqn.find("x") > -1:
          eqn1 = eqn.replace("x","*"+str(pt[0]))
        elif eqn.find("y") > -1:
          eqn1 = eqn.replace("y","*"+str(pt[1]))
        else:
          eqn1 = eqn
        lst = eqn1.split("*")
        for elem in lst:
          num_product = num_product*float(elem)
      product_lst.append(num_product)
    return product_lst

  # This function retuns whether each point lies on positive or negative side
  def yes_or_No_Seperation(pts_lst,lineEq):
    if lineEq.find("+") or lineEq.find("-") > -1:
      # splits the given line equation using + symbol
      pos_line_lst = lineEq.split("+")
      Pos_Multi_lst = []
      for i in pos_line_lst:
        Neg_Multi_lst = []
        # splits the given line equation using - symbol
        if i.find("-") > -1:
          neg_line_lst = i.split("-")
          for j in neg_line_lst:
            multi_lst = Multiply(pts_lst,j)
            Neg_Multi_lst.append(multi_lst)
          # finds the difference for each point in the list
          for counter in range(len(Neg_Multi_lst)-1):
            if counter == 0:
              nlist1 = Neg_Multi_lst[counter]
            nlist2 = Neg_Multi_lst[counter+1]
            diff_list = []
            for i in range(len(nlist1)):
              difference = nlist1[i]-nlist2[i]
              diff_list.append(difference)
            nlist1 = diff_list.copy()
          Pos_Multi_lst.append(nlist1)
        else:
          multi_lst = Multiply(pts_lst,i)
          Pos_Multi_lst.append(multi_lst)
      # finds the sum for each point in the list
      for counter in range(len(Pos_Multi_lst)-1):
        if counter == 0:
          plist1 = Pos_Multi_lst[counter]
        plist2 = Pos_Multi_lst[counter+1]
        sum_list = []
        for i in range(len(plist1)):
          sum = plist1[i]+plist2[i]
          sum_list.append(sum)
        plist1 = sum_list.copy()
      Yes_or_no_lst = []
```

```
    yes_or_no_lst = []
    for item in plist1:
      if item >= 0:
        yes_or_no_lst.append("Yes")
      else:
        yes_or_no_lst.append("No")
  return yes_or_no_lst

# Input for the code
Red= [(1,1),(2,1),(4,2),(2,4),(-1,4),(-6,2)]
Blue= [(-2,-1),(-1,-2),(-3,-2),(-3,-1),(1,-3),(2.3,-0.9)]
Lines=["1x+1y+0","1x-1y+0","1x+0y-3","0x+1y-0.5","0.5xy+7x-3y-235.26"]

# for each line equation it takes the red point list, blue point list and evaluates on whi
for line in Lines:
  rdptlst = yes_or_No_Seperation(Red,line)
  bluptlst = yes_or_No_Seperation(Blue,line)
  if (rdptlst.count("Yes") == len(rdptlst) and bluptlst.count("No") == len(bluptlst)) or (
    print("Yes")
  else:
    print("No")
```

```
No
No
No
Yes
No
```

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

Ex 2: 40, _, _, _, 60 ==> (60+40)/5,(60+40)/5,(60+40)/5,(60+40)/5,(60+40)/5 ==> 20, 20,

Ex 3: 80, _, _, _, _  ==> 80/5,80/5,80/5,80/5,80/5 ==> 16, 16, 16, 16, 16 i.e. the 80 is

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, _, _, _,
    b. now distribute the sum (10+50) missing values in between (10, 10, 12, 12, 12, 12,
    c. now we will distribute 12 to right side missing values (10, 10, 12, 12, 12, 12, 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
```

```
# Given number is divided by length of the list and returns the same length list by fillin
def only_one_numeric(number, lst):
  mod_lst = []
  num = int(number)//len(lst)
  for j in range(len(lst)):
    mod_lst.append(num)
  return mod_lst
# Calculates the average of given list and fill the same number at each position of the li
def first_last_numeric(lst):
  mod_lst = []
  num = (int(lst[0])+int(lst[len(lst)-1]))//len(lst)
  for j in range(len(lst)):
    mod_lst.append(num)
  return mod_lst
# based on the input series this calls the appropriate function and fills the series
def fill_series(ser_lst):
  pos_lst=[]
  for i in range(len(ser_lst)):
    if ser_lst[i].isnumeric() == True:
      pos_lst.append(i)
  if len(pos_lst) == 1:
    mod_ser_lst = only_one_numeric(ser_lst[pos_lst[0]], ser_lst)
  elif len(pos_lst) == 2 and ser_lst[pos_lst[0]].isnumeric()==True and ser_lst[len(ser_lst
    mod_ser_lst = first_last_numeric(ser_lst)
  else:
    mod_ser_lst = []
    for pos in range(len(pos_lst)):
      if pos_lst[pos] == 0:
        ser_lst1 = ser_lst[:pos_lst[pos+1]+1]
        mod_ser_lst1 = first_last_numeric(ser_lst1)
        mod_ser_lst.append(mod_ser_lst1)
      elif pos == 1 and pos_lst[pos-1] == 0:
        continue
      else:
        if pos ==0:
          ser_lst1 = ser_lst[:pos_lst[pos]+1]
          mod_ser_lst1 = only_one_numeric(ser_lst[pos_lst[pos]], ser_lst1)
```

```
            mod_ser_lst.extend(mod_ser_lst1)
          else:
            ser_lst1 = ser_lst[pos_lst[pos-1]+1:pos_lst[pos]+1]
            calc_num = int(ser_lst[pos_lst[pos]])//(pos_lst[pos]-pos_lst[pos-1])
            mod_ser_lst[len(mod_ser_lst)-1] = calc_num
            mod_ser_lst1 = only_one_numeric(ser_lst[pos_lst[pos]], ser_lst1)
            mod_ser_lst.extend(mod_ser_lst1)
            if pos == len(pos_lst)-1 and len(ser_lst) > pos_lst[pos]:
              ser_lst[pos_lst[pos]] = mod_ser_lst[len(mod_ser_lst)-1]
              ser_lst1 = ser_lst[pos_lst[pos]:]
              calc_num = int(ser_lst[pos_lst[pos]])//(len(ser_lst)-pos_lst[pos])
              mod_ser_lst[len(mod_ser_lst)-1] = calc_num
              mod_ser_lst1 = only_one_numeric(ser_lst[pos_lst[pos]], ser_lst1)
              mod_ser_lst1 = mod_ser_lst1[:len(mod_ser_lst1)-1]
              mod_ser_lst.extend(mod_ser_lst1)
    return mod_ser_lst
# accesing input
series = input("Enter the series: ")
ser_lst = series.split(",")
filled_series = fill_series(ser_lst)
#printing final filled series
print("Series is filled as below:")
print(filled_series)
```

```
Enter the series: _,_,30,_,_,_,50,_,_
Series is filled as below:
[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
```

```
# input matrix
matA = [['F1','S1'],['F2','S2'],['F3','S3'],['F1','S2'],['F2','S3'],['F3','S2'],['F2','S1'
fist_col = []
second_col = []
# creating two lists, Lst 1 consists of first and second column combination and lst 2 cons
for i in matA:
  fist_col.append(i[0]+i[1])
  second_col.append(i[1])
# prining the probabilities based on the condition
print("P(F=F1|S==S1)="+str(fist_col.count('F1S1')/second_col.count('S1'))+", "+"P(F=F1|S==
print("P(F=F2|S==S1)="+str(fist_col.count('F2S1')/second_col.count('S1'))+", "+"P(F=F2|S==
print("P(F=F3|S==S1)="+str(fist_col.count('F3S1')/second_col.count('S1'))+", "+"P(F=F3|S==
print("P(F=F4|S==S1)="+str(fist_col.count('F4S1')/second_col.count('S1'))+", "+"P(F=F4|S==
print("P(F=F5|S==S1)="+str(fist_col.count('F5S1')/second_col.count('S1'))+", "+"P(F=F5|S==
```

```
P(F=F1|S==S1)=0.25, P(F=F1|S==S2)=0.3333333333333333, P(F=F1|S==S3)=0.0
P(F=F2|S==S1)=0.25, P(F=F2|S==S2)=0.3333333333333333, P(F=F2|S==S3)=0.33333333333333
P(F=F3|S==S1)=0.0, P(F=F3|S==S2)=0.3333333333333333, P(F=F3|S==S3)=0.33333333333333
P(F=F4|S==S1)=0.25, P(F=F4|S==S2)=0.0, P(F=F4|S==S3)=0.3333333333333333
P(F=F5|S==S1)=0.25, P(F=F5|S==S2)=0.0, P(F=F5|S==S3)=0.0
```

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']
```

```
# Accessing inputs
sent1 = input("Enter the sentence1: ")
sent2 = input("Enter the sentence2: ")
# Splitting the string into sentences
sent1_lst = sent1.split(" ")
sent2_lst = sent2.split(" ")
# Converting list to set
sent1_set = set(sent1_lst)
```

```
sent2_set = set(sent2_lst)
# finding intersection for sets
no_of_comonwrds = len(sent1_set.intersection(sent2_set))
print(no_of_comonwrds)
# fnding S1-S2 and S2-S1
Wrds_S1_nt_S2 = sent1_set.difference(sent2_set)
Wrds_S2_nt_S1 = sent2_set.difference(sent1_set)
print(Wrds_S1_nt_S2)
print(Wrds_S2_nt_S1)
```

```
Enter the sentence1: the first column F will contain only 5 uniques values
Enter the sentence2: the second column S will contain only 3 uniques values
7
{'first', '5', 'F'}
{'S', '3', '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

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

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
```

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

```
import math
# input point list
pts_lst = [[1, 0.4], [0, 0.5], [0, 0.9], [0, 0.3], [0, 0.6], [1, 0.1], [1, 0.9], [1, 0.8]]
calc_num = 0
# performing math operation based on the formula
for i in pts_lst:
  num = i[0]*math.log10(i[1])+(1-i[0])*math.log10(1-i[1])
  calc_num+=num
calc_num = -1/len(pts_lst)*calc_num
print(calc_num)
```

```
0.42430993457031635
```

✓ 0s completed at 6:39 AM