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 [7]:
```
# 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):
    # first, find #col of A and #row of B
    ncolA = len(A[0]) # it will return no.of element in first row(means no.of column in matrix A)
    nrowB = len(B)# it will return no.of row of matrix B

    # let's check for matrix multiplication
    # #col of A should equal to #row of B
    if ncolA != nrowB:
        print('Not Possible')

    else:
        # initialize a 2d matrix using list comprehension
        matrix =[[0 for i in range(len(A))]for j in range(len(B[0]))]

        # implement the logic of matrix implementation
        for i in range(len(A)): # iterate #row in matrix A
            for j in range(len(B[0])): # iterate for #col in matrix B

                # finding resultant matrix's element
                for k in range((len(A[0]))):

                    matrix[i][j] += A[i][k] * B[k][j]

        # return resultant matrix
        return(matrix)
```

In [8]:
```
# case 1: #colA == #rowB
# initialize two matrix
A=[
    [1,2,3],
    [4,5,6]
]

B=[
    [3,4],
    [4,6],
    [4,3]
]
matrix_mul(A,B)
```

Out[8]:
```
[[23, 25], [56, 64]]
```

In [10]:
```
# case 2: #colA != #rowB
A=[
    [1,2,3],
    [4,5,6]
]

B=[
    [5,6],
    [3,4],
    [4,6],
    [4,3]
]
matrix_mul(A,B)
```

```
Not Possible
```

In [ ]:

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 [244]:
```
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(A):
    A=A.copy()
    Sum = 0 # taken 'Sum' as a variable b'z 'sum' is predefined in python
    #print(A)

    A.sort()

    #after sorting add first number twice
    A.insert(0,A[0])
    #print(A)

    #find length of A(list)
    lengthA = len(A)

    #calculate total sum
    for i in range(lengthA):
        Sum += A[i]

    # divide every element of list by it's total sum
    weightedA = []
    for i in range(lengthA):
        res = A[i] /Sum
        weightedA.append(res)
    #print(weightedA)
    # let's find cummulative sum of weightedA
    cumulativeA =[]
    cumulativeA.append(weightedA[0])
    for i in range(1,lengthA):
        cumulativeA.append(weightedA[i])
        cumulativeA[i] = cumulativeA[i]+ cumulativeA[i-1]
    cumulativeA.sort()
    #print(cumulativeA)


    # find a uniform random value between 0.0 and 1.0 because our cumulative list in the range[0.0,1.0]
```

```
    #pick a uniform random variable
    uniform_value = uniform(0.0,1.0)
    #print(uniform_value)

    # let's check the condition
    for i in range(1,lengthA):
        if  uniform_value<cumulativeA[i] and uniform_value>=cumulativeA[i-1]:
            # return selected number from the list
            return A[i]
```

In [245]:
```
def sampling_based_on_magnitude():
    found_random_numbers =[] # list to be used to find the probability of every element
    for i in range(1,100):
        number = pick_a_number_from_list(A)
        found_random_numbers.append(number)
        print(number)

    return found_random_numbers
```

In [246]:
```
A = [0, 5, 27, 6, 13, 28, 100, 45, 10, 79]

found_random_numbers = list(sampling_based_on_magnitude())
```

```
79
79
100
100
45
100
100
100
28
100
13
79
100
10
100
100
28
100
79
79
100
100
100
100
27
13
45
27
100
100
13
45
45
27
6
79
45
27
79
28
100
100
10
```

```
100
79
79
79
100
28
100
45
100
45
100
100
27
79
100
45
100
28
45
5
10
79
6
45
28
6
6
45
45
100
45
27
27
10
28
79
10
45
45
10
100
28
45
```

```
79
6
100
79
100
28
28
100
79
79
100
79
100
```

In [ ]:

In [247]:
```
# let's find probability of every element

probability={}
for i in range(len(A)):
    count = 0
    for j in range(99):
        if A[i] == found_random_numbers[j]:
            count+=1

    probability[A[i]]=count/99
```

In [248]:
```
probability
```

Out[248]:
```
{0: 0.0,
 5: 0.010101010101010102,
 27: 0.0707070707070707,
 6: 0.050505050505050504,
 13: 0.030303030303030304,
 28: 0.10101010101010101,
 100: 0.3333333333333333,
 45: 0.16161616161616163,
 10: 0.06060606060606061,
 79: 0.18181818181818182}
```

We can see in above code that 100 has highest probability, right.

In [ ]:

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 [103]:
```
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):

    list_of_string = list(String)

    result_string=[]
    #initializing digit_count
    digit_count=0

    for i in range(len(list_of_string)):

        if list_of_string[i].isdigit():
            result_string.append('#')

            # count no.of occurences of digits
            digit_count+=1

    # check if digit_count=0 or not
    if digit_count == 0:
        print('empty string')
    else:
        # join the elements of list and return it
        return ''.join(result_string) # modified string which is after replacing the # with digits
```

In [105]:
```
replace_digits('3e4e5r')
```

Out[105]: '###'

In [106]:
```
replace_digits('12345')
```

Out[106]: '#####'

In [108]:
```
replace_digits('The NO.1 Chekuri Sir!')
```

Out[108]:
```
'#'
```

In [109]:
```
replace_digits('Chekuri verma sir')
```

```
empty string
```

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','student7','student8','student9','stud
ent10']
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
```

In [178]:
```
# learned the concept from : https://www.dummies.com/education/math/statistics/how-to-calculate-percentiles-in-statistics/

import math
def percentile(A,percent):
    Data = A.copy()
    # sort the list
    Data.sort()


    # multiply percent by the total number of values n, n is index here of last element
    index = (len(Data) -1) *percent

    # check whether index is whole number or not

        #let's find floor and ceil value of given index
    index_floor = math.floor(index)
    index_ceil  = math.ceil(index)

        # if both index_floor and index_cell is equal
        # it means index is whole number
    if index_floor == index_ceil:
        return (Data[index]+Data[index+1])/2

    #else round the index to nearest integer
    else:
        rounded_index = round(index)
        return Data[rounded_index]
```

In [186]:
```
############################################################
# reference: geeksforgeeks for few concepts(not code)      #
############################################################

# 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):
    #let's first zip the given two lists and change it into dictionary
    zip_stu_marks = dict(zip(students, marks))


    # write code for computing top top 5 students
    # let's sort zipped list
    sorted_top_students = sorted(zip_stu_marks.items(), key= lambda x:(x[1],x[0]),reverse=True)

    # assign top 5 students
    top_5_students = sorted_top_students[0:5]


    # write code for computing top least 5 students
    least_5_students = sorted(zip_stu_marks.items(), key= lambda x:(x[1],x[0]),reverse=False)[0:5]



    # write code for computing top least 5 students

    # find total marks
    total_marks = 0
    for i in range(len(marks)):
        total_marks += marks[i]

    #find 25%ile
    percentile_25 = percentile(marks,0.25)
    #print(percentile_25)

    #find 75%ile
    percentile_75 = percentile(marks,0.75)
```

```
    #print(percentile_75)


    # sorte the students in increasing order of marks
    sorted_students = sorted(zip_stu_marks.items(), key= lambda x:(x[1],x[0]),reverse=False)

    # initializing list to store result
    students_within_25_and_75=[]

    # iterating over complete sorted list and finding desired student
    for i in range(len(sorted_students)):
        if int(sorted_students[i][1]) < int(percentile_75) and int(sorted_students[i][1]) >= int(percentile_2
5):
            students_within_25_and_75.append(sorted_students[i])


    # finally return three found results
    return top_5_students, least_5_students, students_within_25_and_75
```

In [183]:
```
students=['student1','student2','student3','student4','student5','student6','student7','student8','student9',
'student10']
marks = [45, 78, 12, 14, 48, 43, 47, 98, 35, 80]
```

In [187]:
```
top_5_students, least_5_students, students_within_25_and_75 = display_dash_board(students, marks)
```

In [191]:
```
print('a')
for student,mark in top_5_students: # tuple unpacking is used here
    print(student,mark)

print('\nb')
for student,mark in least_5_students:
    print(student,mark)

print('\nc')
for student,mark in students_within_25_and_75:
    print(student,mark)
```

```
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 [ ]:

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)

In [230]:
```
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
    closest_points_to_p = [] # initializing resultant list

    point_dict={} # initializing dictinary to store the points and their distances

    xq,yq = P #unpack P(tuple)

    for point in S:
        x1,y1 = point # unpack point to axis value

        # let's find distance from P(x1,y1) to point(xq,yq) using cosine distance
        dist = math.acos( (x1*xq +y1*yq)/ ( (math.sqrt( (x1**2)+(y1**2) ))*(math.sqrt((xq**2)+(yq**2))) )  )
        point_dict[point] = dist


    #let's sort the dictionary by it's key(distance), sorted function returns list
    point_list = sorted(point_dict.items(),key = lambda x: (x[1]),reverse=False)

#     print(point_list)
    for i in range(5):
        closest_points_to_p.append(point_list[i][0])

    # finally returns the list of closest point
    return closest_points_to_p  # its list of tuples
```

In [231]:
```
S= [(1,2),(3,4),(-1,1),(6,-7),(0, 6),(-5,-8),(-1,-1),(6,0),(1,-1)]
P= (3,-4)
```

In [233]:
```
points = closest_points_to_p(S, P)

#print the returned values
for i in points:
    print(i)
```

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

In [ ]:
```
# intentionally left blank
```

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

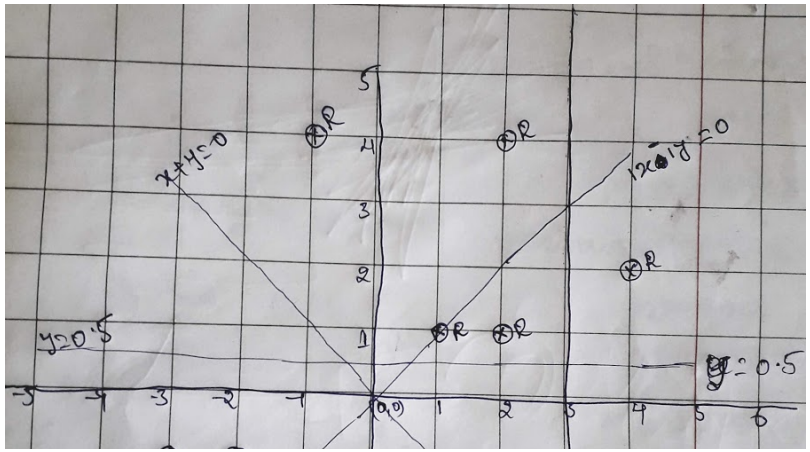
```
Red =[(R11,R12),(R21,R22),(R31,R32),(R41,R42),(R51,R52),..,(Rn1,Rn2)]
Blue=[(B11,B12),(B21,B22),(B31,B32),(B41,B42),(B51,B52),..,(Bm1,Bm2)]
```

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 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"]



In [105]:
```
def coefficient_finder(expression):
    """
    expression ex: ax+by+c

    returns three value: a,b,c
    this function returns coefficient of x and y.
    also reutrns value of c.

    >>> a,b,c = coefficient_finder('1050x-500y+10000')
    >>> a,b,c
    output: (1050.0, -500.0, 10000.0)
    """
    temp_a=expression.split('x')
    a = temp_a[0]
    temp_b = temp_a[1].split('y')
    b = temp_b[0]
    c = temp_b[1]
    return float(a), float(b), float(c)
```

In [116]:
```
def i_am_the_one(red,blue,line):

    # find the coefficients
    a,b,c = coefficient_finder(line) # here m is slope

    # find slope and y-intercept(c)

    if b!=0: # if b is not equal to 0, then divide it by y's coefficient
        m = ((-1)*float(a))/b
        c = ((-1)*float(c))/b
    else:
        m = ((-1)*float(a))
        c = ((-1)*float(c))

    #initializing total_red_points and total_blue_points
    total_red_points = 0
    total_blue_points= 0

    # check for all red points that which side of line it belongs to
    # if point lies below the list subtract 1 from total_red_point
    # if point lies above the list add +1 to the total_red_point


    # iterate for every point in red list
    for x,y in red:

        # if y>mx+c, it means point lies above the line
        if y > (m*x + c):
            # so, add +1 to the total_red_points
            total_red_points +=1
        # if y<mx+c, it means point lies below the line
        elif y < (m*x + c):
            # so, subtract 1 from the total_red_points
            total_red_points -=1
        #elif y== m*x + c  # means point is on the line
        #    do nothing

    # do above task for blue points also
    for x,y in blue:
        if y > (m*x + c):
            total_blue_points +=1
        elif y< (m*x + c):
```

```
            total_blue_points -=1


    # if length of given red points list is equal to found total_red_points
    # and len(red list) is equal to found total_blue_points
    # then given line well separates given points, so return 'YES'. otherwise 'NO'

    if len(red) == abs(total_red_points) and len(blue)== abs(total_blue_points):

        # check, if both category of points(red_points and blue_points) has opposite sign or not
        if (total_red_points<0 and total_blue_points > 0) or (total_red_points>0 and total_blue_points < 0):
            return 'YES'
    else:
        return 'NO'
```

In [117]:
```
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"]
```

In [118]:
```
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
```

In [ ]:
```

```

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 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 [29]:
```
##############################################################################
### I have used print() function, just for testing purpose ##################
### Author: Md Iqbal Bazmi,student at AAIC,pvt,ltd.##########################
##############################################################################
##############################################################################
##############################################################################

# let's design the function for case1,case2, and case3
# case1: _,_,_,40
# case2: 40_,_,_,_
# case3: 10,_,_,_,20

# design function for case1: _,_,_,40
def case1(in_list):
    #find length of in_list
    len_in_list = len(in_list)

    # find the last digit
    last_digit = in_list[-1]

    # calculate value
    value = int(last_digit)/len_in_list

    # finally return the value
    return int(value)

# design function for case2: 40,_,_,_
def case2(in_list):
    # find the length of in_list
    len_in_list = len(in_list)

    # find the first digit
    first_digit = in_list[0]

    # calculate value
    value = int(first_digit)/int(len_in_list)

    # return value
    return int(value)

# design function for case3: 10,_,_,_,20
def case3(in_list):
```

```
    #find the length of in_list
    len_in_list = len(in_list)

    # find first and last digit
    first_digit = int(in_list[0])
    last_digit = int(in_list[-1])

    # calculate value
    value = (first_digit+last_digit)/len_in_list

    # return the value
    return int(value)
```

In [85]:
```
def curve_smoothing(string):

    #initializing count by 0, it will help to keep track of elements
    count = 0

    #convert the given string into list
    string_list = string.split(',')




    # convert string digit to int value
    for i in range(len(string_list)):
        if string_list[i] != '_':
            string_list[i] = int(string_list[i])


    # iterate count <= len(string_list)
    while(count<=len(string_list)):


        # case1: '_,_,_,40', so check whether list starts with '_'
        if string_list[0] == '_':

            # create temporary list to store the desired list
            temp_list=[]

            # flag is used to insert the value in the temporary list
            flag = True

            # iterate till flag==True
            while flag==True:

                #check whether item is '_' or not.
                if string_list[count]=='_':
                    # If yes, append to the temporary list
                    temp_list.append(string_list[count])
                    # and increment the counter
                    count+=1

                # if item is not '_', check if it is digit.
```

```
            elif str(string_list[count]).isdigit():
                #if it is digit then append the item to the temporary list
                # and don't increment the count
                temp_list.append(string_list[count])

                # set flag as 'False' b'z we stop here inserting element into temporary list
                # because starting with '_' and end with 'digit' is completely a case(case1)
                flag=False

        # come out of the data inserting loop

        # pass the whole temporary sublist to the case1 function
        # case1 function will do further operation and will return distribution_item
        returned_item = case1(temp_list)


        # Distribute the returned distribution_item to the main list(string_list)
        for i in range(count+1):
            string_list[i] = returned_item


        # Now check whether sublist and main_list is equal in length or not
        # if both are equal, means we should return the main_list
        if len(string_list) == len(temp_list):
            return string_list # returning list case1

    # case2 or case 3: If not started with '_'
    # check if it is digit
    elif str(string_list[count]).isdigit():
        # if it is digit, store start_index
        # initialize start_index by value of count
        start_index = count
        # Q: Why we are storing start_index?
        # A: Because case3('_,_,"23,_,_,_"')(double quoted string is case3) might lies in
        # between the whole string


        # initialize temporary list
        temp_list = []

        # append the first digit to the temporary list
        temp_list.append(string_list[count])
        # increment the count value
```

```
            count+=1

            # set flag=True for appending the item into the temporary list
            flag = True

            # iterate till flag==True.
            while flag==True:

                # check whether current element is '_' or not
                if string_list[count] == '_':
                    # if so, then check whether it is last item or not
                    if count!= len(string_list):
                        # If so, then append the current item to the temporary list
                        temp_list.append(string_list[count])
                        count+=1 # increment the value of count

                        # check at every insertion of element that current element is last or not
                        if count== len(string_list):
                            # if the current item is last item,
                            # then, it belongs to case2(40,_,_,_)
                            # so, pass the temp_list to the case2 function
                            returned_item = case2(temp_list)

                            # distribute the returned_item to the specified index
                            for i in range(start_index,count):
                                string_list[i] = returned_item

                            # finally, return the string_list(main list)
                            return string_list # returning list case2

                # if given string is starting wiht 'digit' and not ending with '_'
                # it goes to case3(30,_,_,30)

                # so, check whether current element is digit or not
                elif str(string_list[count]).isdigit():

                    # now, check whether found item is last or not
                    if count!= len(string_list):

                        # if current digit is not last element,
                        # append the current element
                        temp_list.append(string_list[count])
                        flag=False
```

```
                    # pass it to case3 function
                    returned_item = case3(temp_list)

                    # update the original list
                    for i in range(start_index,count+1):
                        string_list[i] = returned_item

                    # check if the current digit is last element or not
                    if count == len(string_list)-1:
                        # if so, then return string_list
                        return string_list
```

In [86]:
```
def returnString(smoothed_value):
    '''
        this function return the string format of list.
        with ',' separation

        >>> returnString([10,20,30])
        >>> '10,20,30'
    '''
    test=[]
    for i in range(len(smoothed_value)):
        test.append(str(smoothed_value[i]))
        test.append(str(','))
    test.pop()
    string = ''.join(test)
    return string
```

In [87]:
```
S=  "_,_,30"
smoothed_values= returnString(curve_smoothing(S))
print(smoothed_values)
```

```
10,10,10
```

In [88]:
```
S=  "30,_,_,30"
smoothed_values= returnString(curve_smoothing(S))
print(smoothed_values)
```

```
15,15,15,15
```

In [89]:
```
S=  "30,_,_"
smoothed_values= returnString(curve_smoothing(S))
print(smoothed_values)
```

```
10,10,10
```

In [90]:
```
S='_,_,_,40,_,_,_,70,_,_,_,64,_,_,_'
smoothed_values = returnString(curve_smoothing(S))
smoothed_values
```

Out[90]:
```
'10,10,10,16,16,16,16,16,16,16,16,4,4,4,4'
```

In [ ]:
```
# luckily, it just took me half of the day, not a full day to solve this.
```

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 [119]:
```
# create function to find first and second column

def first_second_col(A):
    '''
    It returns the list of both first and second column of given 2-dimensional matrix
    '''
    # initialize temp first column
    first_col = []
    second_col= []

    for element in A: # iterate for every element in A
        first_col.append(element[0]) # appending elements to the list
        second_col.append(element[1])

    return first_col,second_col # return both lists

# create unique list returning function

def unique_element_list(A):
    '''
        It returns the list of unique elements from given list.
    '''
    # initialize temporary list
    temp_list = []

    # iterate over all element of A
    for element in A:
        if element not in temp_list:
            temp_list.append(element)

    return temp_list
```

In [120]:
```
def conditional_probability(A):

    # first find the unique list of main given list
    unique_list = unique_element_list(A)


    # then find the first column list and second column list
#     F = first_column(A)
#     S = second_column(A)
    F,S = first_second_col(A)

    # Now, find the unique first_elements list and unique second elements'list
    # Q: Why find unique_f and unique_s
    # A: to find no.of possible probabilities to be computed
    unique_f = unique_element_list(F)
    unique_s = unique_element_list(S)


    for f in unique_f:
        for s in unique_s:

            #probability = A.count([f,s])/S.count(s)
            print('P(F={}|S=={}) = {}/{}'.format(f,s,A.count([f,s]),S.count(s)),end=' ')
        print('')
```

In [121]:
```
main_list = [
                ['F1','S1'],['F2','S2'],
                ['F3','S3'],['F1','S2'],
                ['F2','S3'],['F3','S2'],
                ['F2','S1'],['F4','S1'],
                ['F4','S3'],['F5','S1'],
            ]
```

In [122]:
```
# find conditional probabilities
conditional_probability(main_list)
```

```
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
```

In [ ]:
```
```

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 [97]:
```
# 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 string_features(S1, S2):

    # given S1 and S2 are strings
    # so, split it into list of words
    S1 = S1.split(' ')
    S2 = S2.split(' ')

    # let's solve the problem 'a'
    # a. Number of common words between S1,S2
    common_words = 0
    common_list = []

    # iterate for all words in S1
    for i in S1:
        if i in S2: # if S1's word is also present in S2
            common_list.append(i) #then, append to the common_list
# you can use either this code
# uncomment below code to calculate above solutino
#     for j in S2:
#         if j in S1:
#             common_list.append(j)

    a = len(common_list) # find the length of common list and store it to a

    # let's solve problem 'b'
    # b. Words in S1 but not in S2
    b=[]
    for i in S1:
        if i not in S2:
            b.append(i)

    # let's solve problem 'c'
    # c. words in S2 but not in S1
    c=[]
    for j in S2:
        if j not in S1:
            c.append(j)
```

```
    return a, b, c
```

In [99]:
```
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,'\n',b,'\n',c)
```

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

In [100]:
```
S1= "I am thinking to love you"
S2= "I am thinking not to leave you"
a,b,c = string_features(S1, S2)
print(a,'\n',b,'\n',c)
```

```
5
 ['love']
 ['not', 'leave']
```

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
```

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

In [103]:
```
# 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
import math

# you can free to change all these codes/structure
def compute_log_loss(A):

    # let's find length of A, first.
    lengthA = len(A)

    log_loss = 0
    Sum = 0
    for item in A:
        y = item[0]
        y_score = item[1]

        Sum += (y* math.log(y_score,10) + (1-y)* math.log(1-y_score,10))

    loss = (-1/lengthA) * Sum
    return loss
```

In [104]:
```
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
```

In [ ]:
```

```