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 [15]:

```
import random
R1 = int(input("Enter the number of rows for mat1:"))
C1 = int(input("Enter the number of columns for mat1:"))
mat1 = [[int(input()) for x1 in range (C1)] for y1 in range(R1)]
R2 = int(input("Enter the number of rows for mat2:"))
C2 = int(input("Enter the number of columns for mat2:"))
mat2 = [[int(input()) for x2 in range (C2)] for y2 in range(R2)]
def matrix_mul(mat1, mat2):
    # write your code
    result = [ [ 0 for i in range(len(mat2[0])) ] for j in range(len(mat1)) ]
    for i in range(len(mat1)):
        for j in range(len(mat2[0])):
            for k in range(len(mat2)):
                result[i][j] += mat1[i][k] * mat2[k][j]
    return result

matrix_mul(mat1, mat2)
```

```
Enter the number of rows for mat1:3
Enter the number of columns for mat1:3
1
3
4
2
5
7
5
9
6
Enter the number of rows for mat2:3
Enter the number of columns for mat2:3
1
0
0
0
1
0
0
0
1
```

Out[15]:

```
[[1, 3, 4], [2, 5, 7], [5, 9, 6]]
```

Q2: Select a number randomly with probability proportional to its magnitude from the given array of n elements

consider an experiment, selecting an element from the list A randomly with probability proportional to its magnitude. assume we are doing the same experiment for 100 times with replacement, in each experiment you will print a number that is selected randomly from A.

```
Ex 1: A = [0 5 27 6 13 28 100 45 10 79]
let f(x) denote the number of times x getting selected in 100 experiments.
f(100) > f(79) > f(45) > f(28) > f(27) > f(13) > f(10) > f(6) > f(5) > f(0)
```

In [16]:

```
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 exampl
es

A = [0, 5, 27, 6, 13, 28, 100, 45, 10, 79]
# you can free to change all these codes/structure
def pick_a_number_from_list(A):
    # your code here for picking an element from with the probability propotional to it
s magnitude
    #.
    #.
    #.
    print(A)
    sum1 = sum(A)
    x=0
    list1=[]
    for i in A:
        list1.append(x + i/sum1)
        x = x + i/sum1;
    #list1 contsins cumulative sum
    bit = uniform(0,1)
    for i in range (0, len(list1)):
        if bit < list1[i]:
            return A[i]

def sampling_based_on_magnitued():
    for i in range(1,100):
        number = pick_a_number_from_list(A)
        print(number)
sampling_based_on_magnitued()
```

```
[0, 5, 27, 6, 13, 28, 100, 45, 10, 79]
45
[0, 5, 27, 6, 13, 28, 100, 45, 10, 79]
100
[0, 5, 27, 6, 13, 28, 100, 45, 10, 79]
28
[0, 5, 27, 6, 13, 28, 100, 45, 10, 79]
27
[0, 5, 27, 6, 13, 28, 100, 45, 10, 79]
79
[0, 5, 27, 6, 13, 28, 100, 45, 10, 79]
79
[0, 5, 27, 6, 13, 28, 100, 45, 10, 79]
100
[0, 5, 27, 6, 13, 28, 100, 45, 10, 79]
28
[0, 5, 27, 6, 13, 28, 100, 45, 10, 79]
100
[0, 5, 27, 6, 13, 28, 100, 45, 10, 79]
79
[0, 5, 27, 6, 13, 28, 100, 45, 10, 79]
79
[0, 5, 27, 6, 13, 28, 100, 45, 10, 79]
27
[0, 5, 27, 6, 13, 28, 100, 45, 10, 79]
45
[0, 5, 27, 6, 13, 28, 100, 45, 10, 79]
45
[0, 5, 27, 6, 13, 28, 100, 45, 10, 79]
100
[0, 5, 27, 6, 13, 28, 100, 45, 10, 79]
100
[0, 5, 27, 6, 13, 28, 100, 45, 10, 79]
100
[0, 5, 27, 6, 13, 28, 100, 45, 10, 79]
79
[0, 5, 27, 6, 13, 28, 100, 45, 10, 79]
27
[0, 5, 27, 6, 13, 28, 100, 45, 10, 79]
79
[0, 5, 27, 6, 13, 28, 100, 45, 10, 79]
13
[0, 5, 27, 6, 13, 28, 100, 45, 10, 79]
79
[0, 5, 27, 6, 13, 28, 100, 45, 10, 79]
100
[0, 5, 27, 6, 13, 28, 100, 45, 10, 79]
10
[0, 5, 27, 6, 13, 28, 100, 45, 10, 79]
79
[0, 5, 27, 6, 13, 28, 100, 45, 10, 79]
45
[0, 5, 27, 6, 13, 28, 100, 45, 10, 79]
45
[0, 5, 27, 6, 13, 28, 100, 45, 10, 79]
79
[0, 5, 27, 6, 13, 28, 100, 45, 10, 79]
79
[0, 5, 27, 6, 13, 28, 100, 45, 10, 79]
79
[0, 5, 27, 6, 13, 28, 100, 45, 10, 79]
```

```
79
[0, 5, 27, 6, 13, 28, 100, 45, 10, 79]
79
[0, 5, 27, 6, 13, 28, 100, 45, 10, 79]
100
[0, 5, 27, 6, 13, 28, 100, 45, 10, 79]
27
[0, 5, 27, 6, 13, 28, 100, 45, 10, 79]
100
[0, 5, 27, 6, 13, 28, 100, 45, 10, 79]
79
[0, 5, 27, 6, 13, 28, 100, 45, 10, 79]
100
[0, 5, 27, 6, 13, 28, 100, 45, 10, 79]
100
[0, 5, 27, 6, 13, 28, 100, 45, 10, 79]
79
[0, 5, 27, 6, 13, 28, 100, 45, 10, 79]
100
[0, 5, 27, 6, 13, 28, 100, 45, 10, 79]
27
[0, 5, 27, 6, 13, 28, 100, 45, 10, 79]
79
[0, 5, 27, 6, 13, 28, 100, 45, 10, 79]
100
[0, 5, 27, 6, 13, 28, 100, 45, 10, 79]
100
[0, 5, 27, 6, 13, 28, 100, 45, 10, 79]
28
[0, 5, 27, 6, 13, 28, 100, 45, 10, 79]
79
[0, 5, 27, 6, 13, 28, 100, 45, 10, 79]
10
[0, 5, 27, 6, 13, 28, 100, 45, 10, 79]
100
[0, 5, 27, 6, 13, 28, 100, 45, 10, 79]
79
[0, 5, 27, 6, 13, 28, 100, 45, 10, 79]
28
[0, 5, 27, 6, 13, 28, 100, 45, 10, 79]
100
[0, 5, 27, 6, 13, 28, 100, 45, 10, 79]
79
[0, 5, 27, 6, 13, 28, 100, 45, 10, 79]
28
[0, 5, 27, 6, 13, 28, 100, 45, 10, 79]
28
[0, 5, 27, 6, 13, 28, 100, 45, 10, 79]
100
[0, 5, 27, 6, 13, 28, 100, 45, 10, 79]
100
[0, 5, 27, 6, 13, 28, 100, 45, 10, 79]
5
[0, 5, 27, 6, 13, 28, 100, 45, 10, 79]
45
[0, 5, 27, 6, 13, 28, 100, 45, 10, 79]
28
[0, 5, 27, 6, 13, 28, 100, 45, 10, 79]
45
[0, 5, 27, 6, 13, 28, 100, 45, 10, 79]
28
```

```
[0, 5, 27, 6, 13, 28, 100, 45, 10, 79]
100
[0, 5, 27, 6, 13, 28, 100, 45, 10, 79]
100
[0, 5, 27, 6, 13, 28, 100, 45, 10, 79]
45
[0, 5, 27, 6, 13, 28, 100, 45, 10, 79]
27
[0, 5, 27, 6, 13, 28, 100, 45, 10, 79]
79
[0, 5, 27, 6, 13, 28, 100, 45, 10, 79]
100
[0, 5, 27, 6, 13, 28, 100, 45, 10, 79]
79
[0, 5, 27, 6, 13, 28, 100, 45, 10, 79]
28
[0, 5, 27, 6, 13, 28, 100, 45, 10, 79]
79
[0, 5, 27, 6, 13, 28, 100, 45, 10, 79]
100
[0, 5, 27, 6, 13, 28, 100, 45, 10, 79]
79
[0, 5, 27, 6, 13, 28, 100, 45, 10, 79]
13
[0, 5, 27, 6, 13, 28, 100, 45, 10, 79]
79
[0, 5, 27, 6, 13, 28, 100, 45, 10, 79]
79
[0, 5, 27, 6, 13, 28, 100, 45, 10, 79]
100
[0, 5, 27, 6, 13, 28, 100, 45, 10, 79]
100
[0, 5, 27, 6, 13, 28, 100, 45, 10, 79]
100
[0, 5, 27, 6, 13, 28, 100, 45, 10, 79]
79
[0, 5, 27, 6, 13, 28, 100, 45, 10, 79]
5
[0, 5, 27, 6, 13, 28, 100, 45, 10, 79]
100
[0, 5, 27, 6, 13, 28, 100, 45, 10, 79]
79
[0, 5, 27, 6, 13, 28, 100, 45, 10, 79]
79
[0, 5, 27, 6, 13, 28, 100, 45, 10, 79]
5
[0, 5, 27, 6, 13, 28, 100, 45, 10, 79]
100
[0, 5, 27, 6, 13, 28, 100, 45, 10, 79]
45
[0, 5, 27, 6, 13, 28, 100, 45, 10, 79]
45
[0, 5, 27, 6, 13, 28, 100, 45, 10, 79]
100
[0, 5, 27, 6, 13, 28, 100, 45, 10, 79]
6
[0, 5, 27, 6, 13, 28, 100, 45, 10, 79]
100
[0, 5, 27, 6, 13, 28, 100, 45, 10, 79]
10
[0, 5, 27, 6, 13, 28, 100, 45, 10, 79]
```

```
79
[0, 5, 27, 6, 13, 28, 100, 45, 10, 79]
6
[0, 5, 27, 6, 13, 28, 100, 45, 10, 79]
79
[0, 5, 27, 6, 13, 28, 100, 45, 10, 79]
45
[0, 5, 27, 6, 13, 28, 100, 45, 10, 79]
100
[0, 5, 27, 6, 13, 28, 100, 45, 10, 79]
100
[0, 5, 27, 6, 13, 28, 100, 45, 10, 79]
100
[0, 5, 27, 6, 13, 28, 100, 45, 10, 79]
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: ####
```

In [17]:

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

# you can free to change all these codes/structure
# String: it will be the input to your program
def replace_digits(String):
    # write your code
    return '#' * len(re.sub('[^0-9]','', String))
String = input()
print(replace_digits(String))
```

```
a2b3c4
###
```

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','stu
dent7','student8','student9','student10']
Marks = [45, 78, 12, 14, 48, 43, 47, 98, 35, 80]
a.
student8  98
student10 80
student2  78
student5  48
student7  47
b.
student3 12
student4 14
student9 35
student6 43
student1 45
c.
student9 35
student6 43
student1 45
student7 47
student5 48
```

In [3]:

```
import operator
import math
import functools

# you can free to change all these codes/structure
def display_dash_board(students, marks):
  Total_marks_list = dict(zip(students, marks))
  Top_5_students = dict(sorted(Total_marks_list.items(), key = operator.itemgetter(1),
reverse = True)[ :5])
  Least_5_students = dict( sorted(Total_marks_list.items(),key = operator.itemgetter(1
), reverse = False)[ :5])

  print("a.")
  for key,val in Top_5_students.items():
    print(key,val)
  print("-------------------------")
  print("b.")
  for key,val in Least_5_students.items():
    print(key,val)
  print("-------------------------")
  print("c.")
def students_within_25_and_75(marks,percent):
  if not marks:
    return None
  per_cal = len(marks)*percent
  flo = math.floor(per_cal)
  cei = math.ceil(per_cal)
  if flo == cei:
    return marks[int(per_cal)]
  a = marks[int(cei)+1]
  b = marks[int(cei)]
  return (a + b)/2

students=['student1','student2','student3','student4','student5','student6','student7',
'student8','student9','student10']
marks = [45, 78, 12, 14, 48, 43, 47, 98, 35, 80]
A = display_dash_board(students, marks)
B= students_within_25_and_75(marks, 0.25)
C= students_within_25_and_75(marks, 0.75)
Total_marks_list = dict(zip(students, marks))
dictionary = dict(sorted(Total_marks_list.items(),key = operator.itemgetter(1), reverse
= False))
for k,v in dictionary.items():
  if v > B and v < C:
    print(k,v)
```

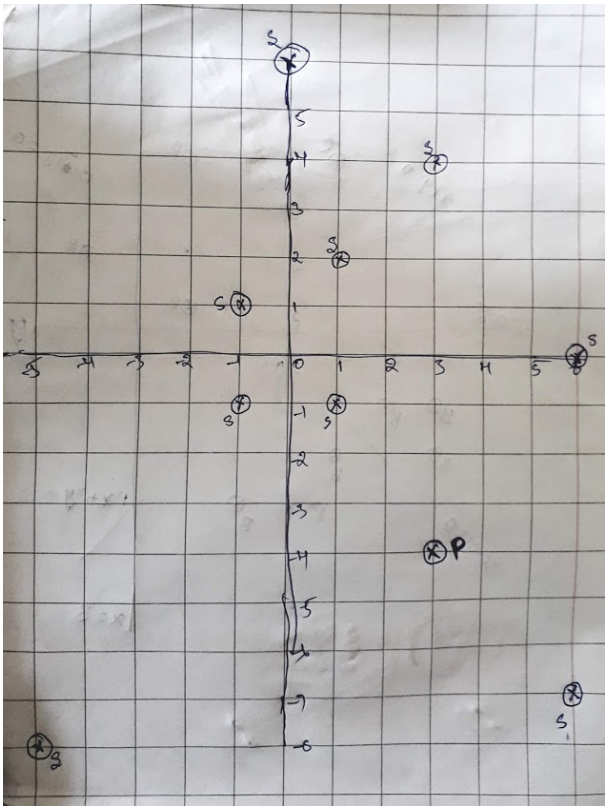
```
a.
student8 98
student10 80
student2 78
student5 48
student7 47
-------------------------
b.
student3 12
student4 14
student9 35
student6 43
student1 45
-------------------------
c.
student9 35
student6 43
student1 45
student7 47
student5 48
```

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

Ex:

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



Output:
(6,-7)
(1,-1)
(6,0)
(-5,-8)
(-1,-1)

In [20]:

```
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 exampl
es
# 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 = sorted(S, key=lambda p: math.acos((p[0]*P[0] + p[1]*P[1])/(ma
th.sqrt(p[0]**2 + p[1]**2) * math.sqrt(P[0]**2 + P[1]**2))))[:5]
    return closest_points_to_p  # its list of tuples

S= [(1,2),(3,4),(-1,1),(6,-7),(0, 6),(-5,-8),(-1,-1),(6,0),(1,-1)]
P= (3,-4)
points = closest_points_to_p(S, P)
print(points) #print the returned values
```

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

Q6: Find Which line separates oranges and apples

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

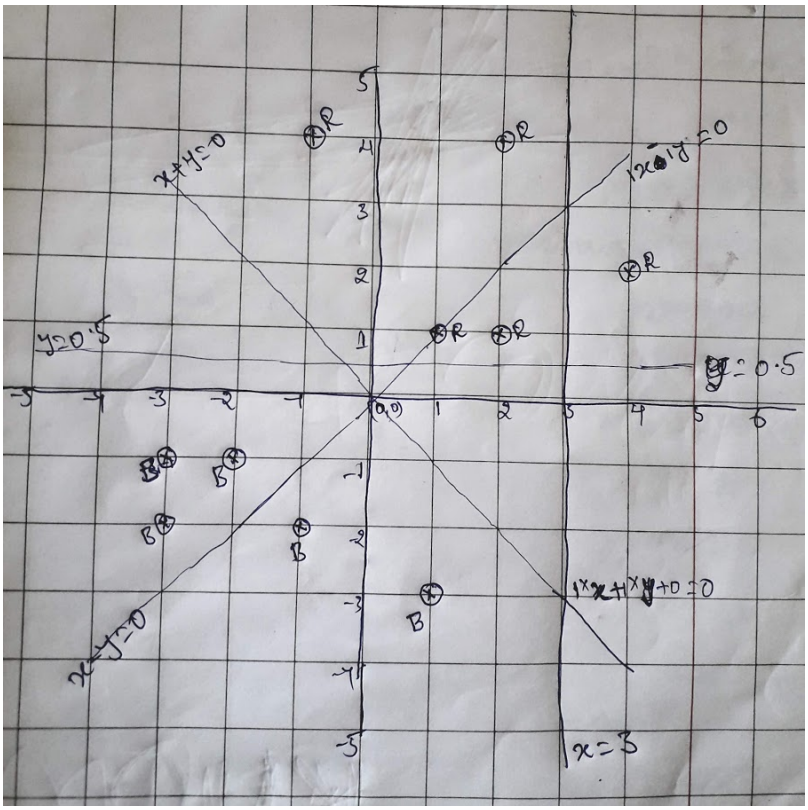
```
Red =[(R11,R12),(R21,R22),(R31,R32),(R41,R42),(R51,R52),..,(Rn1,Rn2)]
Blue=[(B11,B12),(B21,B22),(B31,B32),(B41,B42),(B51,B52),..,(Bm1,Bm2)]
```

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 interc
ept
```

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 [21]:

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

# you can free to change all these codes/structure
def i_am_the_one(red,blue,line):
    # your code
    red = list(map(lambda p: eval(line.replace('x', '*' + str(p[0])).replace('y', '*' +
str(p[1]))), red))
    blue = list(map(lambda p: eval(line.replace('x', '*' + str(p[0])).replace('y', '*'
+ str(p[1]))), blue))

    return "YES" if ((all(list(map(lambda x: x > 0, red))) and all(list(map(lambda x: x
< 0, blue))))
                     or
                      (all(list(map(lambda x: x < 0, red))) and all(list(map(lambda x: x
> 0, blue))))) else "NO"

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

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

```
YES
NO
NO
YES
```

Q7: Filling the missing values in the specified 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 equ
ally to all 4 places

Ex 2: 40, _, _, _, 60 ==> (60+40)/5,(60+40)/5,(60+40)/5,(60+40)/5,(60+40)/5 ==>
 20, 20, 20, 20, 20 i.e. the sum of (60+40) is distributed qually to all 5 place
s

Ex 3: 80, _, _, _, _  ==> 80/5,80/5,80/5,80/5,80/5 ==> 16, 16, 16, 16, 16 i.e. t
he 80 is distributed qually to all 5 missing values that are right to it

Ex 4: _, _, 30, _, _, _, 50, _, _
==> we will fill the missing values from left to right
    a. first we will distribute the 30 to left two missing values (10, 10, 10,
 _, _, _, 50, _, _)
    b. now distribute the sum (10+50) missing values in between (10, 10, 12, 12,
12, 12, 12, _, _)
    c. now we will distribute 12 to right side missing values (10, 10, 12, 12, 1
2, 12, 4, 4, 4)
```

for a given string with comma seprate values, which will have both missing values numbers like ex: "_, _, x, _, _, _" you need fill the missing values Q: your program reads a string like ex: "_, _, x, _, _, _" and returns the filled sequence Ex:

```
Input1: "_,_,_,24"
Output1: 6,6,6,6

Input2: "40,_,_,_,60"
Output2: 20,20,20,20,20

Input3: "80,_,_,_,_"
Output3: 16,16,16,16,16

Input4: "_,_,30,_,_,_,50,_,_"
Output4: 10,10,12,12,12,12,4,4,4
```

In [22]:

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


# you can free to change all these codes/structure
def curve_smoothing(string):
    lst = string.split(',')
    for i in range(len(lst)):
        if lst[i].isdigit():
            for j in range(i+1):
                lst[j] = int(lst[i])//(i+1)
            new_index = i
            new_value = int(lst[i])
            break
    for i in range(new_index + 1, len(lst)):
        if lst[i].isdigit():
            temp=(new_value+int(lst[i]))//(i-new_index+1)
            for j in range(new_index,i+1):
                lst[j]=temp
            new_index=i
            new_value=int(lst[i])
    try:
        for i in range(new_index+1,len(lst)):
            if not(lst[i].isdigit()):
                count=lst.count('_')
                break
        temp1=new_value//(count+1)
        for i in range(new_index,len(lst)):
            lst[i]=temp1
    except:
        pass
    return lst

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,S
1]]

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

In [23]:

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



# you can free to change all these codes/structure
# write your python code here
# you can take the above example as sample input for your program to test
# it should work for any general input try not to hard code for only given input string
s

# you can free to change all these codes/structure
def compute_conditional_probabilites(A):
    # your code
    flist = sorted(set(map(lambda x: x[0], A)))
    slist = sorted(set(map(lambda x: x[1], A)))
    # print the output as per the instructions
    l = [
     [
        "P(F={0}|S=={1})={2}/{3}".format(f, s, len(list(filter(lambda a: (a[0] == f) an
d (a[1] == s), A))), len(list(filter(lambda a: (a[1] == s), A))))
            for s in slist
        ]
        for f in flist
    ]
    for t in l:
        string = ""
        for s in t:
            string += s + ", "
        print(string.strip(", "))

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)
```

```
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
```

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 [24]:

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

# you can free to change all these codes/structure
def string_features(S1, S2):
    # your code
    s1 = set(S1.split(" "))
    s2 = set(S2.split(" "))
    a = len(s1.intersection(s2))
    b = sorted(s1 - s2, key=lambda x: S1.index(x))
    c = sorted(s2 - s1, key=lambda x: S2.index(x))
    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)
print(b)
print(c)
```

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

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
$f(Y, Y_{score}) = -1 * \frac{1}{n}\Sigma_{foreach Y, Y_{score} pair}(Ylog10(Y_{score}) + (1-Y)log10(1-Y_{score}))$ 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:
0.4243099
```

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

In [25]:

```
# write your python code here
# you can take the above example as sample input for your program to test
# it should work for any general input try not to hard code for only given input string
s
from math import log10

# you can free to change all these codes/structure
def compute_log_loss(A):
    # your code
    loss = (-1/len(A)) * sum(list(map(lambda a: (a[0] * log10(a[1])) + ((1 - a[0]) * lo
g10(1 - a[1])), A)))
    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
```

In [4]:

```
!jupyter nbconvert --to html Python_Assignment_1.ipynb
```

```
[NbConvertApp] Converting notebook Python_Assignment_1.ipynb to html
[NbConvertApp] Writing 350520 bytes to Python_Assignment_1.html
```

In [ ]: