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 [456]:
```
# 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 inp
ut examples


# you can free to change all these codes/structure
# here A and B are list of lists


def matrix_mul(A,B):
    C = [[0]*len(B[0]) for x in range(len(A))]
    if len(A) == len(B):
        for i in range(len(A)):
            for j in range(len(B[0])):
                for k in range(len(B)):
                    C[i][j] += A[i][k] * B[k][j]
        for y in C:
            return (C)
    else:
         return "Not possible"

matrix_mul( [[1,2],[3,4]],[[1,4],[4,5],[5,6],[8,9]])
```

Out[456]: '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 [158]:
```
from random import uniform

def pick_a_number_from_list(A):
    A.sort()
    sum_a = sum(A)
    weights = []
    x = 0
    for i in A:
        weights.append(x+i/sum_a)
        x = x+i/sum_a
    p = uniform(0,1)
    for i in range(len(weights)):
        if p < weights[i]:
            return A[i]


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


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

```
100 45 45 79 79 27 100 28 28 6 28 45 45 28 45 100 45 79 100 45 45 45 5 79 27
28 79 45 28 79 100 79 79 100 100 27 100 6 100 100 27 100 45 6 28 79 79 100 10
0 13 79 100 6 100 10 13 100 27 100 100 27 27 100 27 79 79 79 27 100 13 13 79
100 28 79 100 100 79 13 79 79 100 28 27 100 45 100 100 79 100 28 27 100 79 10
0 5 100 5 100
```

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 [160]:
```
import re

def replace_digits(String):
    new_string  = re.sub("[0-9]", '#' ,re.sub("[^0-9]","",String))
    return (new_string)

replace_digits('#2a$#b%c%561#')
```

Out[160]: '####'

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 [203]:
```
Students = ['student1','student2','student3','student4','student5','student6',
'student7','student8','student9','student10']
Marks = [45, 78, 12, 14, 48, 43, 45, 98, 35, 80]


def display_dash_board(students, marks):

    my_dict = dict(zip(Students,Marks))

    top_5_students   = sorted(my_dict.items(),key = lambda x : x[1],reverse=Tr
ue)[:5]

    least_5_students = sorted(my_dict.items(),key = lambda x : x[1])[:5]

    sorted_dict = sorted(my_dict.items(), key=lambda x: x[1])
    n = len(my_dict)

    percentile_25 = int(0.25*n)
    percentile_75 = int(0.75*n)

    students_within_25_and_75 = sorted_dict[percentile_25:percentile_75]

    return top_5_students, least_5_students,students_within_25_and_75

top_5_students, least_5_students,students_within_25_and_75 = display_dash_boar
d(Students,Marks)

print ("Top 5 Students are : {0} \n\nLeast 5 Students are : {1} \n\nStudents W
ithin 25 and 75 percentile are : {2} ".format(top_5_students,least_5_students,
students_within_25_and_75))
```

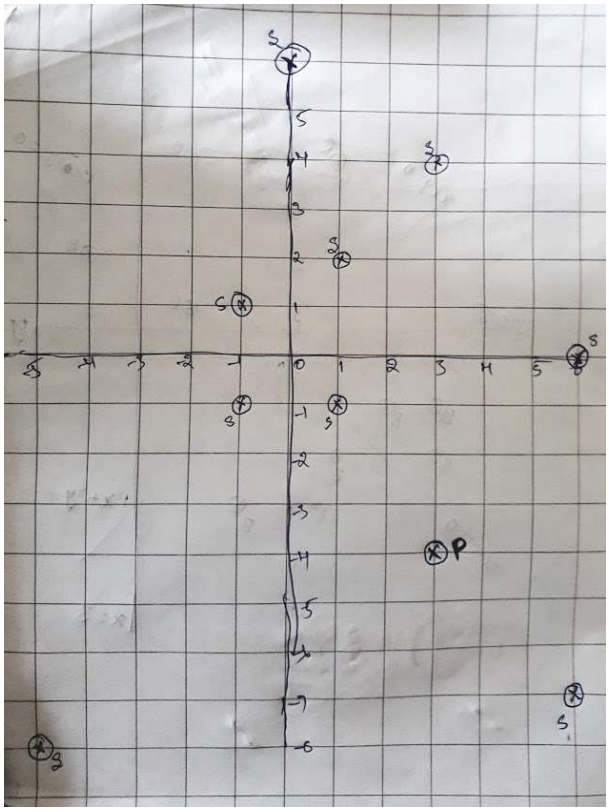
```
Top 5 Students are : [('student8', 98), ('student10', 80), ('student2', 78),
('student5', 48), ('student1', 45)]

Least 5 Students are : [('student3', 12), ('student4', 14), ('student9', 35),
('student6', 43), ('student1', 45)]

Students Within 25 and 75 percentile are : [('student9', 35), ('student6', 4
3), ('student1', 45), ('student7', 45), ('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 [236]:
```
import math

def closest_points_to_p(S, P):
    cosine_dist = []
    for point in S:
        numerator = point[0]*P[0] + point[1]*P[1]
        denominator = math.sqrt((point[0]**2 + point[1]**2)*(P[0]**2+P[1]**2))
        dist = math.acos(numerator/denominator)
        cosine_dist.append(dist)

    my_dict1 = dict(zip(S,cosine_dist))
    points_closet = sorted(my_dict1.items(),key = lambda x : x[1])[:5]

    for item in points_closet:
        print (item[0])

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

closest_points_to_p(S, P)
```

```
(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

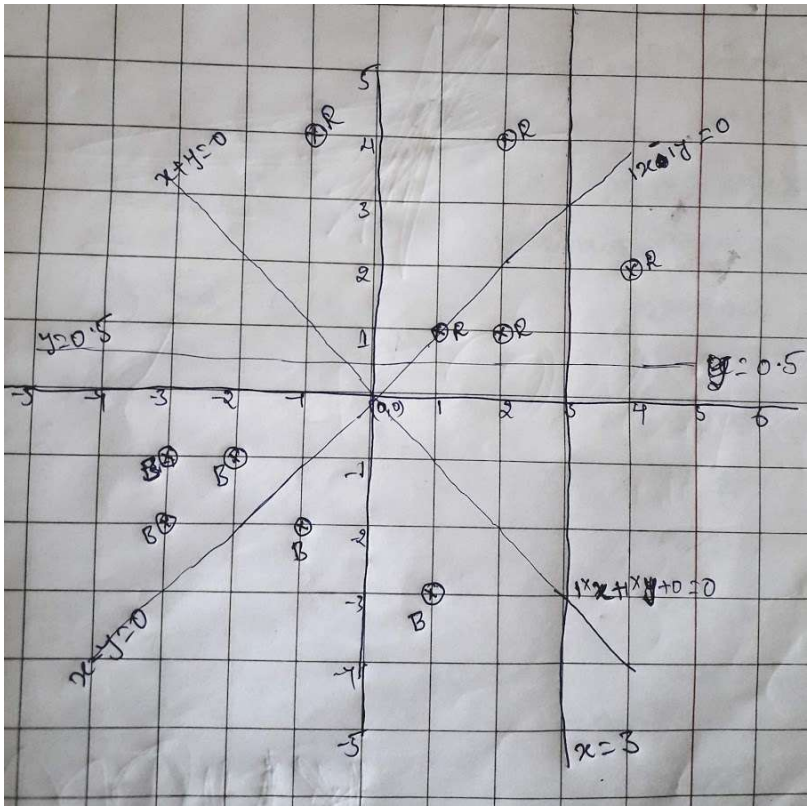
```
Red =[(R11,R12),(R21,R22),(R31,R32),(R41,R42),(R51,R52),..,(Rn1,Rn2)]
Blue=[(B11,B12),(B21,B22),(B31,B32),(B41,B42),(B51,B52),..,(Bm1,Bm2)]
```

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 [457]:
```
## I've approached this question using minimum distance of a point P(x1,y1) fr
om line (Ax + By + C =0) formula which is :
## d(x1,y1) = |Ax1 + By1 + c|/sqrt(A*A + B*B) . However, I've not taken the ab
solute value of Numerator as sign of the equation
## was required to determine the sides of the point. Denominator could have be
en ignored but I've considered it for the sake of
## the formula


import math

def i_am_the_one(red,blue,line):
    count_r = 0
    count_b = 0

#   C = re.findall(r'[\d\.\-\+]+', line)
    line = line.replace('x'," ").replace('y'," ")
    C = line.split(" ")
    for i in range(len(C)):
        C[i] = float(C[i])
    d = math.sqrt((C[0]*C[0]) + (C[1]*C[1]))

    for j in red:
        n = C[0]*j[0] + C[1]*j[1] + C[2]
        dist = n/d
        if dist > 0:
            count_r += 1
        else:
            count_r -= 1

    for k in blue:
        n1 =  C[0]*k[0] + C[1]*k[1] + C[2]
        dist1 = n1/d
        if dist1 > 0:
            count_b += 1
        else:
            count_b -= 1

    if abs(count_b) == count_r:
        return ("YES")
    else:
        return  ("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)
```

```
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 equall
y 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, 1
2, 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 [460]:
```
def curve_smoothing(string):
    C = string
    C = C.replace("_","0")
    C = C.split(",")
    for i in range(len(C)):
        C[i] = int(C[i])

    count = 0
    mid  = 0

    for i in range(len(C)):
        if C[i] == 0:
            count += 1
        else:
            for j in range(0,i+1):
                C[j] = C[i]/(count+1)
            mid = i
            mid_value = C[mid]
            break
    dnm = 1
    check = 0
    for k in range(mid+1,len(C)):
      # print (k)
        if C[k] != 0:
            dnm = (k - mid +1)
            check = k
            break
    check_value = C[check]
    for l in range(mid,check+1):
        C[l] = (mid_value + check_value)/dnm


    last_value = C[check]
    for m in range(check,len(C)):
        C[m] = last_value/(len(C) - check)


    # print(last_value)
    # print (check_value)

    return (C)

S=  "_,_,30,_,_,_,50,_,_"

smoothed_values= curve_smoothing(S)
print (smoothed_values)
```

```
[10.0, 10.0, 12.0, 12.0, 12.0, 12.0, 4.0, 4.0, 4.0]
```

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)=1/2, P(F=F1|S==S2)=1/2, P(F=F1|S==S3)=0/2
b. P(F=F2|S==S1)=1/3, P(F=F2|S==S2)=1/3, P(F=F2|S==S3)=1/3
c. P(F=F3|S==S1)=0/3, P(F=F3|S==S2)=1/2, P(F=F3|S==S3)=1/2
d. P(F=F4|S==S1)=1/2, P(F=F4|S==S2)=0/2, P(F=F4|S==S3)=1/2
e. P(F=F5|S==S1)=1/1, P(F=F5|S==S2)=0/1, P(F=F5|S==S3)=0/1
```

In [416]:
```
def compute_conditional_probabilites(A,F,S):

    count_S = 0
    for item in A:
        if item[1] == S:
            count_S += 1

    count_FS = 0

    for item in A:
        if item[1] == S:
            if item[0] == F:
                count_FS += 1

    P = round(count_FS/count_S,2)

    print ("P(F={0}|S=={1}) = {2}".format(F,S,P))


L = [['F1','S1'],['F2','S2'],['F3','S3'],['F1','S2'],['F2','S3'],['F3','S2'],['F2','S1'],['F4','S1'],['F4','S3'],['F5','S1']]

compute_conditional_probabilites(L,'F1','S1')
compute_conditional_probabilites(L,'F1','S2')
compute_conditional_probabilites(L,'F1','S3')
compute_conditional_probabilites(L,'F2','S1')
compute_conditional_probabilites(L,'F2','S2')
compute_conditional_probabilites(L,'F2','S3')
compute_conditional_probabilites(L,'F3','S1')
compute_conditional_probabilites(L,'F3','S2')
compute_conditional_probabilites(L,'F3','S3')
compute_conditional_probabilites(L,'F4','S1')
compute_conditional_probabilites(L,'F4','S2')
compute_conditional_probabilites(L,'F4','S3')
compute_conditional_probabilites(L,'F5','S1')
compute_conditional_probabilites(L,'F5','S2')
compute_conditional_probabilites(L,'F5','S3')
```

```
P(F=F1|S==S1) = 0.25
P(F=F1|S==S2) = 0.33
P(F=F1|S==S3) = 0.0
P(F=F2|S==S1) = 0.25
P(F=F2|S==S2) = 0.33
P(F=F2|S==S3) = 0.33
P(F=F3|S==S1) = 0.0
P(F=F3|S==S2) = 0.33
P(F=F3|S==S3) = 0.33
P(F=F4|S==S1) = 0.25
P(F=F4|S==S2) = 0.0
P(F=F4|S==S3) = 0.33
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']

In [443]:
```
def string_features(S1, S2):

    S1_split = S1.split(" ")
    S2_split = S2.split(" ")

    common_list = []

    for i in S1_split:
        for j in S2_split:
            if i == j:
                common_list.append(i)
            else:
                continue
    A = len(common_list)
    B = []
    for i in S1_split:
        if i not in S2_split:
            B.append(i)

    C = []
    for i in S2_split:
        if i not in S1_split:
            C.append(i)

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

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

In [455]:
```
from math import log

def compute_log_loss(A):

    n = len(A)

    summation = 0
    for item in A:
        summation += (item[0]*log(item[1],10) + (1-item[0])*log(1-item[1],10))

    loss = -summation/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)
```

```
0.42430993457031635
```