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 [8]:
```
# 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


# you can free to change all these codes/structure
# here A and B are list of lists
def matrix_mul(A, B):
    a_m,a_n=len(A),len(A[0])
    b_m,b_n=len(B),len(B[0])
    print(a_m,a_n)
    print(b_m,b_n)

    if a_n!=b_m:
        return "Not possible"
    else:
        result=[[0 for i in range(b_n)] for j in range(a_m)]
        print(result)
        for i in range(a_m):
            for j in range(b_n):
                for k in range(b_m):
                    result[i][j] += A[i][k] * B[k][j]

        for r in result:
            print(r)
```

In [9]:
```
A   = [[1,2],[3,4]]
B   = [[1,2,3,4,5],[5,6,7,8,9]]
```

In [10]:
```
matrix_mul(A,B)
```

```
2 2
2 5
[[0, 0, 0, 0, 0], [0, 0, 0, 0, 0]]
[11, 14, 17, 20, 23]
[23, 30, 37, 44, 51]
```

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 experiment
s.
f(100) > f(79) > f(45) > f(28) > f(27) > f(13) > f(10) > f(6) > f(5) > f
(0)
```

In [94]:
```
from random import uniform
from collections import Counter
A = [0,5,27,6,13,28,100,45,10,79]
def pick_a_number_from_list(A):
    sum_a=0

    for i in range(len(A)):
        sum_a = sum_a + A[i]


    weight_pro=[]

    j=0
    while j < len(A):
        weight_pro.append(A[j]/sum_a)
        j=j+1


    Cul_weigh=[sum(weight_pro[0:x+1]) for x in range(0,len(weight_pro))]

    i = random.uniform(0.0,1.0)

    lo=0
    hi=len(Cul_weigh)
    while lo < hi:
        mid = (lo+hi)//2
        if i*Cul_weigh[-1] <= Cul_weigh[mid]: hi = mid
        else: lo = mid+1
    return lo

def sampling_based_on_magnitued():
    output=[]
    for i in range(0,100):
        number = pick_a_number_from_list(A)
        output.append(A[number])
    print(Counter(output))
```

In [97]:
```
sampling_based_on_magnitued()
```

```
Counter({100: 29, 79: 26, 45: 17, 28: 11, 27: 7, 6: 4, 5: 3, 10: 2, 13: 1})
```

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 [3]:
```
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

# you can free to change all these codes/structure
# String: it will be the input to your program
def replace_digits(String):
    rep=''.join("#" if c.isdigit() else '' for c in String)

    return rep # modified string which is after replacing the # with digits

String='abc'
replace_digits(String)
```

Out[3]: ''

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','student
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','studen
t6','student7','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 [16]:
```
# 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

# you can free to change all these codes/structure
def display_dash_board(students, marks):
    # write code for computing top top 5 students
    mark_sort=sorted(range(len(marks)), key=lambda k: marks[k])

    least_5_students=[[students[i],marks[i]] for i in mark_sort[:5]]
    # write code for computing top least 5 students
    top_5_students=[[students[i],marks[i]] for i in mark_sort[:-6:-1]]

    # write code for computing top least 5 students
    n = len(marks)
    first_quartile = int(n/4) if (n/4).is_integer() else int(n/4) + 1  # Refernc
    third_quartile = int(3*n/4)
    c=sorted(marks)
    c=c[first_quartile-1:third_quartile]
    students_within_25_and_75 = [[students[marks.index(i)],i] for i in c]

    return top_5_students, least_5_students, students_within_25_and_75

students=['student1','student2','student3','student4','student5','student6','stu
marks = [45, 78, 12, 14, 48, 43, 47, 98, 35, 80]
top_5_students, least_5_students, students_within_25_and_75 = display_dash_board
print(top_5_students,'\n', least_5_students,'\n' ,students_within_25_and_75)
```

```
[['student8', 98], ['student10', 80], ['student2', 78], ['student5', 48], ['stu
dent7', 47]]
 [['student3', 12], ['student4', 14], ['student9', 35], ['student6', 43], ['stu
dent1', 45]]
 [['student9', 35], ['student6', 43], ['student1', 45], ['student7', 47], ['stu
dent5', 48]]
```

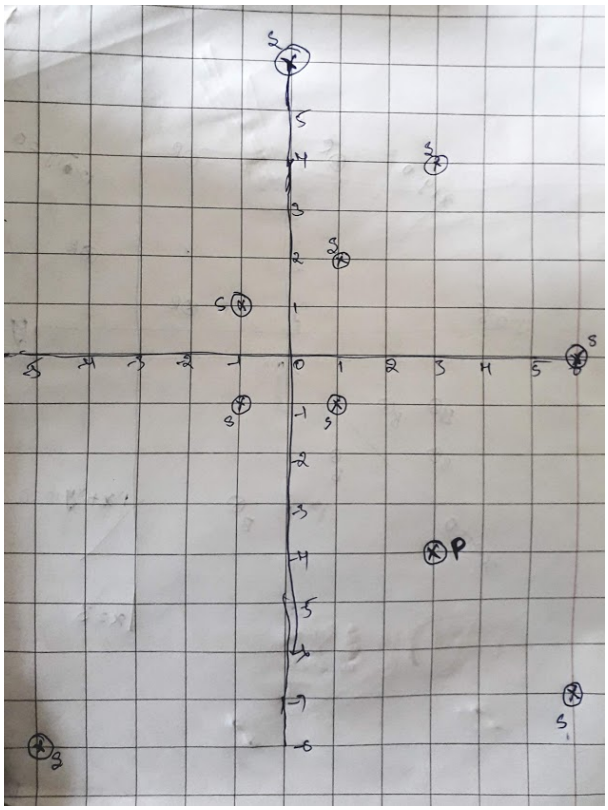
Q5: Find the closest points

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

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 [137]:
```
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
# 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):
    S.sort(key=lambda x: math.acos(((x[0] * P[0]) + (x[1] * P[1])) / ((math.sqrt
    closest_points_to_p=S[:5]
    return closest_points_to_p  # its list of tuples
```

In [139]:
```
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)
```

Out[139]:
```
[(6, -7), (1, -1), (6, 0), (-5, -8), (-1, -1)]
```

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.
```

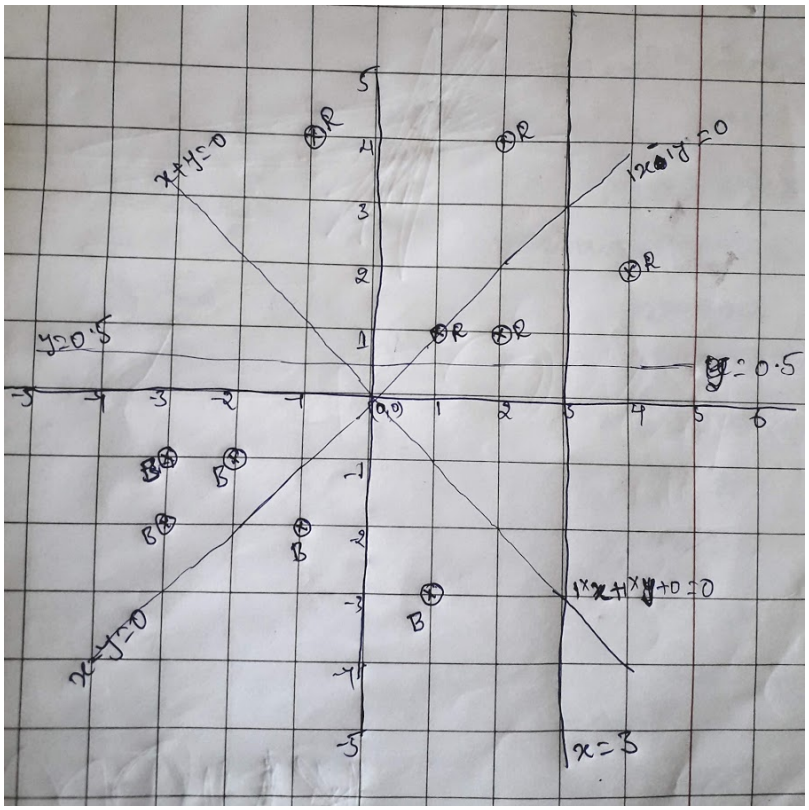
Your task here is to print "YES"/"NO" for each line given. You should print YES, if all the red points are one side of the line and blue points are on other side of the line, otherwise you should print NO.

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"]



Output:

YES

NO

NO

YES

In [6]:
```
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


# you can free to change all these codes/structure
def i_am_the_one(red,blue,Lines):
    for i in Lines:
        #extracting the co-efficcient
        eqn= re.findall(r'[\d\.\-\+]+', i)
        check=[]
        #String to numeric format
        for i in range(len(eqn)):
            eqn[i]=float(eqn[i])

        a=eqn[0];b=eqn[1];c=eqn[2]

        for i in range(0,len(Red)):
            x1=Red[i][0];y1=Red[i][1]
            x2=Blue[i][0];y2=Blue[i][1]

            fx1 = a * x1 + b * y1 - c
            fx2 = a * x2 + b * y2 - c
#         if they lie on same side then value will be positive else negative
            if ((fx1 * fx2) > 0):
                check.append('No')
            else:
                check.append('Yes')
        if "No" in check:
            print('No')
        else:
            print('Yes')

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

i_am_the_one(Red, Blue,Lines)
```

```
Yes
No
No
Yes
```

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 th
e 24 equally to all 4 places

Ex 2: 40, _, _, _, 60 ==> (60+40)/5,(60+40)/5,(60+40)/5,(60+40)/5,(60+4
0)/5 ==> 20, 20, 20, 20, 20 i.e. the sum of (60+40) is distributed quall
y to all 5 places

Ex 3: 80, _, _, _, _  ==> 80/5,80/5,80/5,80/5,80/5 ==> 16, 16, 16, 16, 1
6 i.e. the 80 is distributed qually to all 5 missing values that are rig
ht 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, 1
0, 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, 1
2, 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 [10]:
```
# 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


# you can free to change all these codes/structure


def filling(x, a, b): #Reference Stackoverflow
    if a == -1:
        v = float(x[b])/(b+1)
        for i in range(a+1,b+1):
            x[i] = v
    elif b == -1:
        v = float(x[a])/(len(x)-a)
        for i in range(a, len(x)):
            x[i] = v
    else:
        v = (float(x[a])+float(x[b]))/(b-a+1)
        for i in range(a,b+1):
            x[i] = v
    return x



def curve_smoothing(string):
    x = string.replace(" ","").split(",")
    y = [i for i, v in enumerate(x) if v != '_']
    if y[0] != 0:
        y = [-1] + y
    if y[-1] != len(x)-1:
        y = y + [-1]
    for (a, b) in zip(y[:-1], y[1:]):
        filling(x,a,b)
    return x

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: 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 [73]:
```
firstcol_second_col= []
second_col= []
def compute_conditional_probabilites(A): #Reference mathacademy
    for i in range(len(A)):
        k= A[i][0]+A[i][1]
        second_col.append(A[i][1])
        firstcol_second_col.append(k)
A= [['F1','S1'],['F2','S2'],['F3','S3'],['F1','S2'],['F2','S3'],['F3','S2'],['F2
compute_conditional_probabilites(A)
def conditional_prob(F,S):
    print("conditional probability P(F={0}|S=={1})=".format(F,S),((firstcol_secor


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

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

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 [8]:
```
count=0
distinct_1=[]
distinct_2=[]
S1= "the first column F will contain only 5 unique values"
S2= "the second column S will contain only 3 unique values"
for i in S1.split(' '):
    if i in S2:
        count+=1
    else:
        distinct_1.append(i)
for i in S2.split(' '):
    if i not in S1:
        distinct_2.append(i)

print(count)
print(distinct_1)
print(distinct_2)
```

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

In [9]:
```
# 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

# you can free to change all these codes/structure
def string_features(S1, S2):
    count=0
    distinct_1=[]
    distinct_2=[]
    S1= "the first column F will contain only 5 unique values"
    S2= "the second column S will contain only 3 unique values"
    for i in S1.split(' '):
        if i in S2:
            count+=1
        else:
            distinct_1.append(i)
    for i in S2.split(' '):
        if i not in S1:
            distinct_2.append(i)
    return count,distinct_1,distinct_2

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: 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
```

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

In [42]:
```
# 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

from math import log
# you can free to change all these codes/structure
def compute_log_loss(A):
    n=len(A)
    loss=0
    for i in A:
#         print(i[0],i[1])
        loss+=i[0]*log(i[1],10)+ (1.0-i[0])*log(1.0-i[1],10)
    return ((-1/n)*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
loss=compute_log_loss(A)
print(loss)
```

```
0.42430993457031635
```