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 [1]:
```
#referred and modified---->scribd.com
def matrix_mult(a,b):
    #To check the dimensions are same for the given matrix
    if len(a)==len(b):
        result=[]
        mult_row = len(b[0])*[0]
        for r in range(len(a)):
            result.append(mult_row[:])
        for i in range(len(a)):
            for j in range(len(b[0])):
                sum = 0
                for k in range(len(a[0])):
                    sum = sum + a[i][k]*b[k][j]
                    result[i][j]=sum
        return result
    else:
        return("Multiplication of the given matrix is not possible")


#a = [[1,3,4],[2,5,7],[5,9,6]]
#b = [[1,0,0],[0,1,0],[0,0,1]]
a = [[1,2],[3,4]]
b = [[1,2,3,4,5],[5,6,7,8,9]]
#a = [[1,2],[3,4]]
#b  = [[1,4],[5,6],[7,8],[9,6]]
matrix_mult(a,b)
```

Out[1]:
```
[[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 [2]:
```
#code source--->https://www.kaggle.com/paulrohan2020/pure-python-exercise
import random

#Defining function
def pick_a_number_from_list(A):
    #code for picking an element from with the probability propotional to its magi
    sum = 0
    cum_sum =[]
    for i in range(len(A)):
        sum = sum + A[i]
        cum_sum.append(sum)

    #Generating random numbers
    r = int(random.uniform(0,sum))
    print(r)
    number = 0

    for index in range(len(cum_sum)):
        if (r>=cum_sum[index] and r<cum_sum[index +1]):
            return A[index+1]
    return number

def sampling_based_on_magnitued():
    A = [0,5,27,6,13,28,100,45,10,79]
    a = dict()
    print(A,"\nsum of A:",sum(A))
    for i in range(1,100):
        number = pick_a_number_from_list(A)
        if number not in a:
            a[number] = 1
        else:
            a[number]+=1

            print(number)
    print(a)

sampling_based_on_magnitued()
```

```
[0, 5, 27, 6, 13, 28, 100, 45, 10, 79]
sum of A: 313
96
220
7
270
158
100
77
312
79
42
99
100
25
27
59
28
```

```
202
```

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 [3]:
```
import re

def replace_digits(string):
    for code in string:
        #replacing numbers by # and removing the ones other numbers
        new_s = '#' * len(re.sub(r'\D', '', code))
        print('Input:- {} ,  Output:- {} '.format(code ,new_s))

codes = ['234','a2b3c4','abc','#2a$#b%c%561#']
replace_digits(codes)
```

```
Input:- 234 ,  Output:- ###
Input:- a2b3c4 ,  Output:- ###
Input:- abc ,  Output:-
Input:- #2a$#b%c%561# ,  Output:- ####
```

Q4: Students marks dashboard

consider the marks list of class students given two lists
Students =
['student1','student2','student3','student4','student5','student6','student7','student8','student9','student1
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','student
6','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 [4]:
```
def get_student(Students,Marks):
    # to map students and marks
    diction = dict(zip(Students,Marks))

    print("a. Students who got top 5 ranks:")
    first_five = (sorted(diction.values())[5:])[::-1]
    for m in first_five:
        print(list(diction.keys())[list(diction.values()).index(m)], m)


    print("\nb. Students who got least 5 ranks:")
    last_five = (sorted(diction.values())[:5])

    for m in last_five:
        print(list(diction.keys())[list(diction.values()).index(m)], m)

    _max = max(marks)
    _min = min(marks)
    diff = _max - _min
    #25th and 75th percentile
    pre_25 = diff*0.25
    pre_75 = diff*0.75
    print("\nc. Students who got marks in between 25th and 75th percentile:")
    for m in sorted(diction.values()):
        if ( m >= pre_25 and m <= pre_75):
            print(list(diction.keys())[list(diction.values()).index(m)], m)
        else:
            pass

students=['student1','student2','student3','student4','student5','student6','stud
marks = [45, 78, 12, 14, 48, 43, 47, 98, 35, 80]
get_student(students,marks)
```

```
a. Students who got top 5 ranks:
student8 98
student10 80
student2 78
student5 48
student7 47

b. Students who got least 5 ranks:
student3 12
student4 14
student9 35
student6 43
student1 45

c. Students who got marks in between 25th and 75th percentile:
student9 35
student6 43
student1 45
student7 47
student5 48
```

Q5: Find the closest points

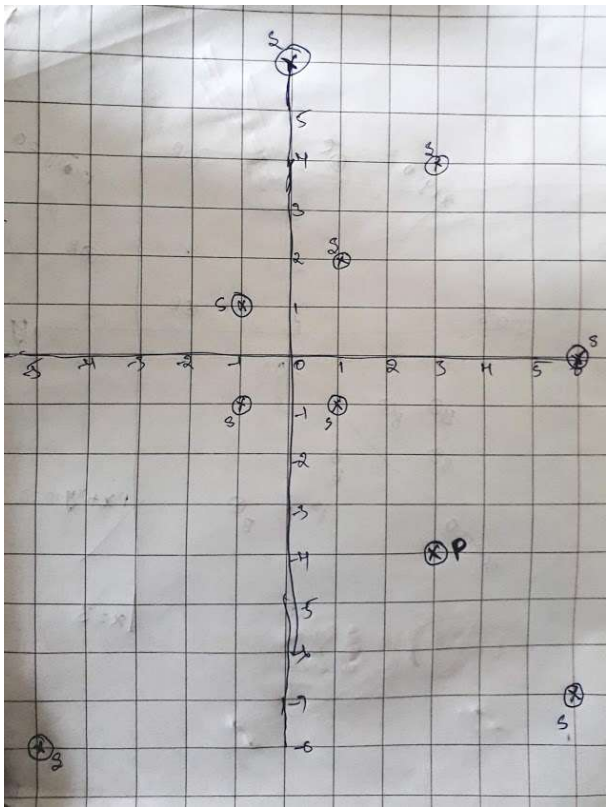
consider you have given n data points in the form of list of tuples like S=[(x1,y1),(x2,y2),(x3,y3),(x4,y4),(x5,y5),..,(xn,yn)] and a point P=(p,q)

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 [5]:
```
#source code--->https://www.kaggle.com/paulrohan2020/pure-python-exercises
import math

S = [(1, 2), (3, 4), (-1, 1), (6, -7), (0, 6), (-5, -8), (-1, -1), (6, 0), (1, -1
P = (3, -4)
def closest_points_to_p(S, P):
    clst_pts = []
    final_list = []

    for point in S:
        deno = math.sqrt((point[0] ** 2) + (point[1] ** 2)) * math.sqrt((P[0] **
        nume = point[0] * P[0] + point[1] * P[1]

        if deno != 0:
            cosine_distance_for_this_point = math.acos(nume / deno)
            clst_pts.append((cosine_distance_for_this_point, point))

    for item in sorted(clst_pts, key=lambda x: x[0])[:5]:
        final_list.append(item[1])

    return final_list

plts = closest_points_to_p(S, P)
print("The 5 closest points(based on cosine distance) in S from P:", *[point for
```

```
The 5 closest points(based on cosine distance) in S from 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
```

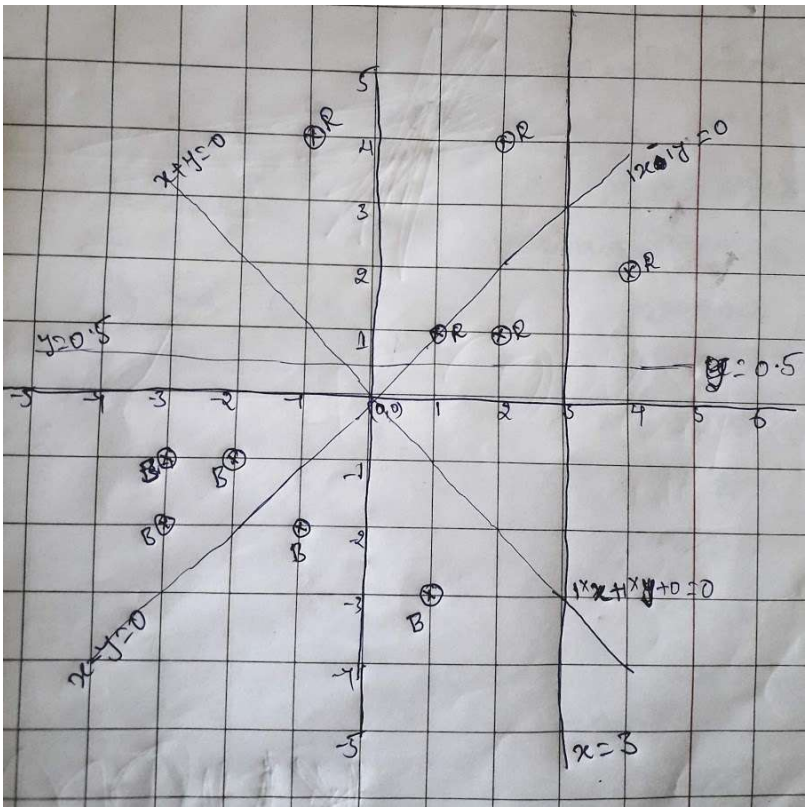
your task is to for each line that is given print "YES"/"NO", you will print yes, if all the red points are one side of the line and blue points are other side of the line, otherwise no

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 [6]:
```
#Modified from--->https://stackoverflow.com/questions/57188227/to-find-whether-a-
import math

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"]
# defining a function with three arguments
def i_am_the_one(a,b,c):
    bl = []
    rd= []
    # iterates through each element in red
    for i in range(len(red)):
        s = ((a*red[i][0])+(b*red[i][1])+c)
        rd.append(s)
    # iterates through each element in blue
    for i in range(len(blue)):
        r = ((a*blue[i][0])+(b*blue[i][1])+c)
        bl.append(r)


    finalrd = all(j > 0 for j in rd)
    finalbl = all(j < 0 for j in bl)

    if (finalbl == True and finalrd == True):
        print("Yes")
    else:
        print("NO" )

for line in lines:
    a, b, c = [float(x.strip()) for x in re.split('x|y', line)]
    i_am_the_one(a,b,c)
```

```
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 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 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, 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 [3]:
```
#https://stackoverflow.com/questions/57179618/filling-the-missing-values-in-the-s

# Replaces all the _'s with (x[a]+x[b])/(b-a+1)
def func(x, a, b):
    if a == -1:
        v = int(x[b])/(b+1)
        for i in range(a+1,b+1):
            x[i] = v
    elif b == -1:
        v = int(x[a])/(len(x)-a)
        for i in range(a, len(x)):
            x[i] = v
    else:
        v = (int(x[a])+int(x[b]))/(b-a+1)
        for i in range(a,b+1):
            x[i] = v
    return x

def curve_smoothing(miss_val):
    #splitting up the string into an array of characters
    x = miss_val.replace(" ","").split(",")

    # If it is not a space, then we add 1 to y[0] which will be used as our index
    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 every iteration
    for (a, b) in zip(y[:-1], y[1:]):
        func(x,a,b)
    return x

# Test cases
miss_val = [
    "_,_,_,24",
    "40,_,_,_,60",
    "80,_,_,_,_",
     "_,_,30,_,_,_,50,_,_"]

j=1
for i in miss_val:
    print ("Input{0}: {1} \nOutput{0}: {2}\n".format(j,i,curve_smoothing(i)))
    j+=1
```

```
Input1: _,_,_,24
Output1: [6.0, 6.0, 6.0, 6.0]

Input2: 40,_,_,_,60
Output2: [20.0, 20.0, 20.0, 20.0, 20.0]

Input3: 80,_,_,_,_
Output3: [16.0, 16.0, 16.0, 16.0, 16.0]
```

```
Input4: _,_,30,_,_,_,50,_,_
Output4: [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)=1/4, 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 [8]:
```
#dictionary of all possible outcomes
def compute_conditional_probabilites(A):
    for i in range(len(A)):
        k = A[i][0] + A[i][1]
        dict_1[k] += 1
        dict_2[A[i][1]] += 1
    #computing and printing probality for each conditional probability
    print('Probability of P(F=F1|S==S1)', (dict_1['F1S3']/dict_2['S3']))

dict_1 = {'F1S1': 0,'F2S1': 0,'F3S1': 0,'F4S1': 0,'F5S1': 0,'F1S2': 0,'F2S2': 0,'
dict_2 = {'S1': 0,'S2': 0,'S3': 0}

A = [['F1', 'S1'], ['F2', 'S2'], ['F3', 'S3'], ['F1', 'S2'], ['F2', 'S3'], ['F3',
compute_conditional_probabilites(A)
```

```
Probability of P(F=F1|S==S1) 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 [9]:
```
#Defining a function
def string_features(S1, S2):
    #taking each words of sentences as a set with the help of split
    a_words = set(S1.split(" "))
    b_words = set(S2.split(" "))
    #finding common words between S1 and S2
    a_ans = len(a_words.intersection(b_words))
    #words in S1 but not in S2
    b_ans = list(a_words - b_words)
    #words in S2 but not in S1
    c_ans = list(b_words - a_words)
    return a_ans,b_ans,c_ans

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:",a,"\nb:",b,"\nc:",c)
```

```
a: 7
b: ['F', '5', 'first']
c: ['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

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

In [10]:
```
#Hints from sitepoint.com

import math
def compute_log_loss(A):
    #computing the summation term
    log_Loss = sum([int(A[i][0])*math.log10(float(A[i][1]))+ (1-int(A[i][0]))*mat
    #Computing -(average of log_loss)
    log_Loss = log_Loss * (-1/(len(A)))
    return log_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("%0.7f"%loss)
```

0.4243099