Q1: Given two matrices please print the product of those two matrices

In [ ]:
```
# 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):
    if len(A[0]) == len(B):
        c,r = len(A), len(B[0])
        Matrix = [[0 for x in range(r)] for y in range(c)]
        for i in range(c):
            for j in range(r):
                for k in range(len(B)):
                    Matrix[i][j] += A[i][k] * B[k][j]
        return print(Matrix)
    else:
        return print('A*B = Not Possible')

A   = [ [1, 3, 4],
        [2, 5, 7],
        [5, 9, 6]
      ]
B   = [ [1, 0, 0],
        [0, 1, 0],
        [0, 0, 1]
      ]
matrix_mul(A, B)

A   = [[1, 2],[3, 4]]
B   = [[1, 2, 3, 4, 5],[5, 6, 7, 8, 9]]
matrix_mul(A, B)

A   = [[1,2],[3, 4]]
B   = [[1,4],[5,6],[7, 8],[9, 6]]
matrix_mul(A, B)
```

```
[[1, 3, 4], [2, 5, 7], [5, 9, 6]]
[[11, 14, 17, 20, 23], [23, 30, 37, 44, 51]]
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 [ ]:
```
A = [0,5, 27, 6, 13, 28, 100, 45, 10, 79]

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):
    # your code here for picking an element from with the probability propotional to its magnitude
    #.
    #.
    #.
    return #selected_random_number

def sampling_based_on_magnitued():
```

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

#sampling_based_on_magnitued()
```

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

In [ ]:
```
#https://stackoverflow.com/questions/59346340/randomly-selection-of-number-from-list-of-integers-which-is-propo
import random

def pick_a_number_from_list(A):
    sum=0
    cum_sum=[]
    for i in range(len(A)):
        sum = sum + A[i]
        cum_sum.append(sum)
    #print(cum_sum)
    r = int(random.uniform(0,sum))
    number=0
    #print(cum_sum,r,sum)
    for index in range(len(cum_sum)):
        if(r>=cum_sum[index] and r<cum_sum[index+1]):
          #print((r,cum_sum[index],r,cum_sum[index+1],index),A[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()
    #A.sort()
    #print(A,sum(A))
    for i in range(1,100):
        number = pick_a_number_from_list(A)
        #print(number)
        if number not in a:
            a[number] = 1
        else:
            a[number]+=1
    print(a)
sampling_based_on_magnitued()
```

```
{79: 20, 100: 39, 6: 2, 27: 9, 28: 8, 45: 13, 13: 4, 10: 3, 5: 1}
```

In [ ]:

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 [ ]:
```
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):
    c=0
    for i in range(len(String)):
        if String[i].isdigit():
            c+=1
    if c==0:
        return print('empty string')
    else:
        return print('#'*c)
#    return() # modified string which is after replacing the # with digits
String = '234'
replace_digits(String)
String = 'a2b3c4'
replace_digits(String)
String = 'abc'
```

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

```
###
###
empty string
####
```

In [ ]:

In [ ]:

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','student7','student8','student9',

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 [ ]:

```
Students = ['student1','student2','student3','student4','student5','student6','student7','student8','student9',
Marks = [45, 78, 12, 14, 48, 43, 47, 98, 35, 80]
# 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):
    d = dict(zip(Students, Marks))
    dic = sorted(d.items(), key=lambda x: x[1])
    my_list = (dic[-5:])
    my_list.sort(key=lambda x: x[1],reverse=True)
    top_5_students = my_list
    least_5_students = dic[0:5]
    s,m=zip(*sorted(zip(Students, Marks)))
    dic = dict()
    for i in range(0,10):
        if m[i]>25 and m[i]<75:
            dic[s[i]] = m[i]
    w=sorted(dic.items(), key=lambda x: x[1])

    students_within_25_and_75 = w

    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_board(Students, Marks)
def printt(listt):
    for i in listt:
```

```
        print(i)
    return
print('a. Who got top 5 ranks, in the descending order of marks')
printt(top_5_students)
print('--'*20)
print('b. Who got least 5 ranks, in the increasing order of marks')
printt(least_5_students)
print('--'*20)
print('c. Who got marks between >25th percentile <75th percentile, in the increasing order of marks')
printt(students_within_25_and_75)
```

```
a. Who got top 5 ranks, in the descending order of marks
('student8', 98)
('student10', 80)
('student2', 78)
('student5', 48)
('student7', 47)
----------------------------------------
b. Who got least 5 ranks, in the increasing order of marks
('student3', 12)
('student4', 14)
('student9', 35)
('student6', 43)
('student1', 45)
----------------------------------------
c. Who got marks between >25th percentile <75th percentile, in the increasing order of marks
('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 $cos^{-1}(\frac{(x \cdot p + y \cdot q)}{\sqrt{(x^2+y^2)} \cdot \sqrt{(p^2+q^2)}})$

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

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 [ ]:
```
S= [(1,2),(3,4),(-1,1),(6,-7),(0, 6),(-5,-8),(-1,-1),(6,0),(1,-1)]
P = [(3,-4)]
d = dict()
import math
for i,pair in enumerate(S):
  #print((pair[0]*P[0]))
  num = ((pair[0]*P[0][0]) + (pair[1]*P[0][1]))
  d1  = ((pair[0]**2) + (pair[1]**2))**(1/2)
  d2  = ((P[0][0]**2) + (P[0][1]**2))**(1/2)
  den = d1*d2
  d[i] = math.acos((num/den))
d = sorted(d, key = d.get)
ind = d[:5]
for i in ind:
  print(S[i])
```

```
(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

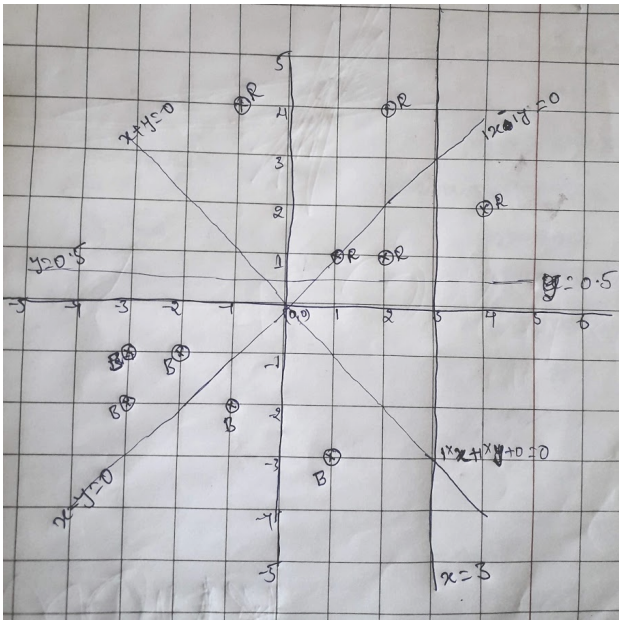
your task is to for each line that is given print "YES"/"NO", you will print yes, if all the red points are one side of the line and blue points are other side of the line, otherwise no

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 [ ]:
```python
import re
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"]

def fun1(q,points):
    count=list()
    for j in range(len(points)):
        if q[0][1] == 'x':
            a = float(q[0][0]) * float(points[j][0])
            #print(a)
        elif q[0][1] == 'y':
            a = float(q[0][0]) * float(points[j][1])
            #print(a)
        if q[2][1] == 'y':
            b = float(q[2][0]) * float(points[j][1])
```

```
            #print(b)
        elif q[2][1] == 'x':
            b = float(q[2][0]) * float(points[j][0])
            #print(b)
        if q[1]=='+':
            c = float(a + b)
        elif q[1] == '-':
            c = float(a-b)
        if q[3] == '+':
            d = c + float(q[4])
        elif q[3]=='-':
            d = c - float(q[4])
        count.append(d)
    return count

def fun2(list1):
    pos_count, neg_count = 0, 0
    for num in list1:
        if num >= 0:
            pos_count += 1
        else:
            neg_count += 1
    #print('pc',pos_count,'nc',neg_count)
    return pos_count,neg_count
for i in range(len(Lines)):
    q = re.split(r'([-+*/()])|\s+', Lines[i])
    c = q.count('')
    while c:
        q.remove('')
        c-=1
#    print(q,len(q))
    red = fun1(q,Red)
    blu = fun1(q,Blue)
    pc_red,nc_blu = fun2(red),fun2(blu)
    #print('r',pc_red,'b',nc_blu)
    if pc_red == (len(Red),0) and nc_blu == (0,len(Blue)):
        print('YES')
    else:
        print('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 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 [ ]:
```python
# 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

#https://stackoverflow.com/questions/56993719/need-to-find-the-missing-values-from-a-string-based-on-smoothening
# you can free to change all these codes/structure
def curve_smoothing(string):
    # your code
    output = string.split(',')
    pos = 0
    next_value = 0
    last_pos = 0
    last_value = 0
    while pos < len(output):
        if output[pos] != '_' or (pos + 1 == len(output)):
            if output[pos] != '_':
                next_value = int(output[pos])
            else:
                next_value = 0
            new_value = (next_value + last_value) / (pos - last_pos + 1)
            for i in range(last_pos, pos + 1):
                output[i] = new_value
            last_value = new_value
            last_pos = pos
        pos += 1
    return output#list of values

inp1 = "_,_,_,24"
inp2 = "40,_,_,_,60"
inp3 = "80,_,_,_,_"
inp4 = "_,_,30,_,_,_,50,_,_"
#smoothed_values= curve_smoothing(inp1)
print('inp1:',curve_smoothing(inp1))
print('inp2:',curve_smoothing(inp2))
print('inp3:',curve_smoothing(inp3))
print('inp4:',curve_smoothing(inp4))
```

```
inp1: [6.0, 6.0, 6.0, 6.0]
inp2: [20.0, 20.0, 20.0, 20.0, 20.0]
inp3: [16.0, 16.0, 16.0, 16.0, 16.0]
inp4: [10.0, 10.0, 12.0, 12.0, 12.0, 12.0, 4.0, 4.0, 4.0]
```

In [ ]:

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 [ ]:
```
# 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 compute_conditional_probabilites(A):
    # your code
    # print the output as per the instructions

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


def fun(f,s,a):
    n = 0
    for i in range(len(a)):
        if f==a[i][0] and s==a[i][1]:
            n+=1
    return n


f = set()
s = dict()
c1,c2,c3 = 0,0,0
a = A
for i in a:
    f.add(i[0])
    if i[1]=='S1':
        c1+=1
    elif i[1]=='S2':
        c2+=1
    elif i[1]=='S3':
        c3+=1
s['S1'] = c1
s['S2'] = c2
s['S3'] = c3
#f = list(f)


for i in sorted(f):
    for j in s.keys():
        n = fun(i,j,a)
        print('Probability of P(F={0}|S=={1}) = {2}/{3}'.format(i,j,n,s[j]))
    print('--'*50)
```

```
Probability of P(F=F1|S==S1) = 1/4
Probability of P(F=F1|S==S2) = 1/3
Probability of P(F=F1|S==S3) = 0/3
----------------------------------------------------------------------------------------------------
Probability of P(F=F2|S==S1) = 1/4
Probability of P(F=F2|S==S2) = 1/3
Probability of P(F=F2|S==S3) = 1/3
----------------------------------------------------------------------------------------------------
Probability of P(F=F3|S==S1) = 0/4
Probability of P(F=F3|S==S2) = 1/3
Probability of P(F=F3|S==S3) = 1/3
----------------------------------------------------------------------------------------------------
Probability of P(F=F4|S==S1) = 1/4
Probability of P(F=F4|S==S2) = 0/3
Probability of P(F=F4|S==S3) = 1/3
----------------------------------------------------------------------------------------------------
Probability of P(F=F5|S==S1) = 1/4
Probability of P(F=F5|S==S2) = 0/3
Probability of P(F=F5|S==S3) = 0/3
----------------------------------------------------------------------------------------------------
```

In [ ]:

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 [ ]:
```
# 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):
    ss1 = set(s1.split())
    ss2 = set(s2.split())
    a = len(ss1.intersection(ss2))
    b = list(ss1-ss2)
    c = list(ss2-ss1)
    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.', a,'\n','b.', b,'\n','c.', c)
```

```
a. 7
 b. ['first', '5', 'F']
 c. ['S', '3', 'second']
```

In [ ]:

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 $Yscore$ will be having float values

Your task is to find the value of $f(Y,Yscore) = -1/n * \Sigma foreach Y,Yscore pair(Y log10(Yscore) + (1-Y) log10(1-Yscore))$ 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

$-18 \cdot ((1 \cdot log10(0.4) + 0 \cdot log10(0.6)) + (0 \cdot log10(0.5) + 1 \cdot log10(0.5)) + ... + (1 \cdot log10(0.8) + 0 \cdot log10(0.2)))$

In [ ]:
```
# 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
import math as m
def compute_log_loss(A):
```

```
    summ = 0
    for i in range(len(A)):
        a = A[i][0]*(m.log10(A[i][1]))
        b = (1-A[i][0])*(m.log10(1-A[i][1]))
        c = a + b
        summ+=c
    loss = (-1/len(A)) * summ
    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
```