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 [3]:

```
###### write your python code here
# you can take the above example as sample input for your program to te
st
# it should work for any general input try not to hard code for only gi
ven input examples


# you can free to change all these codes/structure
# here A and B are list of lists
def matrix_mul(A, B,r,c):
    # write your code
    res=[[0 for i in range(c)] for j in range(r)]
    for i in range(len(A)):
        for j in range(len(B[0])):
            for k in range(len(B)):
                res[i][j]+=(A[i][k]) * (B[k][j])

    return res
print('Enter the no of rows and cols for matrics A')
r1=int(input())
c1=int(input())
print('Enter the no of rows and cols for matrics B')
r2=int(input())
c2=int(input())
if(c1!=r2):
    print('multiplication is not possible')
    exit()

else:
    print('Enter the element of metric A')
    A = [[int(input()) for x in range (c1)] for y in range(r1)]
    print('Enter the element of metric B')
    B=[[int(input()) for x in range(c2)] for y in range(r2)]
    print(matrix_mul(A,B,r1,c2))
```

```
Enter the no of rows and cols for matrics A
2
3
Enter the no of rows and cols for matrics B
3
3
Enter the element of metric A
1
2
3
4
5
6
Enter the element of metric B
0
1
5
4
7
0
0
2
1
[[8, 21, 8], [20, 51, 26]]
```

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 ex
```

periments.
f(100) > f(79) > f(45) > f(28) > f(27) > f(13) > f(10) > f(6) > f(5) > f(0)

In [1]:
```
import pandas as pd
from random import uniform
def pick_a_number_from_list(A,l):
    #print('sonu')
    sum1=0;
    for i in A:
        sum1+=i;
    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]



A = [0,5,27,6,13,28,100,45,10,79]
n = len(A)
for i in range(1,100):
    num = pick_a_number_from_list(A,n)
    print(num)
```

```
100
79
79
27
45
5
79

45
```

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

79
```

```
79
79
79
28
100
79
79
79
100
13
45
100
100
100
45
79
100
45
100
45
13
100
45
79
100
79
100
79
45
13
79
27
100
100
100
100
5
100
27

100
```

```
100
79
79
79
100
100
45
100
100
45
79
100
100
79
```

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 [5]:
```
import re
l = list(map(str,input().split()))
for i in l:
    new_s = '#' * len(re.sub(r'\D', '', i))
    print('Input = {} Output = {}'.format(i, new_s))
```

```
234 a2b3c4 abc #2a$#b%c%561#
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','studer
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','student
5','student6','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 [59]:

```
import math
from collections import Counter
from operator import itemgetter
Students=['student1','student2','student3','student4','student5','stude
nt6','student7','student8','student9','student10']
Marks = [45, 78, 12, 14, 48, 43, 47, 98, 35, 80]
dic=dict(zip(Students,Marks))
#print(dic)
print('a.  Who got top 5 rank in descending order: ')
for key,val in sorted(dic.items(), key=lambda item: item[1],reverse=Tru
e)[:5]:
          print("%s: %s" % (key, val))
print('b.Who got least 5 ranks, in the increasing order of marks')
for key,val in sorted(dic.items(),key=lambda item: item[1],reverse=Fals
e)[:5]:
          print('%s: %s ' % (key,val))
print('c.Who got marks between >25th percentile <75th percentile, in th
e increasing order of marks')
dic = dict(sorted(x.items(), key=lambda kv: kv[1]))
p1=math.floor(len(dic)*.25)
p2=math.ceil(len(dic)*.75)
i=0
for key, value in dic.items():
    i+=1
    if(i>p1 and i<p2):
        print(key,": ",value)
```

```
a.  Who got top 5 rank in descending order:
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 incre
asing order of marks
student9 :  35
student6 :  43
student1 :  45
student7 :  47
student5 :  48
```

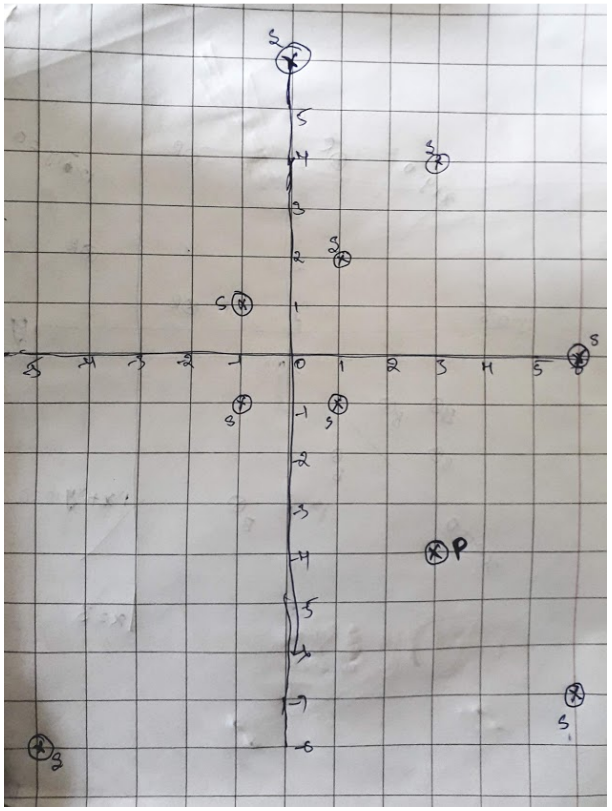
Q5: Find the closest points

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

```
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 [8]:
```
from math import acos
from math import sqrt
m={}
s= [(1,2),(3,4),(-1,1),(6,-7),(0, 6),(-5,-8),(-1,-1),(6,0),(1,-1)]
p=(3,-4)
```

```
l=[()]
l.pop()
for i in range(len(s)):
    a=(s[i][0]*p[0])+(s[i][1]*p[1])
    b=sqrt((s[i][0]*s[i][0])+(s[i][1]*s[i][1]))
    c=sqrt( (p[0]*p[0])+(p[1]*p[1]))
    z=acos(a/(b*c))
    k=(s[i][0],s[i][1])
    m[k]=z
m={k: v for k, v in sorted(m.items(), key=lambda item: item[1])}
flag=0
for i,j in m.items():
    flag+=1
    print(i,j,sep=" ")
    if(flag==5):
        break
```

```
(6, -7) 0.06512516333438509
(1, -1) 0.14189705460416438
(6, 0) 0.9272952180016123
(-5, -8) 1.2021004241368467
(-1, -1) 1.4288992721907328
```

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

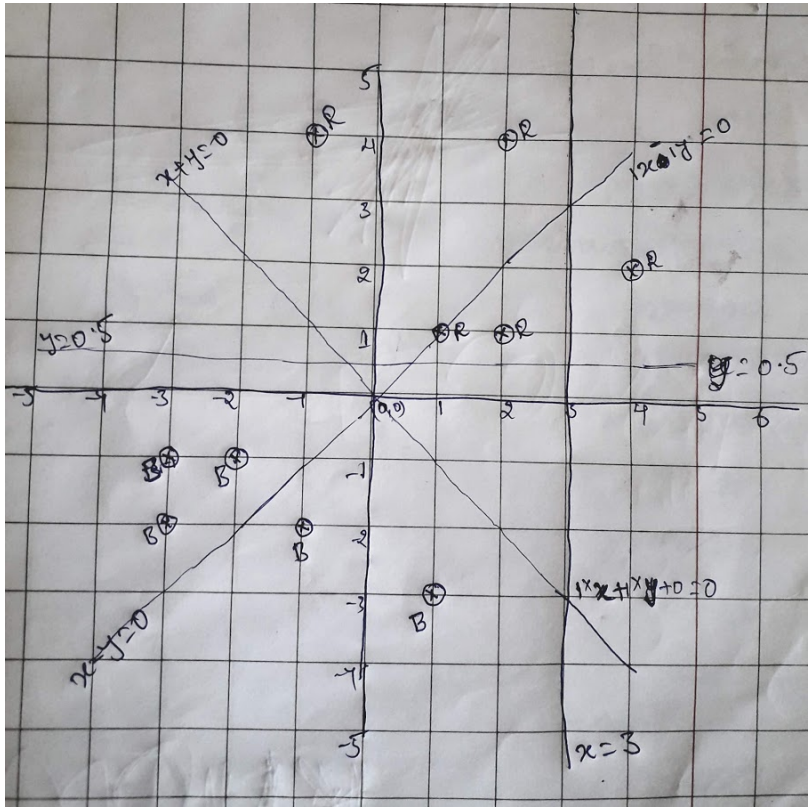
```
Red =[(R11,R12),(R21,R22),(R31,R32),(R41,R42),(R51,R52),..,(Rn1,
Rn2)]
Blue=[(B11,B12),(B21,B22),(B31,B32),(B41,B42),(B51,B52),..,(Bm1,
Bm2)]
```

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 o
f 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 [20]:

```
import math
import re
l=[[]]
l.pop()
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:
    p=re.findall(r'[\d\.\+\-]+',i)
    l.append(p)
#print((l))

for p in l:
    i=0
    k=0
    #print('p: ',p)
    mul1_Red=[]
    mul2_Blue=[]
    for x in Red:
        dist1=(float(p[0])*Red[i][0]) + (float(p[1])*Red[i][1]) + (floa
t(p[2]))
        dist2=(float(p[0])*Blue[i][0]) + (float(p[1])*Blue[i][1]) + (fl
oat(p[2]))
        mul1_Red.append(dist1)
        mul2_Blue.append(dist2)
        i+=1
    #print(mul1_Red)
    #print(mul2_Blue)
    for m in mul1_Red:
        for n in mul2_Blue:
            if( (m>0 and n>0) or (m<0 and n<0)):
                print('NO')
                k=1
                break
        if(k==1): break
    if(k==0):
        print('YES')
```

```
YES
NO
NO
YES
```

In [ ]:

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 distri
buted 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 di
stributed qually to all 5 places

Ex 3: 80, _, _, _, _  ==> 80/5,80/5,80/5,80/5,80/5 ==> 16, 16, 1
6, 16, 16 i.e. the 80 is distributed qually to all 5 missing val
ues 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 value
s (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 (1
0, 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 [12]:
```
def replaceleft(sums,start,end):
    div=end-start+1
    for i in range(start,end+1):
        ans.append(sums/div)
    return (sums/div)
S=input()
a=S.split(',')
ans=[]
end,start,sums=0,0,0
for i in range(len(a)):
    if(a[i]!='_'):
        end=i
        if(start==end):
            sums=int(float(a[i]))+sums
        else:
            sums+=int(float(a[i]))
            sums=replaceleft(sums,start,end)
```

```
            if(i!=len(a)-1):
                ans.pop()
                start=i
    else:
        continue
if(end!=len(a)-1):
    end=len(a)-1
    replaceleft(sums,start,end)
print((ans))
```

```
_,_,30,_,_,_,50,_,_
[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,S
1],[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 [22]:
```
A = [['F1','S1'],['F2','S2'],['F3','S3'],['F1','S2'],['F2','S3'],['F3',
'S2'],['F2','S1'],['F4','S1'],['F4','S3'],['F5','S1']]
cnt=15
while(cnt):
    F=input()
    S=input()
    num=0
    deno=0
    for i in range(len(A)):
        if(A[i][1]==S):
            deno+=1
        if(A[i][1]==S and A[i][0]==F):
            num+=1
    if(deno!=0):
        print(num,'/',deno,end=' ')
        print()
    else:
        print(0)
    cnt-=1
```

```
F1
S1
1 / 4
F1
S2
1 / 3
F1
S3
0 / 3
F2
S1
1 / 4
F2
S2
1 / 3
```

```
F2
S3
1 / 3
F3
S1
0 / 4
F3
S2
1 / 3
F3
S3
1 / 3
F4
S1
1 / 4
F4
S2
0 / 3
F4
S3
1 / 3
F5
S1
1 / 4
F5
S2
0 / 3
F5
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 [13]:

```
s1= "the first column F will contain only 5 uniques values"
s2= "the second column S will contain only 3 uniques values"
s=s1+" "+s2
l1=list(s1.split(' '))
l2=list(s2.split(' '))
print(l1)
print(l2)
l3=[]
l4=[]
cnt=0
for word in l1:
    if(word in l2):
        cnt+=1
    if(word not in s2):
        l3.append(word)
print('a.' ,cnt)
print('b. ',l3)
for w in l2:
    if(w not in s1):
        l4.append(w)
print('c. ',l4)
```

```
['the', 'first', 'column', 'F', 'will', 'contain', 'only', '5', 'unique
s', 'values']
['the', 'second', 'column', 'S', 'will', 'contain', 'only', '3', 'uniqu
es', 'values']
a. 7
```

```
b.  ['first', 'F', '5']
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
$f(Y, Y_{score}) = -1 * \frac{1}{n}\Sigma_{foreachY,Y_{score}pair}(Ylog10(Y_{score}) + (1-Y)log10(1-Y_{sco}$
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
```

$\frac{-1}{8} \cdot ((1 \cdot log_{10}(0.4) + 0 \cdot log_{10}(0.6)) + (0 \cdot log_{10}(0.5) + 1 \cdot log_{10}(0.5)) + \ldots$
$+(1 \cdot log_{10}(0.8) + 0 \cdot log_{10}(0.2)))$

In [14]:
```
#Done
from math import log
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]]
ans=0
for i in range(len(A)):
    ans+=(A[i][0]*log(A[i][1],10)) + ( (1-A[i][0])*log(1-A[i][1],10))
    #print(ans)
ans=(-1/len(A))*ans
print(ans)
```

```
0.42430993457031635
```

In [ ]:

In [ ]: