Python: without numpy or sklearn

```
<h3> Q1: Given two matrices please print the product of those two matrices </h3>
<pre>

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
</pre>
```

In [2]:

```
# 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):



    acc=len(a[0])
    brc=len(b)

    if acc==brc:
        res=[]
        for i in range(len(a)):
            row=[]
            for j in range(len(b[0])):
                prod=0
                for k in range(len(b)):
                    prod+=a[i][k]*b[k][j]
                row.append(prod)
            res.append(row)

    else:
        print("multiplication is not possible")
    return res
a=[[1,2,3],[3,4,5],[7,8,9]]
b=[[1,0,1],[0,1,1],[5,7,1]]
matrix_mul(a, b)
```

Out[2]:

```
[[16, 23, 6], [28, 39, 12], [52, 71, 24]]
```

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 [8]:

```
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
import random
from collections import Counter
ma=[0,8,25,16,13,28,100,47,10,179]
def selectanum(l):
    s = sum(l)
    d1=dict()
    for i in range(len(l)):
        d1[i] = l[i] / s
    dt = dict()
    c = 0
    s = list(d1.values())
    add = 0
    for i in s:
        add += i
        dt[c] = add
        c += 1
    r = random.uniform(0.0,1.0)
    for i in range(len(s)):
        if r < dt[i]:
            val= i + 1
            break
    return val

def sampling_magnitude():
    h=[]
    for i in range(1,100):
        number=selectanum(ma)
        h.append(ma[number-1])
    return Counter(h)

print(sampling_magnitude())
```

```
Counter({179: 48, 100: 18, 47: 13, 25: 5, 10: 4, 28: 4, 16: 4, 13: 3})
```

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 [6]:

```
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
import string
def replace_digits(s):
    l=[]
    for i in s:
        if i in string.digits:
            l.append(i.replace(i,'#'))
        elif i in string.punctuation:
            l.append(i.replace(i,''))
        elif i in string.ascii_letters:
            l.append(i.replace(i,''))
    l=''.join(l)
    if l=='':
        print('empty string')
    return l
print(replace_digits('abc'))
```

```
empty string
```

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','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','stude
nt7','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 [9]:

```
import random
import math

l1 = []
l2 = []
for i in range(1, 11):
    l1.append('student{}'.format(i))
for i in range(10):
    l2.append(random.randint(1, 100))
print(l2)
d = dict(zip(l1, l2))
d = sorted(d.items(), key=lambda x: x[1])
print(dict(d))
first5 = dict(d[:5])
last5 = dict(reversed(d[-5:]))
print('top5 in ascending order')
for i, j in first5.items():
    print(i, '  ', j)
print('top 5 in descending order')
for i, j in last5.items():
    print(i, '  ', j)
d=dict(d)
def percentile(per):
    tp=per*len(d.values())

    if int("{0:.1f}".format(tp)[-1])<5:
        tp=math.floor(tp)
    elif int("{0:.1f}".format(tp)[-1])>=5:
        tp=math.ceil(tp)
    elif int("{0:.1f}".format(tp)[-1])==0:
        tp=sum(list(d.values())[tp+1:tp+3])/2
    return tp
s5=percentile(0.75)
t5=percentile(0.25)

print("students between 25 and 75 (inclusive of 25 and 75 ) percentile")
for i,j in d.items():
    if j>=list(d.values())[t5] and j<=list(d.values())[s5]:
        print(i,j)
```

```
[1, 33, 9, 96, 80, 91, 73, 97, 71, 64]
{'student1': 1, 'student3': 9, 'student2': 33, 'student10': 64, 'student
9': 71, 'student7': 73, 'student5': 80, 'student6': 91, 'student4': 96, 's
tudent8': 97}
top5 in ascending order
student1    1
student3    9
student2    33
student10    64
student9    71
top 5 in descending order
student8    97
student4    96
student6    91
student5    80
student7    73
students between 25 and 75 (inclusive of 25 and 75 ) percentile
student10 64
```

```
student9 71
student7 73
student5 80
student6 91
student4 96
```

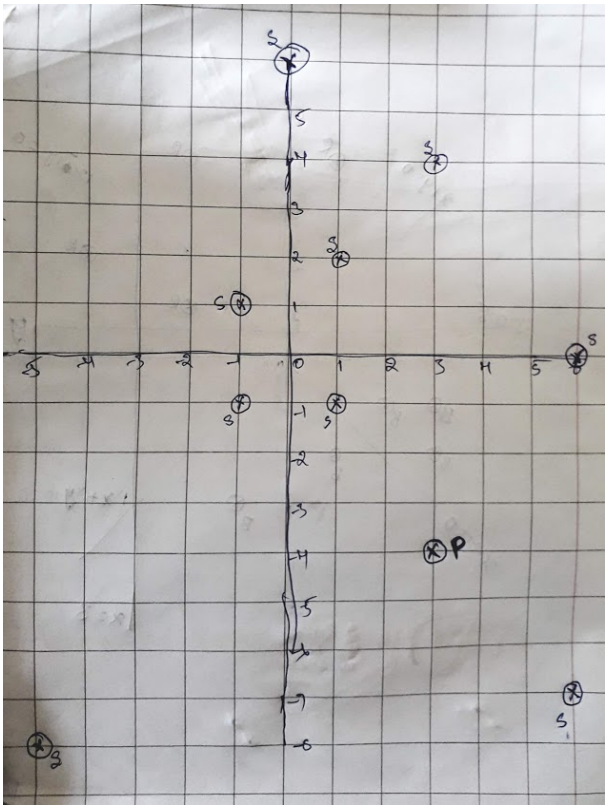
Q5: Find the closest points

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 [10]:

```
import math
s=[(1,2),(3,4),(-1,1),(6,-7),(0,6),(-5,-8),(-1,-1),(6,0),(1,-1)]
p,q=(3,-4)
#s=math.acos((x*p+y*q)/(math.sqrt(x**x+y**y)-math.sqrt(p**p+q*q)))
d={}
for i in s:
    x,y=i
    v=math.acos((x * p + y * q) / (((x * x + y * y)**(1/2))*((p * p + q * q)**(1/2))))
    d[i]=v
print(d)
s=sorted(d,key=lambda x:d[x])
for i in s[:5]:
    print(i)
#items=[]
#for item in sorted(d.items(),key=lambda x:x[1]):
#    items.append(item)
#print(items[:5])
```

```
{(1, 2): 2.0344439357957027, (3, 4): 1.8545904360032246, (-1, 1): 2.99969559
89856287, (6, -7): 0.06512516333438509, (0, 6): 2.498091544796509, (-5, -8):
1.2021004241368467, (-1, -1): 1.4288992721907328, (6, 0): 0.927295218001612
3, (1, -1): 0.14189705460416438}
(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 inter
cept.
```

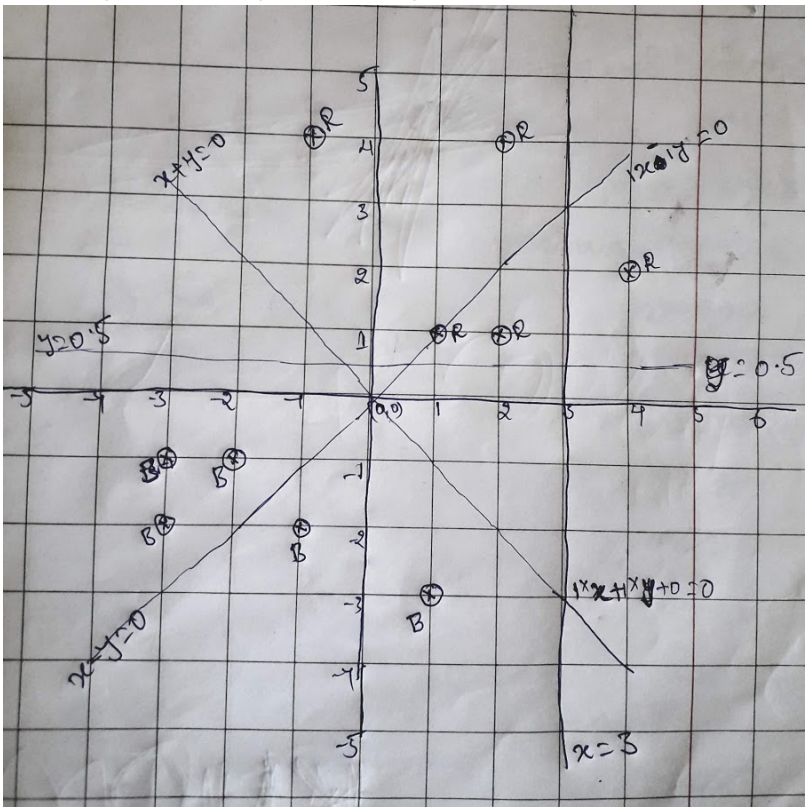
Your task here is to print "YES"/"NO" for each line given. You should print YES, if all the red points are one side of the line and blue points are on other side of the line, otherwise you should print NO.

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 [11]:

```
import re
r=[(1,1),(2,1),(4,2),(2,4),(-1,4)]
blp=[(-2,-1),(-1,-2),(-3,-2),(-3,-1),(1,-3)]
l=['1x+1y+0','1x-1y+0','1x+1y-3','0x+1y-0.5']
lc=[]
#if y < ((-a / b) * x - (c / b)):
#    rl.append('left')
#elif y > ((-a / b) * x - (c / b)):
#    rl.append('right')
#else:
#    rl.append('plane')
for i in l:
    s=re.findall(r'[\d\.\-]+',i)
    lc.append(s)
print(lc)
def side(f,s):
    m=[]
    for i,j in enumerate(f):
        a,b,c=j
        a,b,c=float(a),float(b),float(c)
        rl=[]
        for k in s:
            x,y=k
            if y < ((-a/b)*x-(c/b)):
                rl.append('left')
            elif y > ((-a/b)*x-(c/b)):
                rl.append('right')
            else:
                rl.append('plane')
        m.append(rl)
    return m



for i in range(len(l)):
    if side(lc,r)[i]==['right']*len(r) and side(lc,blp)[i]==['left']*len(blp):
        print('yes')
    else:
        print('No')
```

```
[['1', '1', '0'], ['1', '-1', '0'], ['1', '1', '-3'], ['0', '1', '-0.5']]
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 the 24 equal
ly to all 4 places

Ex 2: 40, _, _, _, 60 ==> (60+40)/5,(60+40)/5,(60+40)/5,(60+40)/5,(60+40)/5 ==> 2
0, 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 [12]:

```
import re
import string
import sys
i1="_,30,_,_,_,50,_"
i1=i1.split(',')
sum=0
sec=0
for i,j in enumerate(i1):
    if j!='_':
        if i==0 or i==len(i1)-1 or (i==0 and i==len(i1)-1):
            sum+=int(j)
            for ix in range(len(i1)):
                i1[ix]=sum/len(i1)
            print(i1)
            sys.exit()

        else:
            sum += int(j)
            k=i
            for o1 in range(k+1):
                i1[o1]=sum/(k+1)
            middle=i1[i]
            break

for ip in range(k+1,len(i1)):
    if i1[ip]!='_':
        sec=i1[ip]
        break
for iq in range(k,ip+1):
    i1[iq] = (int(sec)+middle )/ (ip+1-k)



last=i1[ip]
for ic in range(ip,len(i1)):
    i1[ic]=last/(len(i1)-ip)
print(i1)

#def replace(k):
#    for r in range(k+1):
#        i1[r]=sum/(k+1)
#    return i1
#print(replace(k))
```

```
[15.0, 13.0, 13.0, 13.0, 13.0, 6.5, 6.5]
```

In [ ]:

<h3> Q8: Find the probabilities</h3>

You will be given a list of lists, each sublist will be of length 2 i.e. [[x,y],[p,q],[l,m]]
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)
<pre>
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)
</pre>
Ex:

<pre>
[[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
</pre>

In [15]:

```
# 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


from collections import Counter
ma=[['F1','S1'],['F2','S2'],['F3','S3'],['F1','S2'],['F2','S3'],['F3','S2'],['F2','S1'],['F
#p(f=f1|s==s1)=1/4
fir=sorted(set(map(lambda x:x[0],ma)),key=lambda x:x[1])
sec=sorted(set(map(lambda x:x[1],ma)),key=lambda x:x[1])
print(fir)

c=dict(Counter(sec))
m,n=0,0

def compute(ma):
    for ia in fir:
        for ja in sec:
            num, den = 0, 0
            for i in range(len(ma)):

                if ma[i][1] == ja:
                    den += 1
                    if ma[i][0] == ia:
                        num += 1
            print( "p(f={}|s={})=".format(ia, ja), num / den)
compute(ma)
```

```
['F1', 'F2', 'F3', 'F4', 'F5']
p(f=F1|s=S1)= 0.25
p(f=F1|s=S2)= 0.3333333333333333
p(f=F1|s=S3)= 0.0
p(f=F2|s=S1)= 0.25
p(f=F2|s=S2)= 0.3333333333333333
p(f=F2|s=S3)= 0.3333333333333333
p(f=F3|s=S1)= 0.0
p(f=F3|s=S2)= 0.3333333333333333
p(f=F3|s=S3)= 0.3333333333333333
p(f=F4|s=S1)= 0.25
p(f=F4|s=S2)= 0.0
p(f=F4|s=S3)= 0.3333333333333333
p(f=F5|s=S1)= 0.25
p(f=F5|s=S2)= 0.0
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 [14]:

```
# write your python code here
# you can take the above example as sample input for your program to test
s1="the first column f will contain only 5 unique values"
s2="the second column s will contain only 3 unique values"
def st_feature(s1,s2):
    s1=s1.split(" ")
    s2=s2.split(" ")
    count=0
    for i in s1:
        for j in s2:
            if i==j:
                count+=1
    return count,list(set(s1) - set(s2)),list(set(s2) - set(s1))
s1="the first column f will contain only 5 unique values"
s2="the second column s will contain only 3 unique values"
a,b,c=st_feature(s1,s2)
print(" ",a,"\n",b,"\n",c)
```

```
 7
['first', '5', 'f']
['3', 's', 'second']
```

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
$f(Y, Y_{score}) = -1 * \frac{1}{n}\Sigma_{foreach Y, Y_{score} pair}\,(Y log10(Y_{score}) + (1-Y) log10(1-Y_{score}))$ 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
```

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

In [13]:

```
from math import log10
import math

def compute(s):
    sum=0
    for i in s:
        y,ys=i
        #f=(-1/len(s))
        m=(y*log10(ys)+(1-y)*log10(1-ys))
        sum+=m
    return (-1/len(s))*sum
s=[[1, 0.4], [0, 0.5], [0, 0.9], [0, 0.3], [0, 0.6], [1, 0.1], [1, 0.9], [1, 0.8]]
print(round(compute(s),5))
```

```
0.42431
```

In [ ]:

```

```