# 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 [46]:

```
def matrix_mul(A, B):
    mul=[]
    if(len(A[0]) != len(B)):
        print("Multiplication of matrix not possible")
    else:
        for i in range(len(A)):
            res = []
            for j in range(len(B[i])):
                sum = 0
                for k in range(len(B)):
                    sum = sum + A[i][k]*B[k][j]
                res.append(sum)
            mul.append(res)
    return mul

A1 = [[1,3,4],[2,5,7],[5,9,6]]
B1 = [[1,0,0],[0,1,0],[0,0,1]]

print("A1 * B1 = ",matrix_mul(A1, B1))

print("-"*50)

A2   = [[1,2],[3,4]]
B2   = [[1,4],[5,6],[7,8],[9,6]]
print("A2 * B2 = ",matrix_mul(A2, B2))
```

```
A1 * B1 =  [[1, 3, 4], [2, 5, 7], [5, 9, 6]]
--------------------------------------------------
Multiplication of matrix not possible
A2 * B2 =  []
```

## 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 [128]:

```python
def sampling_based_on_magnitude(cum_sum):
    r = random.uniform(0,1)
    for j in range(len(cum_sum)) :
        if r <  cum_sum[j] :
            print(A[j])

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

S = 0
for i in range(0, len(A)):
    S = S + A[i]
sum = [i/S for i in A]

cum_sum = []
cum_sum.append(sum[0])
for j in range(1, len(sum), 1) :
    cum_sum.append(cum_sum[-1] + sum[j])

op = []
for ele in range(100) :
    op.append(sampling_based_on_magnitude(cum_sum))
```

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

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

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

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

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

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

```
10
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 [117]:

```
import re

def replace_digits(String):
    str1 = re.sub('\D','',String)  #\D : Matches any non-digit character
    str2 = re.sub('\d','#',str1)   #\d : Matches decimal digit
    return(str2)

String = input("A = ")
ans = replace_digits(String)
print(ans)
```

```
A = a1b2c3d4
####
```

## 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','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 [7]:

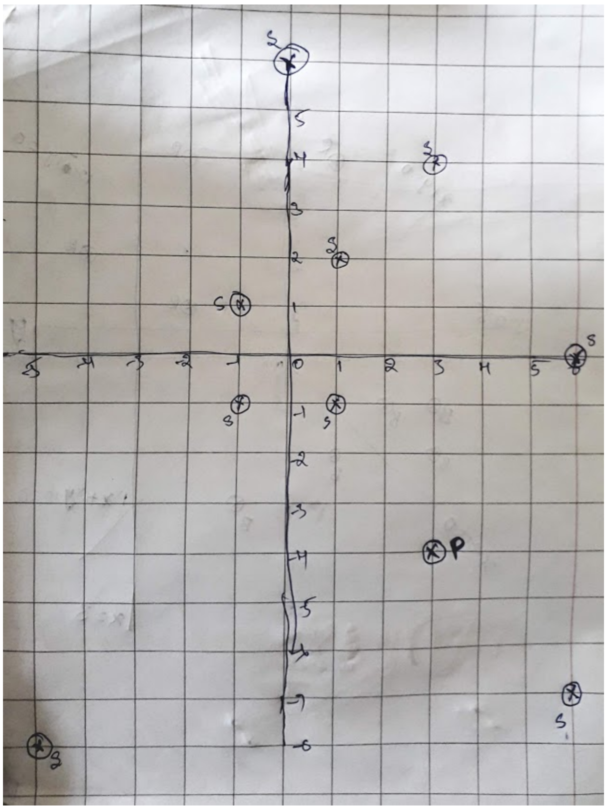
```
def display_dash_board(students, marks):
    res = {students[i]: marks[i] for i in range(len(students))}

    top_5_students = sorted(res.items(), key=lambda x: x[1], reverse=True)[:5]
    print("Top 5 ranks in the descending order of marks :")
    for x, y in top_5_students:
        print(x, y)

    least_5_students = sorted(res.items(), key=lambda x: x[1], reverse=False)[:5]
    print("\nLeast 5 ranks in the increasing order of marks :")
    for x, y in least_5_students:
        print(x, y)

    max_mark = max(res.keys(), key=(lambda k: res[k]))
    min_mark = min(res.keys(), key=(lambda k: res[k]))
    diff = res[max_mark] - res[min_mark]
    pre_25 = diff * 0.25
    pre_75 = diff * 0.75
    students_within_25_and_75 = sorted(res.items(), key=lambda x: x[1], reverse=False)
    print("\nStudents with marks between 25th percentile and 75th percentile :")
    for x, y in students_within_25_and_75:
        if (y>pre_25 and y<pre_75):
            print(x, y)

Students=['student1','student2','student3','student4','student5','student6','student7',
'student8','student9','student10']
Marks = [45, 78, 12, 14, 48, 43, 47, 98, 35, 80]

display_dash_board(Students, Marks)
```

```
Top 5 ranks in the descending order of marks :
student8 98
student10 80
student2 78
student5 48
student7 47

Least 5 ranks in the increasing order of marks :
student3 12
student4 14
student9 35
student6 43
student1 45

Students with marks between 25th percentile 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 [367]:

```
import math

def closest_points_to_p(S, P):
    cosdist = {}
    p = P[0]
    q = P[1]

    for i in range(len(S)):
        x = S[i][0]
        y = S[i][1]
        cosdist[(x,y)] = math.acos((x*p+y*q) / (math.sqrt(x**2 + y**2) * math.sqrt(p**2
+ q**2)))
        sort = sorted(cosdist.items(), key = lambda x: x[1], reverse=False)
    return [sort[i][0] for i in range(5)]

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[367]:

```
[(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

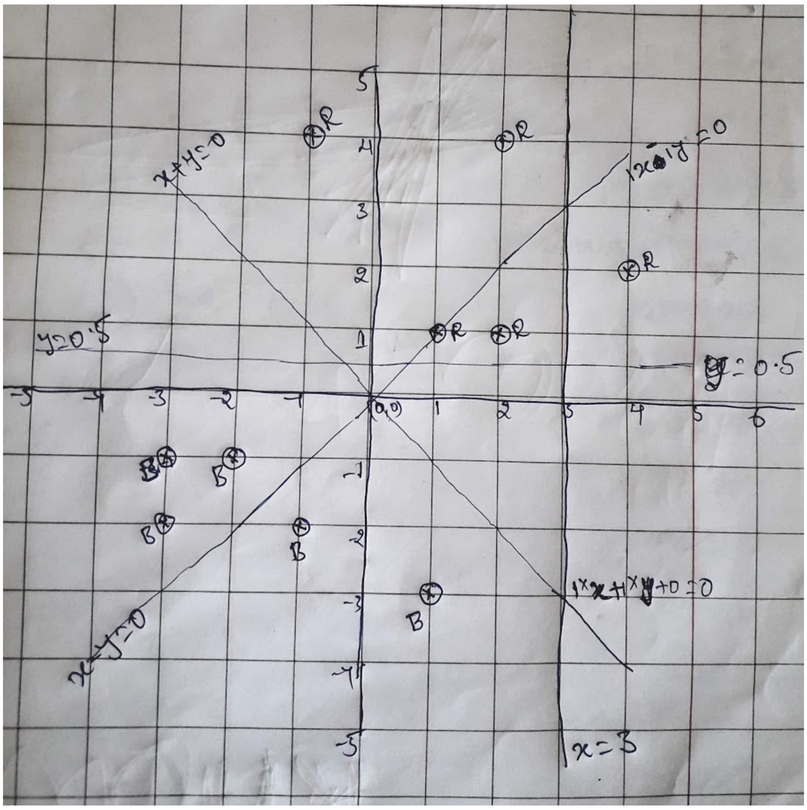
```
Red =[(R11,R12),(R21,R22),(R31,R32),(R41,R42),(R51,R52),..,(Rn1,Rn2)]
Blue=[(B11,B12),(B21,B22),(B31,B32),(B41,B42),(B51,B52),..,(Bm1,Bm2)]
```

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 interc
ept
```

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 [101]:

```
import math
import re

def i_am_the_one(red,blue,line):
    r, b = [], []

    x, y, z = [float(i) for i in re.split('x|y', line)]

    for i in range(len(red)):
        if x*(red[i][0]) + y*(red[i][1]) + z > 0:
            r.append(1)
        else:
            r.append(0)

    for j in range(len(blue)):
        if x*(blue[j][0]) + y*(blue[j][1]) + z > 0:
            b.append(1)
        else:
            b.append(0)

    for i in range(len(r)-1):
        if r[i] == r[i+1]:
            f1 = 1
        else:
            f1 = 0

    for j in range(len(b)-1):
        if b[i] == b[i+1]:
            f2 = 1
        else:
            f2 = 0

    return 'YES' if (f1 == 1 and f2 == 1 and r[0] != b[0]) else 'NO'

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:
    yes_or_no = i_am_the_one(Red, Blue, i)
    print(yes_or_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 equ
ally 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 place
s

Ex 3: 80, _, _, _, _  ==> 80/5,80/5,80/5,80/5,80/5 ==> 16, 16, 16, 16, 16 i.e. t
he 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, 1
2, 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 [99]:

```
def curve_smoothing(string):
    op = string.split(',')
    ind, ind2 = 0, 0
    num, num2 = 0, 0

    for ind in range(len(op)):
        if op[ind] != '_' or (ind + 1 == len(op)):
            if op[ind] != '_':
                num2 = int(op[ind])
            else:
                num2 = 0
            res = (num2 + num) / (ind - ind2 + 1)

            for i in range(ind2, ind + 1):
                op[i] = res
            num = res
            ind2 = ind
    return op

S1 = "_,_,_,24"
S2 = "40,_,_,_,60"
S3 = "80,_,_,_,_"
S4 =  "_,_,30,_,_,_,50,_,_"

print(curve_smoothing(S4))
```

```
[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,S
1]]

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 [217]:

```
def compute_conditional_probabilites(A, a, b):
    count_a, count_b =  0, 0
    ref = []
    for ele in A:
        if b == ele[1]:
            count_b = count_b + 1
            ref.append(ele)

    for i in ref:
        for j in i:
            if a == j :
                count_a = count_a + 1
    print("Probability of P(F= {}|S=={}) = {}/{}".format(a, b, count_a, count_b))



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

for ele in U:
    a, b = ele[0], ele[1]
    compute_conditional_probabilites(A, a, b)
```

```
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
```

## 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 [238]:

```python
def string_features(S1, S2):
    A = S1.split(" ")
    B = S2.split(" ")
    count = 0
    a = []
    for word_a in A:
        for word_b in B:
            if word_a == word_b:
                count= count + 1
    return count, set(A) - set(B), set(B) - set(A)

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. {}\nb. {}\nc. {}".format(a,list(b),list(c)))
```

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

## 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
```

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

In [254]:

```
import math

def compute_log_loss(A):
    n, sc = 0, 0
    for i in A:
        Y, Ys = i[0], i[1]
        sc = sc + (Y * math.log(Ys,10) + (1-Y) * math.log(1-Ys,10))
        n = n + 1
    loss = -1/n * sc
    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
```