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 [ ]:

```
# 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):
    # write your code
    m1, n1 = (len(A), len(A[0]))
    m2, n2 = (len(B), len(B[0]))

    if n1!=m2:
        return 'A*B = Not possible'
    result =[[0 for n in range(n2)] for m in range(m1)]
    for i in range(m1):
        for j in range(n2):
            for k in range(m2):
                result[i][j] += A[i][k]*B[k][j]

    return(result)

matrix_mul(A, B)
```

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 [11]:

```
from random import uniform
from collections import Counter
# 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
    x = uniform(0,1)
    total = sum(A)
    s = [0 for j in range(len(A))]
    s[0] = A[0]
    for i in range(1,len(A)):
        s[i] = s[i-1]+A[i]
        if s[i]/total > x:
            return A[i]

def sampling_based_on_magnitued():
    li = []
    for i in range(1,100):
        number = pick_a_number_from_list(A)
        li.append(number)
    print('Num : Count')
    print('===========')
    for key,value in Counter(li).items():
        print(key, ' : ', value)

A = [0, 5, 27, 6, 13, 28, 100, 45, 10, 79]
sampling_based_on_magnitued()
```

```
Num : Count
===========
100  :  32
27  :  7
45  :  17
28  :  10
13  :  5
6  :  3
79  :  23
10  :  2
```

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 [3]:

```
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):
    # write your code
    reg = re.compile(r'\d')
    reg = re.compile(r'\d')
    s = reg.findall(String)
    s = ('').join(['#']*len(s))
    return(s) # modified string which is after replacing the # with digits

replace_digits(String)
```

Out[3]:

```
'####'
```

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','student7','student8','st
t9','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 [17]:

```
# 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):
    # write code for computing top top 5 students
    dash_dic = {}
    l = len(students)
    for i in range(l):
        dash_dic[students[i]] = marks[i]
    dash_dic = sorted(dash_dic.items(), key=lambda item:item[1])

    top_5_students = dash_dic[:-6:-1]
    # write code for computing top least 5 students
    least_5_students = dash_dic[:5]
    # write code for computing top least 5 students
    index_25 = int(l*0.25)
    index_75 = int(l*0.75)
    students_within_25_and_75 = dash_dic[index_25:index_75]

    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)
print('a.')
for student in top_5_students:
    print(student[0],student[1])
print()
print('b.')
for student in least_5_students:
    print(student[0],student[1])
print()
print('c.')
for student in students_within_25_and_75:
    print(student[0],student[1])
```

```
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
```

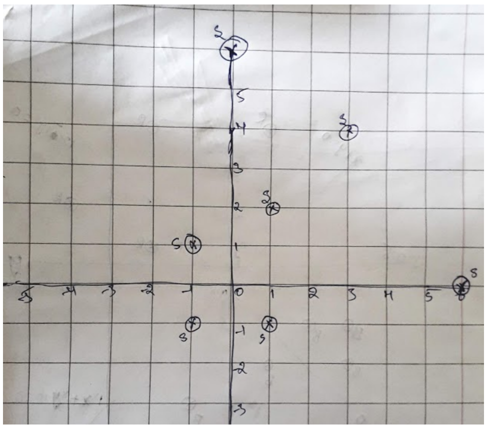
Q5: Find the closest points

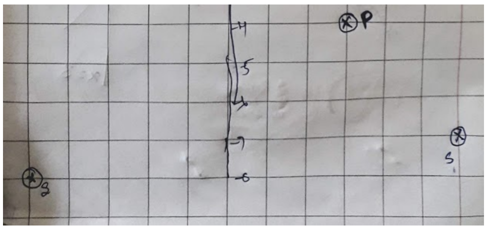
Consider you are given n data points in the form of list of tuples like S=[(x1,y1),(x2,y2),(x3,y3),(x4,y4),(x5,y5),..,(xn,yn)] and a point P=(p,q)
your task is to find 5 closest points(based on cosine distance) in S from P

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 [31]:

```
import math

# 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 S is list of tuples and P is a tuple ot len=2
def closest_points_to_p(S, P):
    # write your code here
    closest_points=[]
    distances = list(map(lambda x:math.acos((x[0]*P[0]+x[1]*P[1])/(math.sqrt(x[0]**2+x[1]**2)*math.s
qrt(P[0]**2+P[1]**2))),S))
    for i in range(5):
        min_index = distances.index(min(distances))
        closest_points.append(S.pop(min_index))
        del distances[min_index]
    return closest_points  # its list of tuples

S= [(1,2),(3,4),(-1,1),(6,-7),(0, 6),(-5,-8),(-1,-1),(6,0),(1,-1)]
P= (3,-4)

points = closest_points_to_p(S, P)
for point in points:
    print(point)
```

```
(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

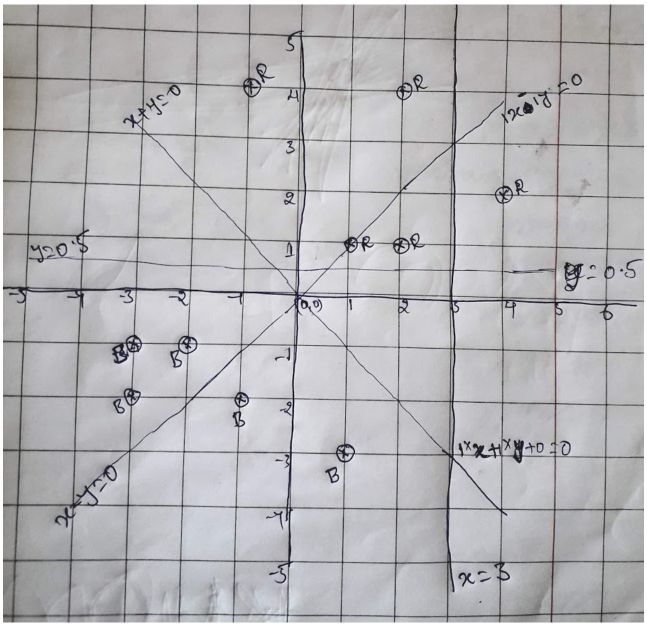
```
Red =[(R11,R12),(R21,R22),(R31,R32),(R41,R42),(R51,R52),..,(Rn1,Rn2)]
Blue=[(B11,B12),(B21,B22),(B31,B32),(B41,B42),(B51,B52),..,(Bm1,Bm2)]
```

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 intercept.
```

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 [35]:

```python
import math
# 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 i_am_the_one(red,blue,line):
    # your code
    a = float(line[:line.index('x')])
    b = float(line[line.index('x')+1:line.index('y')])
    c = float(line[line.index('y')+1:])
    if all(x>0 for x in list(map(lambda y:y[0]*a+y[1]*b+c,red))) & all  (x<0 for x in list(map(lambda y:y[0]*a+y[1]*b+c,blue))):
        return 'YES'
    else:
        return '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) # the returned value
```

```
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 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 di
stributed 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 [15]:

```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


# you can free to change all these codes/structure
def curve_smoothing(string):
    # your code
    tot = 0
    count = 0
    index = 0
    strings = string.split(',')
    l = len(strings)
    values = [0]*l
    for i in range(l):
        if strings[i]=='_':
            count += 1

        else:
            tot = (tot+float(strings[i]))/(count+1)
            for j in range(index,i):
                values[j] = tot
            count = 1 #since the last number has to be included
            index = i

    for j in range(index,l):
        values[j] = tot/(count)

    return values#list of values

inputs = ["_,_,_,24","40,_,_,_,60","80,_,_,_,_","_,_,30,_,_,_,50,_,_"]
for S in inputs:
    smoothed_values= curve_smoothing(S)
    print('input  : ',S)
    print('output : ',smoothed_values)
    print()
```

```
input  :  _,_,_,24
output :  [6.0, 6.0, 6.0, 6.0]

input  :  40,_,_,_,60
```

```
output :  [20.0, 20.0, 20.0, 20.0, 20.0]

input  :  80,_,_,_,_
output :  [16.0, 16.0, 16.0, 16.0, 16.0]

input  :  _,_,30,_,_,_,50,_,_
output :  [10.0, 10.0, 12.0, 12.0, 12.0, 12.0, 4.0, 4.0, 4.0]
```

Q8: Find the probabilities

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 [100]:

```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



# you can free to change all these codes/structure
def compute_conditional_probabilites(A):
    # your code
    str_S1 = ''
    str_S2 = ''
    str_S3 = ''
    for combs in A:
        if combs[1] == 'S1':
            str_S1 += combs[0]
        if combs[1] == 'S2':
            str_S2 += combs[0]
        if combs[1] == 'S3':
            str_S3 += combs[0]
    # print the output as per the instructions
    print('a. P(F=F1|S==S1)={}/{}, P(F=F1|S==S2)={}/{}, P(F=F1|S==S3)={}/{}'.format(str_S1.count('F
1'),len(str_S1)//2,str_S2.count('F1'),len(str_S2)//2, str_S3.count('F1'),len(str_S3)//2))
    print('b. P(F=F2|S==S1)={}/{}, P(F=F2|S==S2)={}/{}, P(F=F2|S==S3)={}/{}'.format(str_S1.count('F
2'),len(str_S1)//2,str_S2.count('F2'),len(str_S2)//2, str_S3.count('F2'),len(str_S3)//2))
    print('c. P(F=F3|S==S1)={}/{}, P(F=F3|S==S2)={}/{}, P(F=F3|S==S3)={}/{}'.format(str_S1.count('F
3'),len(str_S1)//2,str_S2.count('F3'),len(str_S2)//2, str_S3.count('F3'),len(str_S3)//2))
    print('d. P(F=F4|S==S1)={}/{}, P(F=F4|S==S2)={}/{}, P(F=F4|S==S3)={}/{}'.format(str_S1.count('F
4'),len(str_S1)//2,str_S2.count('F4'),len(str_S2)//2, str_S3.count('F4'),len(str_S3)//2))
    print('e. P(F=F5|S==S1)={}/{}, P(F=F5|S==S2)={}/{}, P(F=F5|S==S3)={}/{}'.format(str_S1.count('F
5'),len(str_S1)//2,str_S2.count('F5'),len(str_S2)//2, str_S3.count('F5'),len(str_S3)//2))

A = [['F1','S1'],['F2','S2'],['F3','S3'],['F1','S2'],['F2','S3'],['F3','S2'],['F2','S1'],['F4','S1
'],['F4','S3'],['F5','S1']]
compute_conditional_probabilites(A)
```

```
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
```

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 [105]:

```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

# you can free to change all these codes/structure
def string_features(S1, S2):
    # your code
    set1 = set(S1.split())
    set2 = set(S2.split())
    a = len(set1.intersection(set2))
    b = list(set1.difference(set2))
    c = list(set2.difference(set1))
    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)
print('b.',b)
print('c.',c)
```

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

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

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

In [113]:

```
# 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
import math

# you can free to change all these codes/structure
def compute_log_loss(A):
    # your code
    loss = 0
    for Y in A:
        loss += Y[0]*math.log10(Y[1]) + (1-Y[0])*math.log10(1-Y[1]) #looks like loss function for logistic regression
    loss = -loss/len(A)
    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
```