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 [1]:
```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 examples


# you can free to change all these codes/structure
# here A and B are list of lists

#3*3 matrix
A = [[1,2,3],
    [4,5,6],
    [7,8,9]]
#3*3 matrix
B = [[1,0,0],
    [0,1,0],
    [0,0,1]]
#define function for matrix multiplication
def Matrix_Mul(M, N):
    #mapped all the iterations of A and B matrices by using Zip()
    # First, zipped multiplication of M_row and N_col
    # Second, summation of the zip by coloum wise(N_clol)
    #https://www.geeksforgeeks.org/zip-in-python/
    # Finally all the iterations have been stored in the 'result'
    result = [[sum(x*y for x,y in zip(M_row,N_col)) for N_col in zip(*N)] for M_row in M]
    #printing all the values
    for i in result:
        print(i)
#checks for the possible multiplications by using length
if (len(A)<=len(B)):
    Matrix_Mul(A, B)
#prints if multipllication not possible
else:
    print('Not Possible')
```

```
[1, 2, 3]
[4, 5, 6]
[7, 8, 9]
```

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 [70]:
```python
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

#https://www.geeksforgeeks.org/python-select-random-value-from-a-list/
#picking an element from with the probability propotional to its magnitude
A = [0,5,27,6,13,28,100,45,10,79]
# you can free to change all these codes/structure
def pick_a_number_from_list(A):
    sum = 0
    for i in A:
        sum += i
    lst = []
    for i in range(len(A)):
        lst.append(A[i]/sum)
    lst_1 = []
    lst_1.append(lst[0])
    for i in range(1, len(lst)):
        lst_1.append(lst_1[i-1]+lst[i])
    num = uniform(0.0,1.0)
    for i in range(len(lst_1)):
        if num <= lst_1[i]:
            return A[i]

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

sampling_based_on_magnitued()
```

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

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]:
```python
#importing regular expressions
import re
#string input values with digits and symbols
s = "#2a$#b%c%561#"
#collect only digits using join
t = ''.join(i for i in s if i.isdigit())
print(t) # print only digits

def replace_digits():
    temp = re.sub("\d","#", t) #replace digits with # using 're.sub'
    print (temp)

replace_digits()
```

```
2561
####
```

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','stude
nt8','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 [47]:
```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 examples

# you can free to change all these codes/structure
mydict = {}
def display_dash_board(students, marks):
    for i in range(len(students)):
        mydict[marks[i]]=students[i]
    marks.sort(reverse=True)
    # write code for computing top top 5 students
    top_5_students = marks[:5]
    # write code for computing top least 5 students
    marks.sort()
    least_5_students = marks[:5]
    # write code for computing students within 25 and 75
    if len(marks)%4 == 0:
        from_25 = len(marks)/4    #gives 25th after values
        to_75 = (len(marks)/4)*3  #gives 75th before values
    else:
        from_25 = (len(marks))//4  #gives 25th after values
        to_75 = (len(marks)*3)//4  #gives 75th before values
    marks.sort()
    students_within_25_and_75 = marks[from_25:to_75]

    return top_5_students, least_5_students, students_within_25_and_75

students=['student1','student2','student3','student4','student5','student6','student7','student8','s
tudent9','student10']
marks = [45, 78, 12, 14, 48, 43, 47, 98, 35, 80]
#display_dash_board(students,marks)
top_5_students, least_5_students, students_within_25_and_75 = display_dash_board(students, marks)
print("TOP 5 STUDENTS")
for i in top_5_students:
    print("{} {}".format(mydict[i],i))
print("------------------------------------")
print("LEAST 5 STUDENTS")
for i in least_5_students:
    print("{} {}".format(mydict[i],i))
print("------------------------------------")
print("STUDENTS WITHIN 25 AND 75")
for i in students_within_25_and_75:
    print("{} {}".format(mydict[i],i))
print("------------------------------------")
```

```
TOP 5 STUDENTS
student8 98
student10 80
student2 78
student5 48
student7 47
------------------------------------
LEAST 5 STUDENTS
student3 12
student4 14
student9 35
student6 43
student1 45
------------------------------------
STUDENTS WITHIN 25 AND 75
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 [26]:
```python
import math

def closest_points_to_p(S, P):
    for i in S:
        result = math.acos((i[0]*P[0]+i[1]*P[1])/(math.sqrt(i[0]**2+i[1]**2)*math.sqrt(P[0]**2+P[1]*
*2))) #results the given formulae
        dict[result] = i  #result stored in the dictionary
    closest_points_to_p = []
    for i in sorted(dict.keys()):
        closest_points_to_p.append(dict[i])
    return closest_points_to_p[:5] #prints first 5 key points

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)
print(points)
```

```
[(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
import re # importing regular expressions to find coefficients
#function to get coefficient of lines
def line_coefficient(lines):
    co = []
    temp = []
    z = re.findall(r"[\d\.\-\+]+", lines) #convert to coefficient from alphanumerics and special cha
racters
    temp.append(z)
    for j in range(len(temp[0])):
        a=list(map(float , temp[j]))
        co.append(a)
        return co # returns coefficient of a line


# function to get yes/no with the line
def i_am_the_one (red, blue, line):
    COL=line_coefficient(line) #coefficient of line
    ro=[]
    for i in red:
        e=(COL[0][0]*i[0])+(COL[0][1]*i[1])-COL[0][2] #checks if the line seperate red points
        ry="yes"
        rn="no"
        if e > 0:
            ro.append(ry)
        else:
            ro.append(rn)
    bo=[]
    for j in blue:
        l=(COL[0][0]*j[0])+(COL[0][1]*j[1])-COL[0][2] #checks if the line seperate blue points
        by="yes"
        bn="no"
        if l>0:

            bo.append(by)
        else:
            bo.append(bn)
    fo=list(zip(ro,bo)) #list of ro and bo
    count=0
    for p in range(len(fo)):
        if fo[p]==('yes','no') and ("no","yes"): #checks for ro and bo
            count +=1
    if count==len(fo):
        return print("yes") #returns yes if count matches len(fo)
    if count!=len(fo):
        return print("no")#returns no if count matches len(fo)

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:
    i_am_the_one(Red, Blue, i)
```

```
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, 2
0, 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 dist
ributed 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 [61]:
```python
# https://stackoverflow.com/questions/57179618/filling-the-missing-values-in-the-specified-format-py
thon/57180882
def curve_smoothing(string):
    a = string.split(',')
    count = 0
    middle_store = 0
    first = 0
    lst = []
    start = 0
    # left
    for i in range(len(a)):
        if a[i] == '_':
            count = count + 1  # find number of blanks to the left of a number
        elif count>0:
            middle_store = int(a[i])
            middle_store = (first+middle_store)//(count+1)
            for j in range(start,start+count+1):
                a[j]=middle_store
            start = start +count
            count = 0
            i = i-1
        else:
            first = int(a[i])
            count = count+1
        if i ==len(a)-1:
            if middle_store==0:
                middle_store = (first+middle_store)//(count)
                for j in range(start,len(a)):
                    a[j]=middle_store
            else:
                middle_store=(first+middle_store)//(count+1)
                for j in range(start+1, len(a)):
                    a[j] = middle_store
    return a

A = "_,_,_,24"
B = "40,_,_,_,60"
C = "80,_,_,_,_"
D = "_,_,30,_,_,_,50,_,_"

a = curve_smoothing(A)
b = curve_smoothing(B)
c = curve_smoothing(C)
d = curve_smoothing(D)

print(a)
print(b)
print(c)
print(d)
```

```
[6, 6, 6, 6]
[20, 20, 20, 20, 20]
[16, 16, 16, 16, 16]
[10, 10, 12, 12, 12, 12, 4, 4, 4]
```

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 [9]:
```python
#https://www.sitepoint.com/community/t/how-to-find-the-probabilities-of-a-list-of-lists-using-python
-and-without-any-libraries/333461
Dict_1 = {
 'F1S1':0,
 'F2S1':0,
 'F3S1':0,
 'F4S1':0,
 'F5S1':0,
 'F1S2':0,
 'F2S2':0,
 'F3S2':0,
 'F4S2':0,
 'F5S2':0,
 'F1S3':0,
 'F2S3':0,
 'F3S3':0,
 'F4S3':0,
 'F5S3':0,
}  #storing combinations as keys and values in the dictionary

Dict_2= {
 'S1':0,
 'S2':0,
 'S3':0
}  #another dictionary for column S

def compute_conditional_probabilities(A):
    for i in range(len(A)): #selects each values
        k = A[i][0]+A[i][1] #stores combinations
        Dict_1[k] += 1     #creates dictionary for combinations
        Dict_2[A[i][1]] += 1 #creates dictionary for S column

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

compute_conditional_probabilities(A)

print('Probability of P(F=F1|S==S1)',(Dict_1['F1S1']/Dict_2['S1'])) #prints the probability of selec
ted values
```

```
Probability of P(F=F1|S==S1) 0.25
```

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 [27]:
```python
#https://www.geeksforgeeks.org/linq-set-operator-intersect/

def string_features(S1, S2):
    S1_words = S1.split()  #splitting sentence 1
    S2_words = S2.split()  #splitting sentence 2

    a = set(S1_words).intersection( set(S2_words) )  #prints common words
    b = set(S1_words) - ( set(S2_words) )        #prints s1 - s2
    c = set(S2_words) - ( set(S1_words) )        #prints s2 - s1

    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(len(a),b,c)
```

```
7 {'F', 'first', '5'} {'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 $f(Y, Y_{score}) = -1 * \frac{1}{n}\Sigma_{foreachY,Y_{score}pair}(Ylog10(Y_{score}) + (1-Y)log10(1-Y_{score}))$ 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:
0.44982
```