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]
             [23 30 37 44 51]]

Ex 3: A   = [[1 2]
             [3 4]]
      B   = [[1 4]
             [5 6]
             [7 8]
             [9 6]]
      A*B =Not possible
```

In [ ]:
```python
def matrix_mul(A,B):
    '''This function prints the product of two matrices if possible'''

    row1=len(A)       #Number of rows in first matrix
    col1=len(A[0])    #Number of columns in first matrix
    row2=len(B)       #Number of rows in second matrix
    col2=len(B[0])    #Number of columns in second matrix

    if (col1==row2)==False:
        print("The product of matrices is not possible")


    z=[]           #replacement matrix /Additive Identity for matrices
    while len(z)<row1:
        y=[]
        while len(y)<col2:
            y.append(0)
        z.append(y)

    for i1 in range(row1): #r
        for i2 in range(col2): #index of each list in b
            for x in range(row2): #As len(col1)=len(row2),we can use either of them
                z[i1][i2]+=A[i1][x]*B[x][i2]   #0s will be replaced with values


    return (f"The dot product of the matrices entered is:{z}")

A   = [[1,3,4],
       [2,5,7],
       [5,9,6]]


B   = [[1,0,0],
       [0,1,0],
       [0,0,1]]

matrix_mul(A,B)
```

Q2: Proportional Sampling - 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 [ ]:
```python
def sampling_based_on_magnitued(A):

    '''This function returns the sampling of the elements of the list based on the magnitude'''

    import random

    S=sum(A)
    L=[i/S for i in A] #Normalising values//list comprehension

    X=[L[0]]           #Getting the cumulating sum

    for i in range(1,len(L)):
         X.append(X[-1]+L[i])   #[1]:0+0.01597444089456869 [2]:0.01597444089456869+0.08626198083067092...

    def pick_a_number_from_list(A):
            r=random.uniform(0,1)
            for i in range(len(X)):
                if r<=X[i]:
                    return A[i]

    for i in range(1,100):
            number = pick_a_number_from_list(A)    #printing samples based on magnitude
            print(number)

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

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 [ ]:
```python
String="21ee55wewe"

def replace_digits(String):
        '''This function removes all the not digits and replace the digits with #'''

        import re

        x=re.findall('\d',String)   #Extracting the digits from the String
        x=''.join(x)
        y=re.sub('\d','#',x)      #converting those digits into #

        return(print(y))

replace_digits(String)

#https://www.youtube.com/watch?v=K8L6KVGG-7o&t=971s
```

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, 22, 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, 22, 80]
a.
student8  98
student10 80
student2  78
student5  48
student7  47
b.
student3 12
student4 14
student9 22
student6 43
student1 45
c.
student9 22
student6 43
student1 45
student7 47
student5 48
```

In [ ]:
```python
def display_dash_board(Students,Marks):

            '''This function desplays the DashBoard of students rankings with their respective marks'''

            J=list(zip(Students,Marks))     #creating and ziping a list containing tuples of Students and their marks
            J=sorted(J,key=lambda ext:ext[1])     #1

            print("The top 5 students with their respective marks:")
            for Student,Marks in J[:4:-1]:
                print(Student,': ',Marks)
            print("\nThe least 5 students with their respective scores marks:")
            for Student,Marks in J[:5]:
                print(Student,': ',Marks)

            twentyFifth= 25//len(J)
            seventyFifth= 75//len(J)

            print("\nThe students who got marks between >25th percentile <75th percentile are:")
            for Student,Marks in J[twentyFifth:seventyFifth]:
                print(Student,': ',Marks)


Students=['student1','student2','student3','student4','student5','student6','student7','student8','student9','student10']
Marks = [45, 78, 12, 14, 48, 43, 47, 98, 22, 80]
display_dash_board(Students,Marks)

#1 : the iterables in sorted() are (student1,45),(student2,78)....
#    so for taking second elemnt of each itr i,e element at ind[1],   we defined function ext()
#    and thus so using it's values as keys(sort the values according to keys)


#https://stackoverflow.com/questions/3121979/how-to-sort-a-list-tuple-of-lists-tuples-by-the-element-at-a-given-index
```

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)
</pre>
 Hint - If you write the formula correctly you'll get the distance between points (6,-7) and (3,-4) = 0.065
```

In [ ]:
```python
#Hint - If you write the formula correctly you'll get the distance between points (6,-7) and (3,-4) = 0.065

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


def closest_points_to_p(S, P):
                    '''This function writes the closest points(based on distance) in S from p'''

                    import math

                    p=P[0]
                    q=P[1]
                    PSq=((p**2)+(q**2))**0.5

                    lst1=[]
                    lst2=[]
                    for x,y in S:
                        lst1.append(x*p+y*q)
                        lst2.append(((x**2+y**2)**0.5)*PSq)

                    lst3=[(math.acos(lst1[ind]/lst2[ind]))for ind in range(len(lst1))]


                    #As per the hint we found that lst3[itr] is distance of (x,y) and (p,q)
                    #Our task is to find 5 closest points(based on cosine distance) in S from P i.e,  The distance should be lesser

                    Q=zip(S,lst3)
                    Q=sorted(Q,key=lambda X:X[1])
                    print("The Five closest points are:")
                    for dist in Q[:5]:
                        print(dist[0])
closest_points_to_p(S,P)


#https://www.delftstack.com/howto/python/inverse-cosine-in-python/
```

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 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 [ ]:
```python
def i_am_the_one(red,blue,line):
    '''This code checks whether the line'''

    import re

    #As the equations in Lines are in same syntax ,we can extract values for each variables from each equation i.e, a,b,c with regex
    lines1=[re.findall('[\d.+-]+',line) for line in Lines]

    red=[]
    green=[]

    for ind in range(len(lines1)):
            line1=[]
            for x,y in Red:
                line1.append(float(lines1[ind][0])*x+float(lines1[ind][1])*y+float(lines1[ind][2]))
                line2=[]
            for x,y in Blue:
                line2.append(float(lines1[ind][0])*x+float(lines1[ind][1])*y+float(lines1[ind][2]))
            red.append(line1)
            green.append(line2)

    for ind in range(len(lines1)):
            print("line",ind+1)
            if all(i>0 for i in red[ind]) and all(i<0 for i in green[ind]):  #1
                print("YES \n")
            else:
                print("NO \n")


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"]
i_am_the_one(Red,Blue,Lines)


#1: https://www.geeksforgeeks.org/python-check-if-all-elements-in-list-follow-a-condition/
```

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 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 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, 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 [ ]:
```python
def curve_smoothing(S):
            '''This function fills the missing values in specified format'''
            import re

            x=S.split(",")
            S=[0 if i=="_" else float(i) for i in x ]

            if all(ele==S[0] for ele in S):
                print("Action is not possible on given string")
                return S
            else:
                #1. first we will distribute the 30 to left two missing values (10, 10, 10, _, _, _, 50, _, _)
                count=0
                for i in range(len(S)):
                    if S[i]==0:
                        count+=1
                    else:
                        count+=1
                        x=S[i]/count
                        for j in (range(0,count)):
                            S[j]=x
                            ind=i
                        break

                Add=(S[i])
                count=1
                #2: now distribute the sum (10+50) missing values in between (10, 10, 12, 12, 12, 12, 12, _, _)
                for i in range(ind+1,len(S)):
                    if S[i]==0:
                        count+=1
                    else:
                        count+=1
                        x=((Add+S[i])/count)
                        for j in (range(ind,i+1)):
                            S[j]=x
                            ind=i

                #3: now we will distribute 12 to right side missing values (10, 10, 12, 12, 12, 12, 4, 4, 4)
                count=1
                Add=S[ind]
                for i in range(ind+1,len(S)):
                    if S[i]==0:
                        count+=1
                for i in range(ind,len(S)):
                    x=Add/count
                    S[i]=x

                return S

S="_,_,_,24"
smoothed_values= curve_smoothing(S)
print(smoothed_values,"\n")
S="40,_,_,_,60"
smoothed_values= curve_smoothing(S)
print(smoothed_values,"\n")
S="80,_,_,_,_"
smoothed_values= curve_smoothing(S)
print(smoothed_values,"\n")
S="_,_,30,_,_,_,50,_,_"
smoothed_values= curve_smoothing(S)
print(smoothed_values,"\n")
S=  "7,7,7,7,7,7,7,7"
smoothed_values= curve_smoothing(S)
print(smoothed_values)
```

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,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 [ ]:
```python
def compute_conditional_probabilities(A):
            '''this function computes the conditional probabilities'''

            F=sorted(list(set([A[i][0] for i in range(len(A))])))
            S=sorted(list(set([A[i][1] for i in range(len(A))])))

            p1=[]
            for i in S:
                count=0
                for j in range(len(A)):
                    if i in A[j]:
                        count+=1
                p1.append(count)

            c=[]
            for i in F:
                for j in S:
                    count=0
                    for k in range(len(A)):
                        if i in A[k]:
                            if j in A[k]:
                                count+=1
                    c.append(count)

            p2=[c[i:i+len(S)] for i in range(0, len(c), len(S))]

            for i in range(len(p2)):              #1
                for j in range(len(S)):
                    print(f"P(F=={F[i]}|S=={S[j]})={p2[i][j]}/{p1[j]}")
                if i!=len(p2)-1:
                    print(",")

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)

#1: https://datagy.io/python-split-list-into-chunks/
```

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 [ ]:
```python
def string_features(S1, S2):
    '''This function computes:
       a. Number of common words between S1, S2
       b. Words in S1 but not in S2
       c. Words in S2 but not in S1'''

    x=set(S1.split())
    y=set(S2.split())   #converted to set() because one word from one sentence can be repeated multiple times in other

    count=0
    for i in x:
        if i in y:
            count+=1
    a=(f"There are {count} number of common words between S1,S2")

    inS1=[]
    for i in x:
        if i not in y:
            inS1.append(i)

    inS2=[]
    for i in y:
        if i not in x:
            inS2.append(i)

    b,c=inS1,inS2
    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(f"{a}\n\nWords in S1 but not in S2 {b}\nWords in S2 but not in S1 {c}")
```

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
```

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

In [ ]:
```python
def compute_log_loss(A):
        '''This function computes the value of  f(Y,Y_score)'''
        import math

        n=len(A)
        x=0

        for Y,Y_score in A:
            x+=((Y*math.log10(Y_score)+(1-Y)*math.log10(1-Y_score)))   #1

        loss= -x/n

        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)
#1:   https://www.geeksforgeeks.org/log-functions-python/
```