Python Mandotary Assignment : Without Numpy & Sklearn

Q 1) Print the product of two matrices which is given.

```
A = [[1,2,3], # Matrix A
     [4,5,6],
     [7,8,9]]
B = [[9,8,7], # Matrix B
     [6,5,4],
     [3,2,1]]

C = [[0, 0, 0], # Result Matrix
    [0, 0, 0],
    [0, 0, 0]]

for i in range(len(A)): # Iterating by rows of A
    for j in range(len(B[0])): # Iterating by column of B
        for k in range(len(B)): # Iterating by rows of B
            C[i][j] += A[i][k] * B[k][j] # Multiplying Both Matrix
(3x3)
for r in C:
    print(r)
```

```
[60, 48, 36]
[168, 138, 108]
[276, 228, 180]
```

Q2: Proportional Sampling - Select a number randomly with probability proportional to its magnitude from the given array of n elements.

```
import random
A = [1,2,3,4,5,6]

#Sum of all the elements in the array
S = sum(A)

#Calculating normalized sum
norm_sum = [ele/S for ele in A]

#Calculating cumulative normalized sum
cum_norm_sum = []
cum_norm_sum.append(norm_sum[0])
for itr in range(1, len(norm_sum), 1) :
   cum_norm_sum.append(cum_norm_sum[-1] + norm_sum[itr])

def prop_sampling(cum_norm_sum) :
    r = random.random()
    for itr in range(len(cum_norm_sum)) :
       if r <  cum_norm_sum[itr] :
           return A[itr]
```

```
#Sampling 100 elements from the given list with proportional sampling
sampled_elements = []
for itr in range(100) :
   sampled_elements.append(prop_sampling(cum_norm_sum))

C = (sorted(A, reverse=True))
print(C)
for ele in C:
    print("number:{0}, frequency :{1}".format(ele,
sampled_elements.count(ele)))
```

```
[6, 5, 4, 3, 2, 1]
number:6, frequency :30
number:5, frequency :24
number:4, frequency :19
number:3, frequency :8
number:2, frequency :9
number:1, frequency :10
```

Q3: Replace the digits in the string with '#'

```
test_str = '234, a2b3c4, abc, #2a$#b%c%561#'

print('The Original string is : ' + str(test_str))

k = '#'

for ele in test_str:
    if ele.isdigit():  # Checking digits in the string
        test_str = test_str.replace(ele, k) # replacing all digits in
string

print("the resultant string : "+ str(test_str))
```

```
The Original string is : 234, a2b3c4, abc, #2a$#b%c%561#
the resultant string : ###, a#b#c#, abc, ##a$#b%c%####
```

Q4: 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.

c) Who got marks between >25th percentile <75th percentile, in the increasing order of marks.

```
Students=['student1','student2','student3','student4','student5','stud
ent6','student7','student8','student9','student10']
Marks = [45, 78, 12, 14, 48, 43, 47, 98, 22, 80]

Increasing_Marks = sorted(Marks) # First sorting marks in increasing
order
```

```
print("Marks in Increasing Order : ", Increasing_Marks)

Decreasing_Marks = sorted(Marks, reverse = True) # Sorting marks in
decreasing order
print("Marks in Decreasing Order : ", Decreasing_Marks)

print('---Answer of first part of question.---')
print("Student got Top 5 rank in the Descending Order :
",Decreasing_Marks[:5])

print('---Answer of Second part of question.---')
print("Student got least 5 rank in the Increasing Order :
",Increasing_Marks[:5])

import math

def percentile(Marks, percentile):
    size = len(Marks)
    return sorted(Marks)[int(math.ceil((size * percentile) / 100)) -
1]

p25 = percentile(Marks, 25)
p75 = percentile(Marks, 75)

print('---Answer of Third part of question.---')

print('25th and 75th percentile:',p25,p75)
print('Marks between 25th and 75th percentile:')
for i in Increasing_Marks:
    if i > 22 and i < 78: # here 22, 78 are 25th, 75th percentile
respectively.
        print(i)
```

```
Marks in Increasing Order :  [12, 14, 22, 43, 45, 47, 48, 78, 80, 98]
Marks in Decreasing Order :  [98, 80, 78, 48, 47, 45, 43, 22, 14, 12]
---Answer of first part of question.---
Student got Top 5 rank in the Descending Order :  [98, 80, 78, 48, 47]
---Answer of Second part of question.---
Student got least 5 rank in the Increasing Order :  [12, 14, 22, 43,
45]
---Answer of Third part of question.---
25th and 75th percentile: 22 78
Marks between 25th and 75th percentile:
43
45
47
48
```

Q5: Find 5 closest points(based on cosine distance) in S from P

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

P= (3,-4)

```
import math

x = []
value = [(1, 2), (3, 4), (-1, 1), (6, -7), (0, 6), (-5, -8), (-1, -1),
(6, 0), (1, -1)]
for a,b in value:
    numr = a*3 + b*(-4)
    denm = math.sqrt(a**2 + b**2) * math.sqrt(3**2 + (-4)**2)
    x.append(math.acos(numr/denm))

Y = [S for S in sorted(zip(S,X), key = lambda i: i[1])]
k = Y[:5] # first 5 values from starting
for i, j in k:
    print(i) #printing first 5 closest point from P(3,-4)
```

```
(6, -7)
(1, -1)
(6, 0)
(-5, -8)
(-1, -1)
```

Q6: Find Which line separates oranges and apples

Task is 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

```
import math

def line_seperator(red,blue,line):
    #code for Red
    for i in red:
        eq=line.replace('x','*'+str(i[0]))
        eq=eq.replace('y','*'+str(i[1]))
        answer=eval(eq)
        if answer>0:
            pass
        else:
            return "NO"

    # Code for Blue
    for j in blue:
        eq1=line.replace('x','*'+str(j[0]))
        eq1=eq1.replace('y','*'+str(j[1]))
        answer1=eval(eq1)
        if answer1<0:
            pass
        else:
            return "NO"
    return "Yes"

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 = line_seperator(Red, Blue, i)
    print(yes_or_no)
```

```
Yes
NO
NO
Yes
```

Q7: Filling the missing values in the specified formate

```
def replace(string):
    lst=string.split(',')
    for i in range(len(lst)):
        if lst[i].isdigit():
            for j in range(i+1):
                lst[j]=int(lst[i])//(i+1)
            new_index=i
            new_value=int(lst[i])
            break
    for i in range(new_index+1,len(lst)):
        if lst[i].isdigit():
            temp=(new_value+int(lst[i]))//(i-new_index+1)
            for j in range(new_index,i+1):
                lst[j]=temp
            new_index=i
            new_value=int(lst[i])
    try:
        for i in range(new_index+1,len(lst)):
            if not(lst[i].isdigit()):
                count=lst.count('_')
                break
        temp1=new_value//(count+1)
        for i in range(new_index,len(lst)):
            lst[i]=temp1
    except:
        pass
    return lst

string =[
    "_,_,_,24",
    "40,_,_,_,60",
    "80,_,_,_,_",
     "_,_,30,_,_,_,50,_,_"]
for i in string:
    print (replace(i))
```

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

Q8: 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)

```
A = [['F1','S1'],['F2','S2'],['F3','S3'],['F1','S2'],['F2','S3'],
['F3','S2'],['F2','S1'],['F4','S1'],['F4','S3'],['F5','S1']]
from fractions import Fraction
def values(F,S):
    num=0
    den=0
    for i in range(len(A)):
        if(A[i][1]==S):
            den=den+1
            if(A[i][0]==F):
               num=num+1
    print('P(F={}|S=={})='.format(F,S), Fraction(num,den))

find_F = ['F1', 'F2', 'F3', 'F4', 'F5']
find_S = ['S1', 'S2', 'S3']
for k in find_F:
    for m in find_S:
        values(k,m)
```

```
P(F=F1|S==S1)= 1/4
P(F=F1|S==S2)= 1/3
P(F=F1|S==S3)= 0
P(F=F2|S==S1)= 1/4
P(F=F2|S==S2)= 1/3
P(F=F2|S==S3)= 1/3
P(F=F3|S==S1)= 0
P(F=F3|S==S2)= 1/3
P(F=F3|S==S3)= 1/3
P(F=F4|S==S1)= 1/4
P(F=F4|S==S2)= 0
```

```
P(F=F4|S==S3)= 1/3
P(F=F5|S==S1)= 1/4
P(F=F5|S==S2)= 0
P(F=F5|S==S3)= 0
```

Q9: Given two sentances S1, S2 , 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

```
S1= "the first column F will contain only 5 uniques values"
S2= "the second column S will contain only 3 uniques values"

w1 = S1.split()
w2 = S2.split()

commonwords = []
for word in w1:
    if (word in w2):
        commonwords.append(word)

print(commonwords) # print commomn words between S1 & S2
print(len(commonwords)) # print length of commomn words

def uncommon_words(S1):
    count = {}
    for word in S1.split():
        count[word] = count.get(word, 0) + 1
    # words of string s2
    # return required list of words
    for word in S2.split():
        count[word] = count.get(word, 0) + 1

    return [word for word in count if count[word] == 1]

# Print required answer
print(uncommon_words(S1))
```

```
['the', 'column', 'will', 'contain', 'only', 'uniques', 'values']
7
['first', 'F', '5', 'second', 'S', '3']
```

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 $f(Y, Y_{score}) = -\frac{1*1}{n} \Sigma_{foreachY, Y_{score} pair}(Y log 10(Y_{score}) + (1-Y) log 10(1-Y_{score}))$ 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

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

```
from math import *
def comp_log_loss(A):
    sum = 0
    for i in A:
        sum+=i[0]*log10(i[1])+(1-i[0])*log10(1-i[1])
    loss=(-1/len(A))*sum
    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]]
log_loss= comp_log_loss(A)
print(log_loss)
```

```
0.42430993457031635
```