1.Python Assignment

April 20, 2020

.

Python: without numpy or sklearn

Q1: Given two matrices please print the product of those two matrices

[6]:
```
# 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
    """This function returns the matrix multiplication of two matrices."""
    if(len(A[0])!=len(B)):
        return 'Not possible'
    else:
        output=[]
        for m in range(len(A)):
            sam=[]
            for q in range(len(B[0])):
                temp=0
                for n in range(len(B)):
                    temp+=(A[m][n]*B[n][q])
                sam.append(temp)
            output.append(sam)
        else:
            return output

A = [[1,3,4],[2,5,7],[5,9,6]]
B = [[1,0,0],[0,1,0],[0,0,1]]
print(matrix_mul(A,B))
```

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

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.

[7]:
```
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

# 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

    prob=[]
    for i in A:                    # normalization of numbers by sum of list.
        prob.append(i/sum(A))

    for i in range(1,len(prob)):
        prob[i]=prob[i]+prob[i-1] ## cumulative sum..

    sample=uniform(0,1)
    for i in range (len(prob)):
        if sample<prob[i]:
            return A[i]



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

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

```
100
79
100
79
79
28
13
79
10
28
100
```

79
100
79
45
100
100
45
45
45
100
45
100
28
79
100
45
100
45
28
45
79
79
100
79
27
79
79
100
27
45
79
100
100
100
100
79
79
79
28
28
28
100
100
27
100
27
45
100

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

[8]:
```
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

    m=re.findall("\d",string)
    str_new='#'*len(m)
    return str_new     # modified string which is after replacing the # with␣
 ↪digits

String='#2a$#b%c%561#'
print(replace_digits(String))
```

####

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

[9]:
```
# 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

students=['student1','student2','student3','student4','student5','student6','student7','studen
marks = [45, 78, 12, 14, 48, 43, 47, 98, 35, 80]


# you can free to change all these codes/structure
def display_dash_board(students, marks):
    # write code for computing top top 5 students

    dashboard=[(students[i],marks[i]) for i in range(len(marks))]
    dashboard=dict(dashboard)
    tot=max(marks)

    lst2={k: v for k, v in sorted(dashboard.items(), key=lambda item: item[1],␣
 ↪reverse=True)}
    lst={}
```

```
    count=0
    for k,v in lst2.items():
        lst[k]=v
        count+=1
        if(count==5):
            break;

    top_5_students=lst

    lst2={k: v for k, v in sorted(dashboard.items(),key=lambda item: item[1],␣
 ↪reverse=False)}
    lst={}
    count=0
    for k,v in lst2.items():
        lst[k]=v
        count+=1
        if (count==5):
            break;
    least_5_students=lst

    students_within_25_and_75 = {k: v for k, v in sorted(dashboard.items(),key␣
 ↪=lambda item: item[1])
                                 if (v > 0.25*tot and v<0.75*tot)}

    ## sorting based on value so we change the key to value.

    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 k,v in top_5_students.items():
    print(k , v)
print('b.')
for k,v in least_5_students.items():
    print(k , v)
print('c.')
for k,v in students_within_25_and_75.items():
    print(k , v)
```

```
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 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 $cos^{-1}(\frac{(x\cdot p+y\cdot q)}{\sqrt{(x^2+y^2)}\cdot\sqrt{(p^2+q^2)}})$

[5]:
```
from math import acos,sqrt

# 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 dist(x,y):
    return ((x[0]*y[0])+(x[1]*y[1]))/
 ↪(sqrt((x[0]**2)+(x[1]**2))*sqrt((y[0]**2)+(y[1]**2)))

# 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

    lst={}
    for i in range(len(S)):
        lst.update({i: acos(dist(S[i],P))})

    lst={k:v for k,v in sorted(lst.items(), key = lambda item: item[1],reverse=␣
 ↪False)}

    closest_points_to_p=lst.keys()

    return closest_points_to_p  # 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 = list(closest_points_to_p(S, P))
for i in range(5):
    print(S[points[i]]) #print the returned values
```

```
(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

and set of line equations(in the string formate, i.e list of strings)

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

[6]:
```
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

    dist_red={}
    equ=''
    for i in range(len(line)):
        if(line[i]!='x' and line[i]!='y' and line[i]!='+' and line[i]!='-'):
            equ+=line[i]
            continue;
        elif(line[i]=='x'):
            equ+='*x'
            continue;
        elif(line[i]=='y'):
            equ+='*y'
            continue;
        else:
            equ+=line[i]

    expr=equ
    for i in range(len(red)):
        (x,y)=(red[i][0],red[i][1])
```

```
        dist=eval(expr)
        dist=True if dist>0 else False
        dist_red.update({i:dist})

    dist_blue={}
    for i in range(len(blue)):
        (x,y)= (blue[i][0],blue[i][1])
        dist=eval(expr)
        dist=True if dist>0 else False
        dist_blue.update({i:dist})

    if ((all(dist_red.values()) and all([filter(lambda x: x>0 , dist_blue.
↪values())])) or
        (all([filter(lambda x : x<0, dist_red.values())]) and all(dist_blue.
↪values()))):
        return 'YES'
    else:
        return 'NO'
    #yes/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 formate

You will be given a string with digits and '_'(missing value) symbols you have to replace the '_' symbols as explained

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:

[7]:
```
# 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

    lst=string.split(',')

    lst2=[]
    count=0
    tot=0
    last=0
    for i in range(len(lst)):
        if lst[i]=='_' and i!=(len(lst)-1):
            count+=1
            continue;
        elif (lst[i]!='_' and i==0):

            tot+=int(lst[i])
            count+=1
            continue;
        else:
            count+=1
            if (lst[i]!='_'):
                tot+=int(lst[i])

            for j in range(last,i):
                lst2.append(str((tot)//count))
            else:
                last=i
                tot=tot//count
                count=1

    lst2.append(lst2[len(lst2)-1])
    lst2=','.join(lst2)
    return lst2#list of values

S= str(input("Enter the input string. \n"))
smoothed_values= curve_smoothing(S)
print(smoothed_values)
```

```
Enter the input string.
_,_,30,_,_,_,50,_,_
10,10,12,12,12,12,4,4,4
```

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)

Ex:

[8]:
```
# 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
    S1tot=S2tot=S3tot=0
    S1F1tot=S1F2tot=S1F3tot=S1F4tot=S1F5tot=0
    S2F1tot=S2F2tot=S2F3tot=S2F4tot=S2F5tot=0
    S3F1tot=S3F2tot=S3F3tot=S3F4tot=S3F5tot=0
    for i in range(len(A)):
        S1tot+=1 if (A[i][1]=='S1') else 0
        S2tot+=1 if (A[i][1]=='S2') else 0
        S3tot+=1 if (A[i][1]=='S3') else 0
        S1F1tot+=1 if (A[i][0]=='F1' and A[i][1]== 'S1') else 0
        S1F2tot+=1 if (A[i][0]=='F2' and A[i][1]== 'S1') else 0
        S1F3tot+=1 if (A[i][0]=='F3' and A[i][1]== 'S1') else 0
        S1F4tot+=1 if (A[i][0]=='F4' and A[i][1]== 'S1') else 0
        S1F5tot+=1 if (A[i][0]=='F5' and A[i][1]== 'S1') else 0
        S2F1tot+=1 if (A[i][0]=='F1' and A[i][1]== 'S2') else 0
        S2F2tot+=1 if (A[i][0]=='F2' and A[i][1]== 'S2') else 0
        S2F3tot+=1 if (A[i][0]=='F3' and A[i][1]== 'S2') else 0
        S2F4tot+=1 if (A[i][0]=='F4' and A[i][1]== 'S2') else 0
        S2F5tot+=1 if (A[i][0]=='F5' and A[i][1]== 'S2') else 0
        S3F1tot+=1 if (A[i][0]=='F1' and A[i][1]== 'S3') else 0
        S3F2tot+=1 if (A[i][0]=='F2' and A[i][1]== 'S3') else 0
        S3F3tot+=1 if (A[i][0]=='F3' and A[i][1]== 'S3') else 0
        S3F4tot+=1 if (A[i][0]=='F4' and A[i][1]== 'S3') else 0
        S3F5tot+=1 if (A[i][0]=='F5' and A[i][1]== 'S3') else 0

    print('a. P(F=F1|S==S1)=',S1F1tot/S1tot, ' P(F=F1|S==S2)=', S2F1tot/S2tot,␣
 ↪'P(F=F1|S==S3)=',S3F1tot/S3tot)
    print('b. P(F=F2|S==S1)=',S1F2tot/S1tot, ' P(F=F2|S==S2)=', S2F2tot/S2tot,␣
 ↪'P(F=F2|S==S3)=',S3F2tot/S3tot)
    print('c. P(F=F3|S==S1)=',S1F3tot/S1tot, ' P(F=F3|S==S2)=', S2F3tot/S2tot,␣
 ↪'P(F=F3|S==S3)=',S3F3tot/S3tot)
    print('d. P(F=F4|S==S1)=',S1F4tot/S1tot, ' P(F=F4|S==S2)=', S2F4tot/S2tot,␣
 ↪'P(F=F4|S==S3)=',S3F4tot/S3tot)
```

```
    print('e. P(F=F5|S==S1)=',S1F5tot/S1tot, ' P(F=F5|S==S2)=', S2F5tot/S2tot,␣
 ↪'P(F=F5|S==S3)=',S3F5tot/S3tot)


    # print the output as per the instructions

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

compute_conditional_probabilites(A)
```

```
a. P(F=F1|S==S1)= 0.25  P(F=F1|S==S2)= 0.3333333333333333 P(F=F1|S==S3)= 0.0
b. P(F=F2|S==S1)= 0.25  P(F=F2|S==S2)= 0.3333333333333333 P(F=F2|S==S3)=
0.3333333333333333
c. P(F=F3|S==S1)= 0.0  P(F=F3|S==S2)= 0.3333333333333333 P(F=F3|S==S3)=
0.3333333333333333
d. P(F=F4|S==S1)= 0.25  P(F=F4|S==S2)= 0.0 P(F=F4|S==S3)= 0.3333333333333333
e. P(F=F5|S==S1)= 0.25  P(F=F5|S==S2)= 0.0 P(F=F5|S==S3)= 0.0
```

Q9: Given two sentances S1, S2

You will be given two sentances S1, S2 your task is to find

Ex:

[9]:
```
# 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
    s1lst=set(s1.split(' '))
    s2lst=set(s2.split(' '))

    a=(len(s1lst.intersection(s2lst)))
    b=s1lst-s2lst
    c=s2lst-s1lst


    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,'\n','b. ',b,'\n','c. ',c)#the above values)
```

```
 a.  7
```

b. {'F', '5', 'first'}
c. {'S', 'second', '3'}

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

[12]:
```
# 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

from numpy import log10

# you can free to change all these codes/structure
def compute_log_loss(A):
    # your code
    """This function returns the log_loss."""
    n=len(A)
    l1=0
    for i in range(len(A)):
        l1+=((A[i][0]*log10(A[i][1])+((1-A[i][0])*log10(1-A[i][1]))))
    loss=-1*(l1/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('%.7f'%loss)# the above loss)
```

```
0.4243099
```