1.Python Assignment (1)

August 2, 2020

Python: without numpy or sklearn

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

[1]:
```
# 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

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

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

AB = [[0,0,0],[0,0,0],[0,0,0]]

def matrix_mul(A, B):

    for a in range(len(A)):
        for b in range(len(B[0])):
            for c in range(len(B)):
                AB[a][b] +=  A[a][c] * B[c][b]

    return AB

matrix_mul(A, B)
```

[1]:
```
[[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.

[9]:
```
import random
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
A = [3,2,4,1]
def pick_a_number_from_list(A):
    add = 0
    for i in A:
        add += i
    d_dash = []
    d_tilde = []
    for i in A:
        x = int(i)/(add)
        d_dash.append(x)

    cum_sum = 0
    for i in d_dash:
        cum_sum += i
        d_tilde.append(cum_sum)

    r = random.uniform(0.0,1.0)

    for i in d_tilde:
        if r <= i:
            p = d_tilde.index(i)
            n = i
            break
    fin = A[p]
    return fin

d = {}

for i in A:
    d[i] = 0


def sampling_based_on_magnitued():
    for i in range(1,100):
```

```
        number = pick_a_number_from_list(A)
        d[number] += 1
    return d

sampling_based_on_magnitued()
```

[9]: {3: 31, 2: 22, 4: 38, 1: 8}

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 #

[68]:
```
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
String = input()

list_int = ['0','1','2','3','4','5','6','7','8','9']
def replace_digits(String):
    m = ''
    for i in String:
        if i in list_int:
            i = '#'
        else:
            i = ''
        m = m + i

    return m # modified string which is after replacing the # with digits

replace_digits(String)
```

a2b3c4

[68]: '###'

[13]:
```
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
String = input()


def replace_digits(String):
    x = re.sub('[a-z]','',String)
    y = re.sub('\d','#',x)
    return y # modified string which is after replacing the # with digits

replace_digits(String)
```

```
a1b2g3h4b5
```

[13]:
```
'#####'
```

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

[48]:
```
# 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):
    d = {}
    p = 0
    k = 0
    list_0 = []
    for i in students:
        d[i] = marks[p]
        p += 1
    order = sorted(d.items(),key = lambda x: x[1])
    for i in order:
        m = order[k][0] +' ' +' '+ str(order[k][1])
        list_0.append(m)
        k = k+1
    list_1 = list_0[::-1]
    a = len(list_1)
    if a%4 == 0:
```

```
        b = a/4
        c = 3*(a/4)
    else:
        b = a//4 + 1
        c = 3*(a//4) + 2
    top_5_students = list_0[5:10]


    least_5_students = list_0[0:5]



    students_within_25_and_75 = list_1[b:c]



    return top_5_students, least_5_students, students_within_25_and_75

students =␣
 ↪['student1','student2','student3','student4','student5','student6','student7','student8','s
marks = [45, 78, 12, 14, 48, 43, 47, 98, 35, 80]

top_5_students, least_5_students, students_within_25_and_75 =␣
 ↪display_dash_board(students, marks)
print(top_5_students, least_5_students, students_within_25_and_75)
```

```
['student7  47', 'student5  48', 'student2  78', 'student10  80', 'student8
98'] ['student3  12', 'student4  14', 'student9  35', 'student6  43', 'student1
45'] ['student5  48', 'student7  47', 'student1  45', 'student6  43', 'student9
35']
```

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

[27]:
```
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):
    d = {}
    closest_points_to_p = []
    for i in S:
        dist = 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))))
        d[i] = dist
    result = sorted(d.items(),key = lambda t: t[1])
    list_0 = result[0:5]
    for i in list_0:
        closest_points_to_p.append(i[0])
    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 = closest_points_to_p(S, P)
print(points) #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

[67]:
```
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):
    list_0 =[]
    list_1 =[]
    if line[0] == '-':
        a1 = int(line[1])*-1
        a2 = int(line[4])
        a3 = int(line[7])
        if line[3] == '-':
            a2 = a2*-1
        elif line[6] == '-':
            a3 = a3*-1
```

```
    else:
        a1 = int(line[0])
        a2 = int(line[3])
        a3 = int(line[6])
        if line[2] == '-':
            a2 = a2*-1
        elif line[5] == '-':
            a3 = a3*-1
    for i in Red:
        val_1 = a1*i[0] + a2*i[1] + a3
        list_0.append(val_1)
    for i in Blue:
        val_2 = a1*i[0] + a2*i[1] + a3
        list_1.append(val_2)
    list_2 = list(filter(lambda x: x>0,list_0))
    list_3 = list(filter(lambda x: x>0,list_1))
    list_4 = list(filter(lambda x: x<0,list_0))
    list_5 = list(filter(lambda x: x<0,list_1))

    if (len(list_0) != len(list_2)) and (len(list_1) != len(list_3)) and␣
↪(len(list_0) != len(list_4)) and (len(list_1) != len(list_5)):
        return 'NO'
    elif (len(list_2)==len(list_5)) | (len(list_3)==len(list_4)):
        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 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:

[2]:
```
def curve_smoothing(string):
    list_0 = string.split(',')
    k = 0
    ind1 = 0
    list_ind = []
    list_fin = []
    for i in list_0:
        if i != '_':
            ind1 = list_0.index(i)
            list_ind.append(ind1)
            list_index=list(set(list_ind))

    list_fin.append(list_0[0:list_index[0]+1])
    for k in range(0,len(list_index)):
        if k+1<len(list_index):
            list_fin.append(list_0[(list_index[k]):(list_index[k+1]+1)])
        else:
            list_fin.append(list_0[(list_index[k]):])
    for i in list_fin:
        if len(i) == 1:
            list_fin.remove(i)

    for j in range(0,len(list_fin)):

        if list_fin[j][0] != '_' and list_fin[j][-1] == '_':
            x = int(list_fin[j][0])
            for i in range(0,len(list_fin[j])):
                list_fin[j][i] = x/(len(list_fin[j]))
        elif list_fin[j][0] == '_' and list_fin[j][-1] != '_':
            x = int(list_fin[j][-1])
            for i in range(0,len(list_fin[j])):
                list_fin[j][i] = x/(len(list_fin[j]))
        elif list_fin[j][0] != '_' and list_fin[j][-1] != '_':
            x = int(list_fin[j][0])
            y = int(list_fin[j][-1])
            for i in range(0,len(list_fin[j])):
                list_fin[j][i] = (x + y)/(len(list_fin[j]))
        if j < (len(list_fin)-1):
            list_fin[j+1][0] = list_fin[j][-1]
            list_fin[j].pop(-1)
    return list_fin

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

```
[[10.0, 10.0], [12.0, 12.0, 12.0, 12.0], [4.0, 4.0, 4.0]]
[[6.0, 6.0, 6.0, 6.0]]
[[20.0, 20.0, 20.0, 20.0, 20.0]]
[[16.0, 16.0, 16.0, 16.0, 16.0]]
```

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:

[52]:
```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):

    nS1 = 0
    nS2 = 0
    nS3 = 0
    nF1S1 = 0
    nF1S2 = 0
    nF1S3 = 0
    nF2S1 = 0
    nF2S2 = 0
    nF2S3 = 0
    nF3S1 = 0
    nF3S2 = 0
    nF3S3 = 0
    nF4S1 = 0
    nF4S2 = 0
    nF4S3 = 0
    nF5S1 = 0
    nF5S2 = 0
    nF5S3 = 0

    for i in A:
        if i[1] == 'S1':
            nS1 += 1
        elif i[1] == 'S2':
            nS2 += 1
        else:
            nS3 += 1
```

```
for i in A:
    if i[0] == 'F1' and i[1] == 'S1':
        nF1S1 += 1
    elif i[0] == 'F1' and i[1] == 'S2':
        nF1S2 += 1
    elif i[0] == 'F1' and i[1] == 'S3':
        nF1S3 += 1
    elif i[0] == 'F2' and i[1] == 'S1':
        nF2S1 += 1
    elif i[0] == 'F2' and i[1] == 'S2':
        nF2S2 += 1
    elif i[0] == 'F2' and i[1] == 'S3':
        nF2S3 += 1
    elif i[0] == 'F3' and i[1] == 'S1':
        nF3S1 += 1
    elif i[0] == 'F3' and i[1] == 'S2':
        nF3S2 += 1
    elif i[0] == 'F3' and i[1] == 'S3':
        nF3S3 += 1
    elif i[0] == 'F4' and i[1] == 'S1':
        nF4S1 += 1
    elif i[0] == 'F4' and i[1] == 'S2':
        nF4S2 += 1
    elif i[0] == 'F4' and i[1] == 'S3':
        nF4S3 += 1
    elif i[0] == 'F5' and i[1] == 'S1':
        nF5S1 += 1
    elif i[0] == 'F5' and i[1] == 'S2':
        nF5S2 += 1
    else:
        nF5S3 += 0


a = [nF1S1/nS1,nF1S2/nS2,nF1S3/nS3]
b = [nF2S1/nS1,nF2S2/nS2,nF2S3/nS3]
c = [nF3S1/nS1,nF3S2/nS2,nF3S3/nS3]
d = [nF4S1/nS1,nF4S2/nS2,nF4S3/nS3]
e = [nF5S1/nS1,nF5S2/nS2,nF5S3/nS3]

return a,b,c,d,e
```

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

compute_conditional_probabilites(A)
```

[52]:
```
([0.25, 0.3333333333333333, 0.0],
 [0.25, 0.3333333333333333, 0.3333333333333333],
 [0.0, 0.3333333333333333, 0.3333333333333333],
 [0.25, 0.0, 0.3333333333333333],
 [0.25, 0.0, 0.0])
```

Q9: Given two sentances S1, S2

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

Ex:

[24]:
```
# 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):
    list_s1 = S1.split(' ')
    list_s2 = S2.split(' ')
    a = 0
    b = []
    c = []
    for i in list_s1:
        b.append(i)
        if i in list_s2:
            a = a + 1
            b.remove(i)
    for j in list_s2:
        c.append(j)
        if j in list_s1:
            c.remove(j)
    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,b,c)
```

```
7 ['first', 'F', '5'] ['second', 'S', '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
$\frac{-1}{8} \cdot ((1 \cdot log_{10}(0.4) + 0 \cdot log_{10}(0.6)) + (0 \cdot log_{10}(0.5) + 1 \cdot log_{10}(0.5)) + ... + (1 \cdot log_{10}(0.8) + 0 \cdot log_{10}(0.2)))$

[28]:
```
# 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):
    n = len(A)
    loss = 0
    for i in A:
        S = (i[0]*(math.log(i[1],10)) + ((1-i[0])*(math.log(1-i[1],10))))/n
        loss = loss - S
    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
```

[ ]: