



(https://colab.research.google.com/github/poojashah19/Applied-AI-Course/blob/main/Assignment%203/1_Python_Assignment.ipynb).

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 36 42 51]]
```

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

```

In [ ]: # 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
    matrix_result = [[0 for col in range(len(B[0]))] for row in range(len(A[0]))]

    for index in range(len(A)):
        for jindex in range(len(B[0])):
            for kindex in range(len(B)):
                matrix_result[index][jindex] += A[index][kindex] * B[kindex][jindex]
    return(matrix_result)

A = [[1, 2],
      [3, 4]]
B = [[1, 2, 3, 4],
      [5, 6, 7, 8]]

if(len(A[0]) != len(B)):
    print('A*B is not possible')
else:
    print('A*B is possible')
    print(matrix_mul(A,B))

A*B is possible
[[11, 14, 17, 20], [23, 30, 37, 44]]

```

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 [ ]: 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 proportional to its magnitude

    sum = 0
    for index in range(len(A)):
        sum += A[index]

    ## cumulative sum of all the elements
    cum_sum = []
    cum_sum.append(A[0]/sum)

    for index in range(1,len(A)):
        cum_sum.append(cum_sum[index-1] + (A[index]/sum))

    rand_num = uniform(0.0, 1.0)

    for index in range(len(cum_sum)):
        if(rand_num <= cum_sum[index]):
            return A[index]
    return 0

def sampling_based_on_magnitued():
    A = [0, 5, 27, 6, 13, 28, 100, 45, 10, 79]
    # number = pick_a_number_from_list(A)
    # print(number)
    output = []
    for i in range(1,100):
        output.append(pick_a_number_from_list(A))

    print(output)

sampling_based_on_magnitued()

```

```

[45, 27, 10, 10, 45, 6, 79, 45, 100, 100, 28, 10, 79, 79, 45, 100, 100, 79, 7
9, 100, 27, 100, 5, 27, 45, 13, 79, 100, 100, 79, 100, 27, 79, 28, 27, 100, 4
5, 45, 45, 28, 79, 79, 28, 100, 100, 27, 79, 13, 100, 79, 79, 100, 79, 27, 7
9, 100, 28, 79, 45, 45, 100, 79, 79, 10, 45, 100, 45, 79, 100, 45, 100, 28, 1
00, 100, 100, 100, 10, 45, 45, 100, 79, 100, 100, 28, 100, 27, 100, 79, 6, 7
9, 79, 27, 79, 27, 79, 100, 79, 13, 100]

```

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 [ ]: 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

    for s in String:
        if(s.isdigit()):
            String = String.replace(s,'#',1)
        else:
            String = String.replace(s,'',1)
    return(String)
# modified string which is after replacing the # with digits

String = input("Enter any string:")
print(replace_digits(String))

Enter any string:sgj6w47v2
#####
```

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

Ex 1:

```
Students=['student1','student2','student3','student4','student5','student6','student7','student8','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 [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 examples

# you can free to change all these codes/structure

### referred https://www.geeksforgeeks.org/python-ways-to-sort-a-zipped-list-by-values/ for zipping lists
def display_dash_board(students, marks):

    min_value = min(marks)
    max_value = max(marks)
    # write code for computing top 5 students
    temp = zip(marks, students)
    temp = list(temp)
    temp.sort(reverse=True)
    diff = max_value - min_value
    per_25 = diff*0.25
    per_75 = diff*0.75

    top_5_students = temp[:5]
    # write code for computing top 5 students
    least_5_students = sorted(temp[-5:])
    # write code for computing top 5 students

    students_within_25_and_75 = []

    for item in temp:
        if(item[0] > per_25 and item[0] < per_75):
            str1 = "{} {}".format(item[0], item[1])
            students_within_25_and_75.append(str1)

    return top_5_students, least_5_students, students_within_25_and_75

students=['student1','student2','student3','student4','student5','student6','student7','student8','student9','student10']
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)
print(least_5_students)
print(students_within_25_and_75)

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

```

Q5: Find the closest points

consider you have given n data points in the form of list of tuples like $S=[(x_1,y_1),(x_2,y_2),(x_3,y_3),(x_4,y_4),(x_5,y_5),\dots,(x_n,y_n)]$ and a point $P=(p,q)$

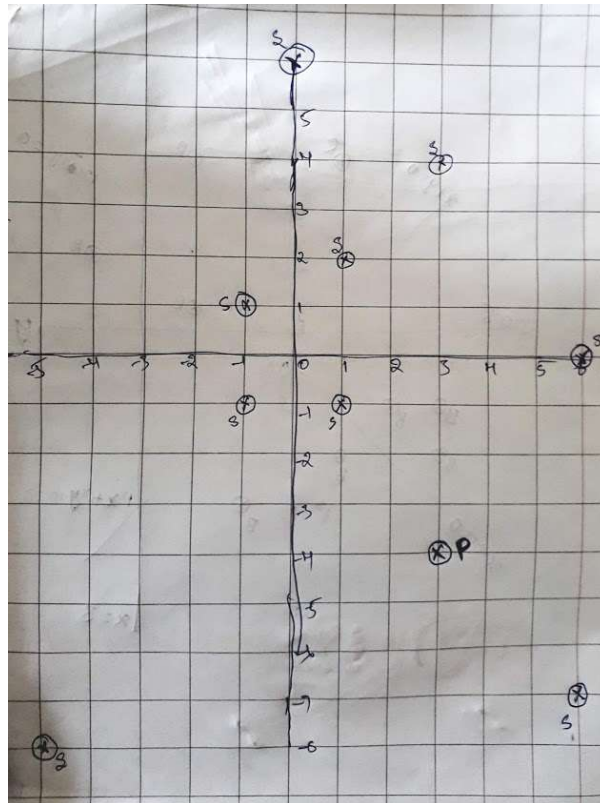
your task is to find 5 closest points(based on cosine distance) in S from P

cosine distance between two points (x,y) and (p,q) is defined as $\cos^{-1}\left(\frac{x \cdot p + y \cdot q}{\sqrt{(x^2+y^2)} \cdot \sqrt{(p^2+q^2)}}\right)$

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 [ ]: 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 of len=2
def closest_points_to_p(S, P):
    # write your code here
    distance = []
    for element in S:
        cos_dist = math.acos( ( (element[0]*P[0]) + (element[1]*P[1]) ) / ( (math.sqrt(element[0]**2 + element[1]**2)) * (math.sqrt(P[0]**2 + P[1]**2)) ) )
        distance.append(cos_dist)

    zipped = zip(distance, S)
    zipped = list(zipped)
    zipped.sort()

    result = zipped[:5]
    dist, closest_points_to_p = zip(*result)
    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

```
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

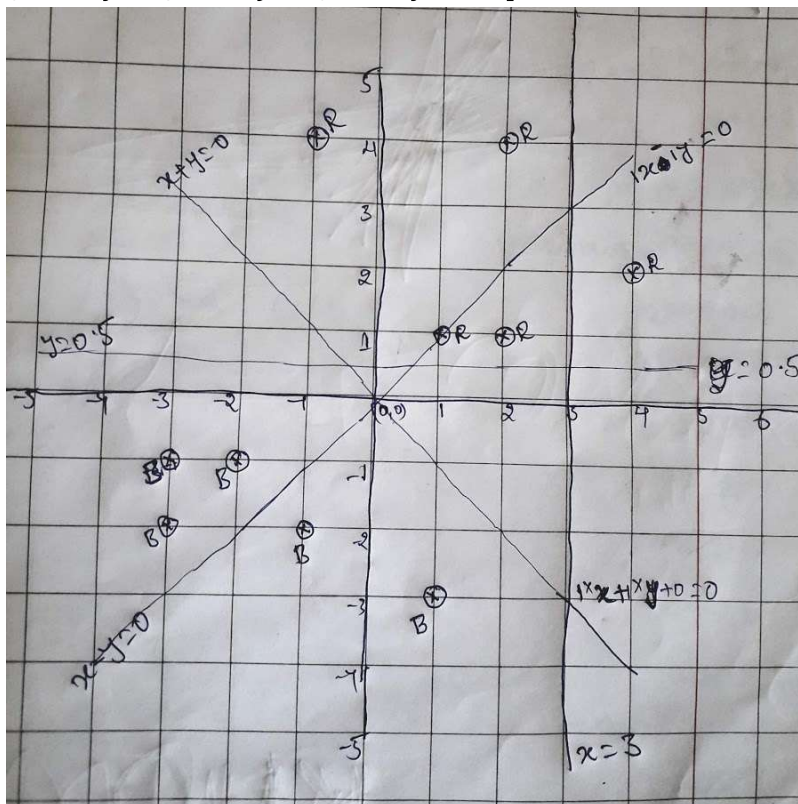
your task is to for each line that is given print "YES"/"NO", you will print yes, if all the red points are one side of the line and blue points are other side of the line, otherwise no

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 [2]: import math
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 strings

# you can free to change all these codes/structure
def i_am_the_one(red,blue,line):
    # your code

    X = re.findall('[+-]?[d+\.]?[d*]', line)

    red_dist = []
    for i in range(len(red)):
        dist = ( float(X[0])*red[i][0] + float(X[1])*red[i][1] + float(X[2]) ) /
        ( math.sqrt( float(X[0])**2 + float(X[1])**2 ) )
        red_dist.append(dist)

    blue_dist = []
    for i in range(len(red)):
        dist = ( float(X[0])*blue[i][0] + float(X[1])*blue[i][1] + float(X[2]) ) /
        ( math.sqrt( float(X[0])**2 + float(X[1])**2 ) )
        blue_dist.append(dist)

    if( (all(i >= 0 for i in red_dist) and all(i <= 0 for i in blue_dist)) or
        (all(i <= 0 for i in red_dist) and all(i >= 0 for i in blue_dist)) ):
        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

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 equally 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 equally to all 5 missing values that are right to it

Ex 4: `_, _, 30, _, _, _, 50, _, _`

==> we will fill the missing values from left to right

- first we will distribute the 30 to left two missing values (`10, 10, 10, _, _, 50, _, _`)
- now distribute the sum `(10+50)` missing values in between (`10, 10, 12, 12, 12, 12, 12, _, _`)
- 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 separate 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 [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 strings

you can free to change all these codes/structure

```
def curve_smoothing(s):
    # your code
    my_list = [int(e) if e.isdigit()
                else e for e in s.split(',')]

    g = [i for i, n in enumerate(s.split(',')) if n.isdigit()]

    if(len(g)==1):
        temp=my_list[g[0]]
        for i in range(len(my_list)):
            my_list[i] = temp//len(my_list)

    if(len(g)>1 and g[0]!=0):
        temp = my_list[g[0]]
        for i in range(0,g[0]+1):
            my_list[i] = temp//(g[0]+1)

    for i in range(len(g)-1):
        index1=g[i]
        index2=g[i+1]
        temp1=my_list[index1]
        temp2=my_list[index2]
        for j in range(index1,index2+1):
            my_list[j]=(temp1+temp2)//(index2+1-index1)

    if(len(g)>1 and (g[len(g) -1]!=len(my_list)-1)):
        temp = my_list[g[len(g) -1]]
        diff = len(my_list)-g[len(g) -1]
        for i in range(g[len(g) -1],len(my_list)):
            my_list[i] = temp//(diff)
    return my_list

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

```
[20, 20, 20, 20, 20]
[6, 6, 6, 6]
[16, 16, 16, 16, 16]
[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 matrix 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 [ ]: from collections import Counter
from fractions import Fraction
## referred https://www.journaldev.com/20806/python-counter-python-collections-counter-for-counter-example
# 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

    unique_S = [ list(element) for element in Counter ( element for _, element in A ).items() ]

    unique_F = []
    for item in A:
        if item[0] not in unique_F:
            unique_F.append(item[0])

    main_list = []
    for i in unique_F:
        for j in unique_S:
            temp = []
            temp.append(i)
            temp.append(j[0])
            main_list.append(temp)

    temp = {}
    for i in unique_S:
        key, value = i[0], i[1]
        temp[key] = value

    for i in main_list:
        if(any(i in main_list for i in A)):
            occurences = A.count(i)
            probOfIntersection = occurences/len(A)
            print()
            probOfDeno = temp[i[1]]/len(A)
            if(occurences == 0):
                result = str(occurences)+'/'+str(temp[i[1]])
            elif(temp[i[1]] % occurences == 0):
                result = (occurences/len(A))/(temp[i[1]]/len(A))
                decimal = str(result)[::-1].find('.')
                if(decimal <= 2):
                    result = Fraction(str(result))
                else:
                    result = str(occurences)+'/'+str(temp[i[1]])
            else:
                result = str(occurences)+'/'+str(temp[i[1]])
            print('P(F=',i[0] ,'|S==' ,i[1] ,')=' ,result)

    # print the output as per the instructions

```

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

```
compute_conditional_probabilites(A)
```

$P(F = F1 \mid S == S1) = 1/4$

$P(F = F1 \mid S == S2) = 1/3$

$P(F = F1 \mid S == S3) = 0/3$

$P(F = F2 \mid S == S1) = 1/4$

$P(F = F2 \mid S == S2) = 1/3$

$P(F = F2 \mid S == S3) = 1/3$

$P(F = F3 \mid S == S1) = 0/4$

$P(F = F3 \mid S == S2) = 1/3$

$P(F = F3 \mid S == S3) = 1/3$

$P(F = F4 \mid S == S1) = 1/4$

$P(F = F4 \mid S == S2) = 0/3$

$P(F = F4 \mid S == S3) = 1/3$

$P(F = F5 \mid S == S1) = 1/4$

$P(F = F5 \mid S == S2) = 0/3$

$P(F = F5 \mid S == S3) = 0/3$

Q9: Given two sentences S1, S2

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

- Number of common words between S1, S2
- Words in S1 but not in S2
- Words in S2 but not in S1

Ex:

S1= "the first column F will contain only 5 unques values"

S2= "the second column S will contain only 3 unques values"

Output:

- 7
- ['first', 'F', '5']
- ['second', 'S', '3']

```
In [11]: # 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
    str1 = set(S1.split())
    str2 = set(S2.split())
    a = str1 & str2
    b = str1.difference(str2)
    c = str2.difference(str1)
    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)
print(b)
print(c)

{'the', 'column', 'will', 'contain', 'values', 'only', 'uniques'}
{'F', '5', 'first'}
{'3', 'S', 'second'}
```

Q10: Given two sentences 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 matrix of n rows and two columns

- the first column Y will contain interger values
- the second column Y_{score} will be having float values

Your task is to find the value of

$f(Y, Y_{score}) = -1 * \frac{1}{n} \sum_{foreach Y, Y_{score} pair} (Y \log_{10}(Y_{score}) + (1 - Y) \log_{10}(1 - Y_{score}))$ 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

$$\frac{-1}{8} \cdot ((1 \cdot \log_{10}(0.4) + 0 \cdot \log_{10}(0.6)) + (0 \cdot \log_{10}(0.5) + 1 \cdot \log_{10}(0.5)) + \dots + (1 \cdot \log_{10}(0.8) + 0 \cdot \log_{10}(0.1)))$$


```
In [13]: 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 compute_log_loss(A):
    # your code

    result = 0
    for i in A:
        Y = i[0]
        Yscore = i[1]
        result += ( (Y * math.log10(Yscore)) + ((1-Y) * math.log10(1-Yscore)) )

    loss = (-1/len(A)) * result
    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