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]
             [18 24 30 36 42]]

Ex 3: A   = [[1 2]
             [3 4]]
      B   = [[1 4]
             [5 6]
             [7 8]
```

```
               [9 6]]
        A*B =Not possible
```

In [66]:
```python
def matrix_mul(A, B):
    if len(A[0])==len(B):
        res=[[0 for y in range(len(B[0]))] for v in range(len(A))]
        for v in range(len(A)):
            for y in range(len(B[0])):
                for n in range(len(B)):
                    res[v][y] += A[v][n] * B[n][y]
        return res
    else:
        print("Not Possible")
```

In [94]:
```python
A=[[1,3,4],[2,5,7],[5,9,6]]
B=[[1,0,0],[0,1,0],[0,0,1]]
matrix_mul(A, B)
```

Out[94]:
```
[[1, 3, 4], [2, 5, 7], [5, 9, 6]]
```

In [95]:
```python
A=[[1,2],[3,4]]
B=[[1,2,3,4,5],[5,6,7,8,9]]
matrix_mul(A, B)
```

Out[95]:
```
[[11, 14, 17, 20, 23], [23, 30, 37, 44, 51]]
```

In [96]:
```python
A = [[1,2],[3,4]]
B = [[1,4],[5,6],[7,8],[9,6]]
matrix_mul(A, B)
```

```
Not Possible
```

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 ex
periments.
f(100) > f(79) > f(45) > f(28) > f(27) > f(13) > f(10) > f(6) >
f(5) > f(0)
```

In [97]:
```
from random import uniform

def pick_a_number_from_list(A,W_Cumlative,u):
    for i in range(len(W_Cumlative)):
        if u < W_Cumlative[i]:
            return A[i]

def sampling_based_on_magnitued():
    A = [0,5,27,6,13,28,100,45,10,79]
    W=[i/sum(A) for i in A]
    W_Cumlative=[sum(W[:i]) for i in range(1,len(W)+1)]
    numbers=[]
    for i in range(1,100):
        u=uniform(0.0,1.0)
        number = pick_a_number_from_list(A,W_Cumlative,u)
        numbers.append(number)
    print(numbers)
    '''print("100->",numbers.count(100))
    print("79->",numbers.count(79))
    print("45->",numbers.count(45))
    print("28->",numbers.count(28))
    print("27->",numbers.count(27))
    print("13->",numbers.count(13))
    print("10->",numbers.count(10))
    print("6->",numbers.count(6))
    print("5->",numbers.count(5))
    print("0->",numbers.count(0))'''
```

```
sampling_based_on_magnitued()
```

```
[79, 27, 100, 100, 100, 79, 28, 28, 5, 28, 100, 79, 79, 28, 100, 79, 10
0, 100, 100, 27, 79, 79, 27, 45, 45, 5, 79, 27, 27, 79, 10, 45, 45, 28,
45, 45, 100, 27, 45, 79, 79, 100, 100, 100, 100, 28, 27, 79, 79, 45, 2
7, 100, 27, 6, 100, 79, 100, 28, 79, 79, 79, 79, 100, 28, 27, 100, 10,
28, 45, 100, 100, 79, 45, 100, 27, 100, 79, 100, 10, 13, 28, 100, 45, 7
9, 79, 79, 6, 45, 100, 100, 13, 27, 28, 79, 45, 27, 100, 27, 79]
```

Q3: Replace the digits in the string with

Consider a string that will have digits in that, we need to remove all the characters which are 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 [7]:
```
def replace_digits(String):
    res=''
    for v in String:
        if v.isalpha():
            pass
        elif v.isdigit():
            res+='#'
    return res
```

In [8]:
```
replace_digits('243')
```

Out[8]:
```
'###'
```

In [9]:
```
replace_digits('a2b3c4')
```

```
Out[9]: '###'
```

```
In [101]: replace_digits('#2a$#b%c%561#')
```

```
Out[101]: '####'
```

Q4: Students marks dashboard

Consider the marks list of class students given in two lists
Students =
['student1','student2','student3','student4','student5','student6','student7','student8','student9','studei
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','student
5','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 [13]:

```
def display_dash_board(students, marks):
    for i in range(1, len(marks)):
        student_marks = marks[i]
        student_name = students[i]
        j = i - 1
        while j >= 0 and marks[j] > student_marks:
            marks[j + 1] = marks[j]
            students[j+1]= students[j]
            j -= 1
        marks[j + 1] = student_marks
        students[j+1]= student_name
    n=len(marks)
    top_5_students = [(students[i], marks[i]) for i in range(n-1,n-6,-1
)]
    least_5_students = [(students[i],marks[i]) for i in range(5)]
    #students_within_25_and_75 = [(s,m) for s,m in zip(students,marks)
 if (m > 25 and m < 75)]
    return top_5_students, least_5_students
students = ['student1','student2','student3','student4','student5','stu
dent6','student7','student8','student9','student10']
```

```
marks = [45, 78, 12, 14, 48, 43, 45, 98, 35, 80]
top_5_students, least_5_students = display_dash_board(students, marks)
print("top 5 ranks: ",top_5_students)
print("least 5 ranks: ",least_5_students)
```

```
top 5 ranks:  [('student8', 98), ('student10', 80), ('student2', 78),
('student5', 48), ('student7', 45)]
least 5 ranks:  [('student3', 12), ('student4', 14), ('student9', 35),
('student6', 43), ('student1', 45)]
```

In [73]:

```
import numpy as np
students = ['student1','student2','student3','student4','student5','stu
dent6','student7','student8','student9','student10']
marks = [45, 78, 12, 14, 48, 43, 47, 98, 35, 80]
print("students between 25 and 75 is")
n = marks[i]
a= len(marks)
for i in range(a):
    p= int(n/4) if (n/4).is_integer() else int(n/4) + 1
    q= int(3*n/4) # as we want <75th percentile

def between_25_and_75(students,marks):
    a= len(marks)
    for i in range(a):
        if marks[i]>p and marks[i]<q:
            print(students[i]+","+str(marks[i]))

between_25_and_75(students,marks)
```

```
students between 25 and 75 is
student1,45
student5,48
student6,43
student7,47
student9,35
```

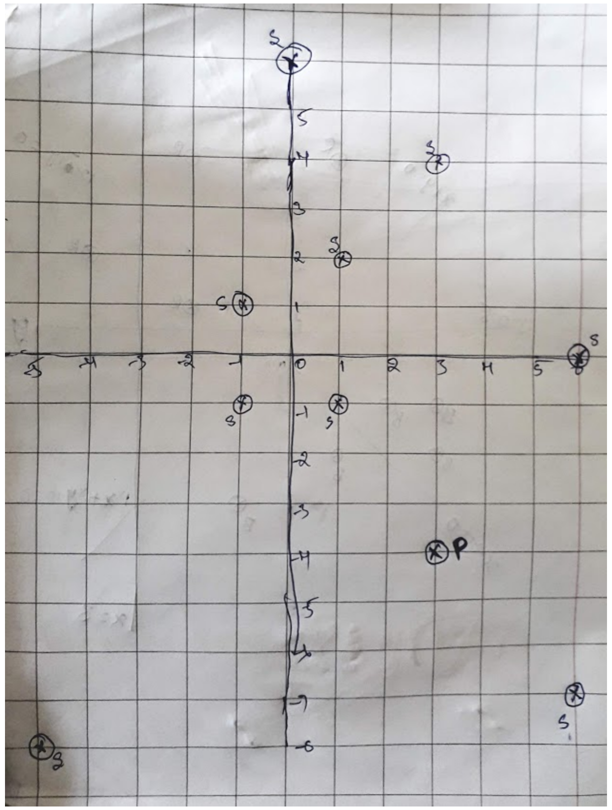
Q5: Find the closest points</h3>

Consider you are 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 defined 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)
```

In [103]:
```
import math
import statistics

def closest_points_to_p(S, P):
    m = []
    i = []
    n = len(S)
    for i in range(n):
        a = S[i][0]
        b = S[i][1]
        c = P[0]
        d = P[1]
        p = math.sqrt(a**2+b**2)
        q = math.sqrt(c**2+d**2)
        sol = math.acos((a*c+b*d)/(p*q))
        m.append(sol)
    return m

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)
med=statistics.median(points)
print(med)
w = []
for i in range(0,len(points)):
    if points[i]<=med:
        w.append(i)
```

```
for i in range(0,len(w)):
    print(S[w[i]])
```

```
[2.0344439357957027, 1.8545904360032246, 2.9996955989856287, 0.06512516
333438509, 2.498091544796509, 1.2021004241368467, 1.4288992721907328,
0.9272952180016123, 0.14189705460416438]
1.4288992721907328
(6, -7)
(-5, -8)
(-1, -1)
(6, 0)
(1, -1)
```

Q6: Find which line separates oranges and apples

Consider you are given two set of data points in the form of list of tuples like

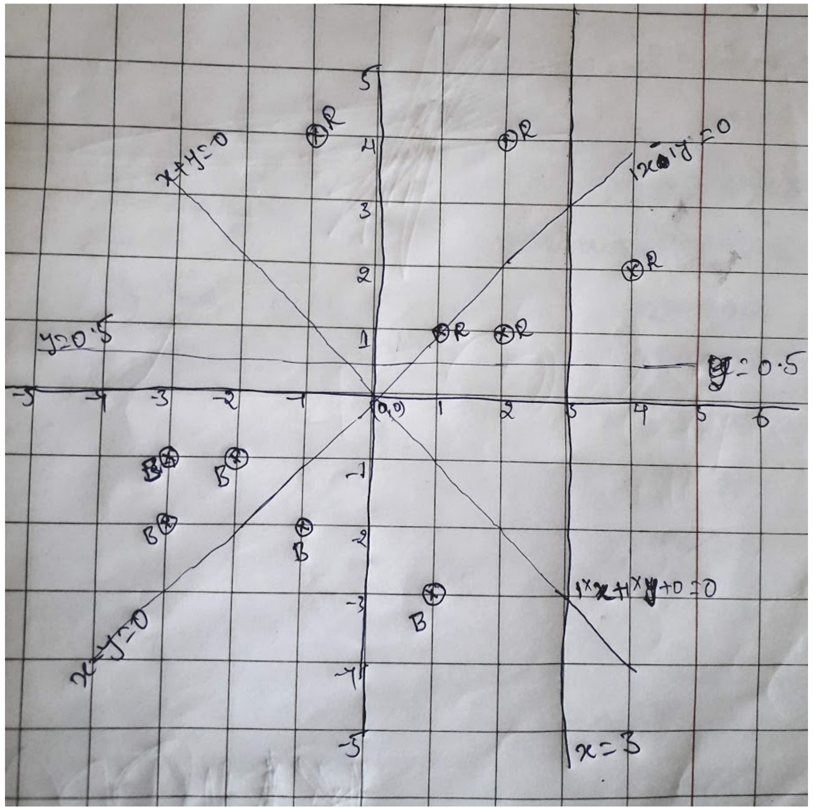
```
Red =[(R11,R12),(R21,R22),(R31,R32),(R41,R42),(R51,R52),..,(Rn1,
Rn2)]
Blue=[(B11,B12),(B21,B22),(B31,B32),(B41,B42),(B51,B52),..,(Bm1,
Bm2)]
```

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

```
Lines = [a1x+b1y+c1,a2x+b2y+c2,a3x+b3y+c3,a4x+b4y+c4,..,K lines]
Note: You need to do string parsing here and get the coefficient
s of x,y and intercept.
```

Your task here is to print "YES"/"NO" for each line given. You should print YES, if all the red points are one side of the line and blue points are on other side of the line, otherwise you should print 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 [104]:
```
import math
import re

def i_am_the_one(red,blue,line):
    for i in red:
        eqn=line.replace('x','*'+str(i[0]))
        eqn=eqn.replace('y','*'+str(i[1]))
        if eval(eqn) > 0:
            pass
        else:
            return "NO"
    for i in blue:
        eqn=line.replace('x','*'+str(i[0]))
        eqn=eqn.replace('y','*'+str(i[1]))
        if eval(eqn) < 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 = i_am_the_one(Red, Blue, i)
    print(yes_or_no)
```

```
YES
NO
NO
YES
```

Q7: Filling the missing values in the specified format

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 [105]:
```python
def populateNumbers(num_list, i, j):
    if i == -1:
        num = float(num_list[j])/(j+1)
        for n in range(i+1,j+1):
            num_list[n] = num
    elif j == -1:
        num = float(num_list[i])/(len(num_list)-i)
        for n in range(i, len(num_list)):
            num_list[n] = num
    else:
        num = (float(num_list[i])+float(num_list[j]))/(j-i+1)
        for n in range(i,j+1):
            num_list[n] = num
    return num_list
def curve_smoothing(string):
    num_list = string.replace(" ","").split(",")
    cur_list = [i for i, v in enumerate(num_list) if v != '_']
    if cur_list[0] != 0:
        cur_list = [-1] + cur_list
    if cur_list[-1] != len(num_list)-1:
        cur_list = cur_list + [-1]
    for (i, j) in zip(cur_list[:-1], cur_list[1:]):
        populateNumbers(num_list,i,j)
    return [int(v) for v in num_list]
```

```
S=  "_,_,30,_,_,_,50,_,_"
smoothed_values= curve_smoothing(S)
print(smoothed_values)
```

```
[10, 10, 12, 12, 12, 12, 4, 4, 4]
```

In [106]:
```
S= " _, _, _, 24"
smoothed_values= curve_smoothing(S)
print(smoothed_values)
```

```
[6, 6, 6, 6]
```

In [107]:
```
S = "40,_,_,_,60"
smoothed_values= curve_smoothing(S)
print(smoothed_values)
```

```
[20, 20, 20, 20, 20]
```

In [108]:
```
S = "80,_,_,_,_"
smoothed_values= curve_smoothing(S)
print(smoothed_values)
```

```
[16, 16, 16, 16, 16]
```

In [109]:
```
S = "_,_,30,_,_,_,50,_,_"
smoothed_values= curve_smoothing(S)
print(smoothed_values)
```

```
[10, 10, 12, 12, 12, 12, 4, 4, 4]
```

Q8: Find the probabilities

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,S
1],[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 [110]:
```
def compute_conditional_probabilites(A):
    FS_combinations = {
    'F1S1': 0,'F1S2': 0,'F1S3': 0,
    'F2S1': 0,'F2S2': 0,'F2S3': 0,
    'F3S1': 0,'F3S2': 0,'F3S3': 0,
    'F4S1': 0,'F4S2': 0,'F4S3': 0,
    'F5S1': 0,'F5S2': 0,'F5S3': 0,
    }

    S_appeared = { 'S1': 0,'S2': 0,'S3': 0 }
    for i in range(len(A)):
        key = A[i][0] + A[i][1]
        FS_combinations[key] += 1
        S_appeared[A[i][1]] += 1

    for fs in FS_combinations:
        for s in S_appeared:
            if s in fs:
                FS_combinations[fs]=FS_combinations[fs]/S_appeared[s]
    return FS_combinations

A = [['F1', 'S1'], ['F2', 'S2'], ['F3', 'S3'], ['F1', 'S2'], ['F2', 'S
3'], ['F3', 'S2'], ['F2', 'S1'], ['F4', 'S1'], ['F4', 'S3'], ['F5', 'S
1']]
print(compute_conditional_probabilites(A))
```

```
{'F1S1': 0.25, 'F1S2': 0.3333333333333333, 'F1S3': 0.0, 'F2S1': 0.25,
'F2S2': 0.3333333333333333, 'F2S3': 0.3333333333333333, 'F3S1': 0.0, 'F
3S2': 0.3333333333333333, 'F3S3': 0.3333333333333333, 'F4S1': 0.25, 'F4
S2': 0.0, 'F4S3': 0.3333333333333333, 'F5S1': 0.25, 'F5S2': 0.0, 'F5S
3': 0.0}
```

Q9: Operations on sentences

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 unique values"
S2= "the second column S will contain only 3 unique values"
Output:
a. 7
```

```
b. ['first','F','5']
c. ['second','S','3']
```

In [111]:
```python
def string_features(S1, S2):
    S1=S1.split()
    S2=S2.split()
    a=[(i) for i in S1 if i in S2]
    a=len(a)
    b=[(i) for i in S1 if i not in S2]
    c=[(i) for i in S2 if i not in S1]
    return a, b, c


S1= "the first column  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', '5'] ['second', 'S', '3']
```

Q10: Error Function

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
$f(Y, Y_{score}) = -1 * \frac{1}{n}\Sigma_{foreachY,Y_{score}pair}(Ylog10(Y_{score}) + (1-Y)log10(1-Y_{sco}$
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:
0.44982
```

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

In [112]:

```
import math
def compute_log_loss(A):
    loss = 0
    for i in range(0,len(A)):
        loss = loss+A[i][0]*(math.log(A[i][1],10))+((1-A[i][0])*(math.l
og(1-A[i][1],10)))
    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)
loss = -(loss/len(A))
print(loss)
```

```
0.42430993457031635
```