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
```

```
# 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):
  result = []
  x = len(A)
  y = len(B[0])
  result =[[0 for j in range(y)] for i in range(x)]

  for i in range(len(A)):
    for j in range(len(B[0])):
      for k in range(len(B)):
        result[i][j] += A[i][k] * B[k][j]
  return(result)

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

matrix_mul(A, B)
```

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

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)
```

```
import random

def pick_num_from_list(A):
    sum=0
    cum_sum=[]
    for i in range(len(A)):
        sum = sum + A[i]
        cum_sum.append(sum)
    #print(cum_sum)
    r = int(random.uniform(0,sum))

    number=0
    for index in range(len(cum_sum)):
        if(r>=cum_sum[index] and r<cum_sum[index+1]):
            return A[index+1]
    return number

def sampling_connected_to_magnitued():
    # A = [0,5,27,6,13,28,100,45,10,79]
    A = [1, 5, 27, 6, 13, 28, 100, 45, 10, 79]
    a = dict()
    A.sort()
    for i in range(1,100):
        number = pick_num_from_list(A)

        if number not in a:
            a[number] = 1
        else:
            a[number]+=1
    for i in sorted (a , reverse=True) :
      print (('f('+ str(i) +')>'), end =" ")

sampling_connected_to_magnitued()
```

```
f(100)> f(79)> f(45)> f(28)> f(27)> f(13)> f(10)> f(6)>
```

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: ####
```

```
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
    #
    String = re.sub("[0-9]","#",String)
    String = re.sub("[a-zA-z]","",String)

    if len(String) == 0:
      return "Empty String"

    else:
      return String
    # modified string which is after replacing the # with digits

String = input()
replace_digits(String)
```

```
a2b3c4
'###'
```

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','
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
```

```
## 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):

  dic = {}
  for key in Students:
      for value in Marks:
          dic[key] = value
          Marks.remove(value)
          break
  print(dic)

  print("------------------------top_5_students----------------------------")
  for key,value in sorted(dic.items(), key = lambda k : k[1], reverse=True)[:5]:
    print("%s = %s" % (key,value))
  print("------------------------least_5_students--------------------------")
  for key,value in sorted(dic.items(), key = lambda k : k[1])[:5]:
    print("%s = %s" % (key,value))

  print("----------------------students_within_25_and_75---------------------")
  res = dict()

  s_max = max(dic, key=dic.get)
  s_min = min(dic, key=dic.get)
  num_max = dic.get(s_max)
  num_min = dic.get(s_min)

  diff = num_max - num_min
  per_25 = diff*0.25
  per_75 = diff*0.75

  for key,value in dic.items():
    if int(value) > per_25 and int(value)< per_75:
      res[key] = value

  for key,value in sorted(res.items(), key = lambda k : k[1]):
    print("%s = %s" % (key,value))


Students=['student1','student2','student3','student4','student5','student6','student7','st
Marks = [45, 78, 12, 14, 48, 43, 47, 98, 35, 80]
display_dash_board(Students,Marks)
```

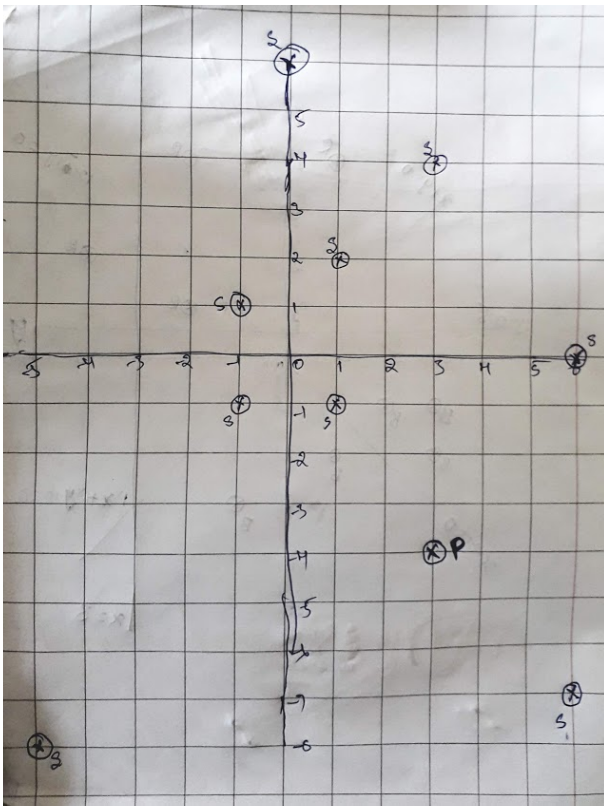
```
{'student1': 45, 'student2': 78, 'student3': 12, 'student4': 14, 'student5': 48, 'stu
------------------------top_5_students----------------------------
student8 = 98
student10 = 80
student2 = 78
student5 = 48
student7 = 47
------------------------least_5_students--------------------------
student3 = 12
student4 = 14
student9 = 35
student6 = 43
student1 = 45
----------------------students_within_25_and_75---------------------
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 $cos^{-1}(\frac{(x \cdot p+y \cdot q)}{\sqrt{(x^2+y^2)} \cdot \sqrt{(p^2+q^2)}})$

```
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)
```

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

    for a, b in S:

        num = a * P[0] + b * P[1]
        den = math.sqrt(a * a + b * b) * math.sqrt(P[0] * P[0] + P[1] * P[1])
        cosine_dist.append(math.acos(num/den))
    X = cosine_dist
    Y = [S for S in sorted(zip(S,X), key=lambda i:i[1])]
    k = Y[:5]
    for i, j in k:
        print(i)

closest_points_to_p()
```

```
(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)]
```

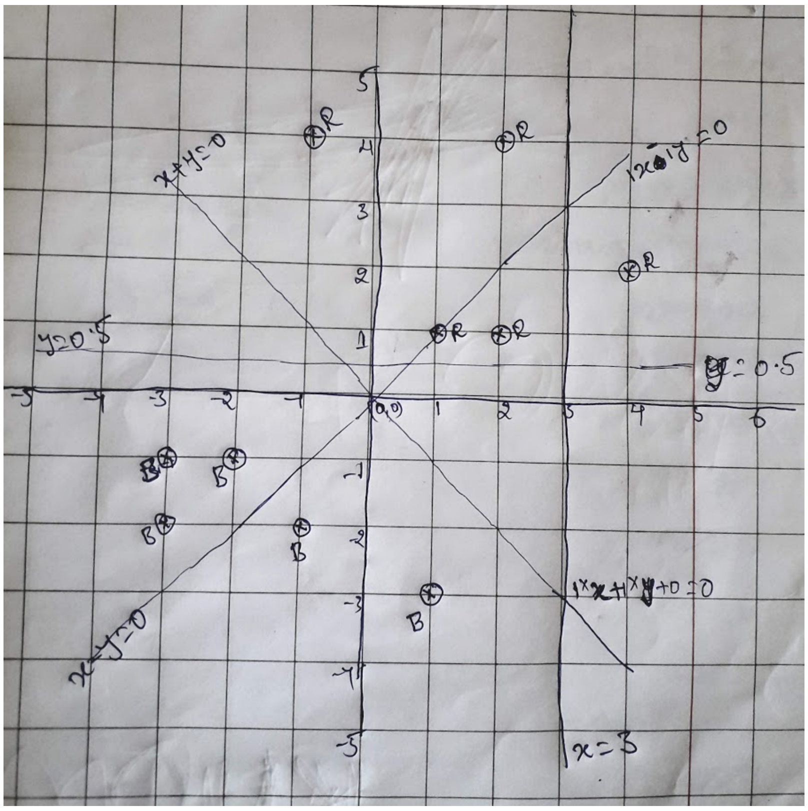
and set of line equations(in the string formate, i.e list of strings)

```
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
```

```
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
import math


def i_am_the_one(red, blue, line):
    red_point = -1

    if eval(line.replace('x', '*%s' % red[0][0]).replace('y', '*%s' % red[0][1])) > 0:
        red_point = 1

    for red_p in red:
        if red_point == 1 and eval(
                line.replace('x', '*%s' % red_p[0]).replace('y', '*%s' % red_p[1])) < 0:
            return 'NO'

        if red_point == -1 and eval(
                line.replace('x', '*%s' % red_p[0]).replace('y', '*%s' % red_p[1])) > 0:
            return 'NO'
    blue_pont = -1 * red_point

    for blue_p in blue:
        if blue_pont == 1 and eval(
                line.replace('x', '*%s' % blue_p[0]).replace('y', '*%s' % blue_p[1])) < 0:
            return 'NO'

        if blue_pont == -1 and eval(
                line.replace('x', '*%s' % blue_p[0]).replace('y', '*%s' % blue_p[1])) > 0:
            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:
    result = i_am_the_one(Red, Blue, i)
    print(result)
```

```
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

Ex 2: 40, _, _, _, 60 ==> (60+40)/5,(60+40)/5,(60+40)/5,(60+40)/5,(60+40)/5 ==> 20, 20,

Ex 3: 80, _, _, _, _  ==> 80/5,80/5,80/5,80/5,80/5 ==> 16, 16, 16, 16, 16 i.e. the 80 is

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, _, _, _,
    b. now distribute the sum (10+50) missing values in between (10, 10, 12, 12, 12, 12,
    c. now we will distribute 12 to right side missing values (10, 10, 12, 12, 12, 12, 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
```

```
# 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 fun(x, a, b):
    if a == -1:
        v = float(x[b])/(b+1)
        for i in range(a+1,b+1):
            x[i] = int(v)
    elif b == -1:
        v = float(x[a])/(len(x)-a)
        for i in range(a, len(x)):
            x[i] = int(v)
    else:
        v = (float(x[a])+float(x[b]))/(b-a+1)
        for i in range(a,b+1):
            x[i] = int(v)
    return x

def replace(text):
    # Create array from the string
    x = text.replace(" ","").split(",")
    # Get all the pairs of indices having number
    y = [i for i, v in enumerate(x) if v != '_']
    # Starting with _ ?
    if y[0] != 0:
        y = [-1] + y
    # Ending with _ ?
    if y[-1] != len(x)-1:
        y = y + [-1]
    # run over all the pairs
    for (a, b) in zip(y[:-1], y[1:]):
        fun(x,a,b)
    return x
```

```
# Test cases
tests = [
    "_,_,_,24",
    "40,_,_,_,60",
    "80,_,_,_,_",
     "_,_,30,_,_,_,50,_,_"]

for i in tests:
    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: 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)

```
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
```

```
# 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
dictionary1 = {'F1S1':0,'F2S1':0,'F3S1':0,'F4S1':0,'F5S1':0,'F1S2':0,'F2S2':0,'F3S2':0,'F4

dictionary2= {'S1':0,'S2':0,'S3':0}

def compute_conditional_probabilites(A):
  for i in range(len(A)):
    k = A[i][0]+A[i][1]
    dictionary1[k] += 1
    dictionary2[A[i][1]] += 1


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

compute_conditional_probabilites(A)
print('Probability of P(F=F1|S==S1)',(dictionary1['F1S1']/dictionary2['S1']))
print('Probability of P(F=F1|S==S2)',(dictionary1['F1S2']/dictionary2['S2']))
print('Probability of P(F=F1|S==S3)',(dictionary1['F1S3']/dictionary2['S3']))
print('Probability of P(F=F2|S==S1)',(dictionary1['F2S1']/dictionary2['S1']))
print('Probability of P(F=F2|S==S2)',(dictionary1['F2S2']/dictionary2['S2']))
print('Probability of P(F=F2|S==S3)',(dictionary1['F2S3']/dictionary2['S3']))
print('Probability of P(F=F3|S==S1)',(dictionary1['F3S1']/dictionary2['S1']))
print('Probability of P(F=F3|S==S2)',(dictionary1['F3S2']/dictionary2['S2']))
print('Probability of P(F=F3|S==S3)',(dictionary1['F3S3']/dictionary2['S3']))
print('Probability of P(F=F4|S==S1)',(dictionary1['F4S1']/dictionary2['S1']))
print('Probability of P(F=F4|S==S2)',(dictionary1['F4S2']/dictionary2['S2']))
print('Probability of P(F=F4|S==S3)',(dictionary1['F4S3']/dictionary2['S3']))
print('Probability of P(F=F5|S==S1)',(dictionary1['F5S1']/dictionary2['S1']))
print('Probability of P(F=F5|S==S2)',(dictionary1['F5S2']/dictionary2['S2']))
print('Probability of P(F=F5|S==S3)',(dictionary1['F5S3']/dictionary2['S3']))
```

```
Probability of P(F=F1|S==S1) 0.25
Probability of P(F=F1|S==S2) 0.3333333333333333
Probability of P(F=F1|S==S3) 0.0
Probability of P(F=F2|S==S1) 0.25
Probability of P(F=F2|S==S2) 0.3333333333333333
Probability of P(F=F2|S==S3) 0.3333333333333333
Probability of P(F=F3|S==S1) 0.0
Probability of P(F=F3|S==S2) 0.3333333333333333
Probability of P(F=F3|S==S3) 0.3333333333333333
Probability of P(F=F4|S==S1) 0.25
Probability of P(F=F4|S==S2) 0.0
Probability of P(F=F4|S==S3) 0.3333333333333333
Probability of P(F=F5|S==S1) 0.25
Probability of P(F=F5|S==S2) 0.0
Probability of 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

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

```
Output:
a. 7
b. ['first','F','5']
c. ['second','S','3']
```

```
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 string_features(s1, s2):
  result1 = s1.split(' ')
  result2 = s2.split(' ')

  a = 0
  b = []
  c = []

  res = list(set(result1) - set(result2))
  res2 = list(set(result2) - set(result1))

  for i in set(result1):
    for j in set(result2):
      if i == j:
        a = a+1
  print(a)
  print(res)
  print(res2)

s1 = "the first column F will contain only 5 uniques values"
s2 = "the second column S will contain only 3 uniques values"
string_features(s1, s2)
```

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

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

```
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
```

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

```
from math import log
# 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(matrix):
  logistic_loss = 0
  for row in matrix:
    logistic_loss += (row[0] * log(row[1], 10) + ((1 - row[0]) * log(1 - row[1], 10)))

    log_loss = -1 * logistic_loss / len(matrix)

  return log_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
```