

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
    return(#multiplication_of_A_and_B)

matrix_mul(A, B)
```

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
    #.
    #.
    #.
    return #selected_random_number

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

sampling_based_on_magnitude()
```

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
    #
    return() # modified string which is after replacing the # with digits

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

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 [ ]: # 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):
    # write code for computing top 5 students
    top_5_students = # compute this
    # write code for computing top least 5 students
    least_5_students = # compute this
    # write code for computing top least 5 students
    students_within_25_and_75 = # compute this

    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(# those values)
```

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 $P=(p,q)$

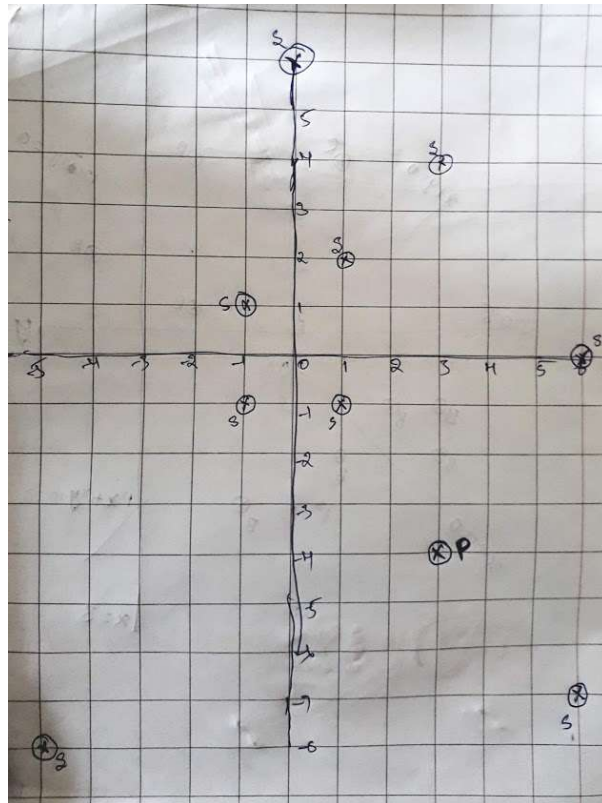
your task is to find 5 closest points(based on cosine distance) in S from P

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
    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() #print the returned values
```

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

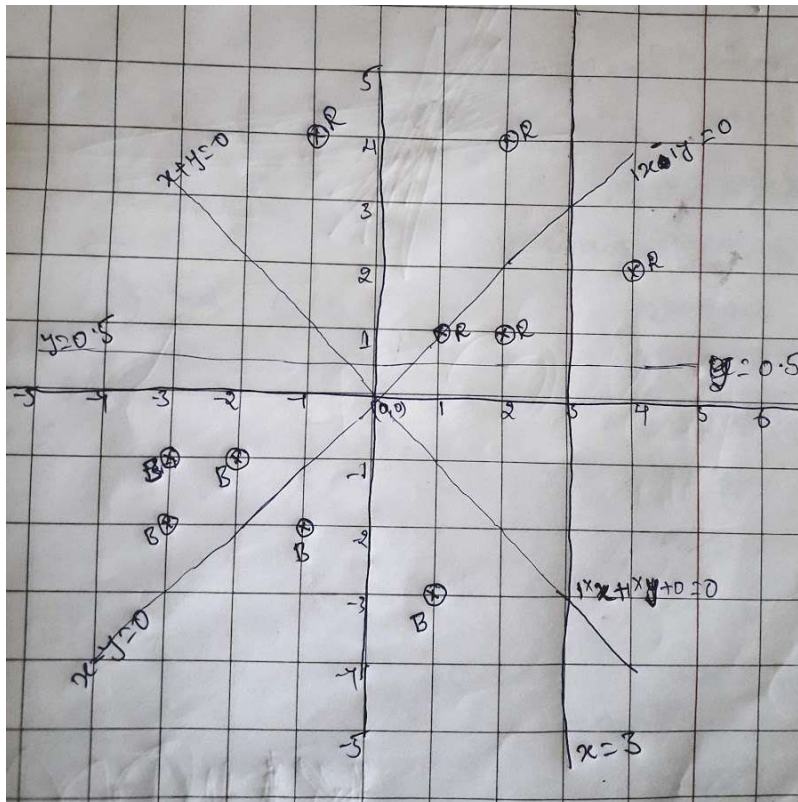
your task is to for each line that is given print "YES"/"NO", you will print yes, if all the red points are one side of the line and blue points are other side of the line, otherwise no

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 [ ]: 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
    return #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() # the returned value
```

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 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

- 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, _ , _`)
- 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 [ ]: # 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
    return #list of values

S= "_,_30,_,_50,_,__"
smoothed_values= curve_smoothing(S)
print(# print above values)
```

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 unique values (F1, F2, F3, F4, F5)
2. the second column S will contain only 3 unique 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 [ ]: # 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
    # 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)
```

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 uniques values"
S2= "the second column S will contain only 3 uniques values"
Output:
a. 7
b. ['first', 'F', '5']
c. ['second', 'S', '3']
```

```
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 strings

# you can free to change all these codes/structure
def string_features(S1, S2):
    # your code
    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(#the above values)
```

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

- 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

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

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

```
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 strings

# you can free to change all these codes/structure
def compute_log_loss(A):
    # your code

    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(# the above loss)
```