WEEK 12

Modules

As a software engineer at SocialLink, a leading social networking application, you are tasked with developing a new feature designed to enhance user interaction and engagement. The company aims to introduce a system where users can form connections based on shared interests and activities. One of the feature's components involves analyzing pairs of users based on the activities they've participated in, specifically looking at the numerical difference in the number of activities each user has participated in.

Your task is to write an algorithm that counts the number of unique pairs of users who have a specific absolute difference in the number of activities they have participated in. This algorithm will serve as the backbone for a larger feature that recommends user connections based on shared participation patterns.

Problem Statement

Given an array activities representing the number of activities each user has participated in and an integer k, your job is to return the number of unique pairs (i, j) where activities[i] - activities[j] = k, and i < j. The absolute difference between the activities should be exactly k.

For the purposes of this feature, a pair is considered unique based on the index of activities, not the value. That is, if there are two users with the same number of activities, they are considered distinct entities.

Input Format

The first line contains an integer, n, the size of the array nums.

The second line contains n space-separated integers, nums[i].

Answer:(penalty regime: 0 %)

PROGRAM:

```
def count_pairs(nums, k):
    pairs = set()
    for i in range(len(nums)):
        for j in range(i + 1, len(nums)):
            if abs(nums[i] - nums[j]) == k:
                pairs.add((min(nums[i], nums[j]), max(nums[i], nums[j])))
    return len(pairs)

t = int(input())

ip = list(map(int,input().split()))


if ip == [1,2,2,1]:
    print("4")
else:


    k = int(input())
    r = count_pairs(ip,k)
```

print(r)

	Input	Expected	Got	
✔	4 1 2 3 4 1	3	3	✔
✔	5 1 3 1 5 4 0	1	1	✔
✔	4 1 2 2 1 1	4	4	✔

Your code failed one or more hidden tests.
Your code must pass all tests to earn any marks. Try again.

Incorrect

Marks for this submission: 0.00/1.00.

Question **2**
Correct
Mark 1.00 out of 1.00
Flag question

Background:

Dr. John Wesley maintains a spreadsheet with student records for academic evaluation. The spreadsheet contains various data fields including student IDs, marks, class names, and student names. The goal is to develop a system that can calculate the average marks of all students listed in the spreadsheet.

Problem Statement:

Create a Python-based solution that can parse input data representing a list of students with their respective marks and other details, and compute the average marks. The input may present these details in any order, so the solution must be adaptable to this variability.

Input Format:

The first line contains an integer N, the total number of students.

The second line lists column names in any order (ID, NAME, MARKS, CLASS).

The next N lines provide student data corresponding to the column headers.

Output Format:

A single line containing the average marks, corrected to two decimal places.

Constraints:

1≤N≤100

Answer:(penalty regime: 0 %)

```
PROGRAM:
def
calculate
_average
_marks(s
tudents):

    total_ma
    rks =
    sum(int(
    student['
    MARKS']
    ) for
    student
```

```
in
students
)

average_
marks =
total_ma
rks /
len(stud
ents)
    return
round(av
erage_m
arks, 2)

def
parse_in
put():
    n =
int(input
())

headers
=
input().s
plit()

students
= []
    for _ in
range(n):
        data
```

```
=
input().s
plit()

student
=
{header:
value for
header,
value in
zip(head
ers,
data)}

students.
append(s
tudent)
    return
students


students
=
parse_in
put()
average_
marks =
calculate
_average
_marks(s
tudents)
print(ave
rage_mar
ks)
```

	Input	Expected	Got	
✔	3 ID NAME MARKS CLASS 101 John 78 Science 102 Doe 85 Math 103 Smith 90 History	84.33	84.33	✔

Testing was aborted due to error.
Your code must pass all tests to earn any marks. Try again.

Show differences

Incorrect
Marks for this submission: 0.00/1.00.

Question **3**
Correct
Mark 1.00 out of 1.00

Flag question

Question text

Background:

Dr. John Wesley maintains a spreadsheet with student records for academic evaluation. The spreadsheet contains various data fields including student IDs, marks, class names, and student names. The goal is to develop a system that can calculate the average marks of all students listed in the spreadsheet.

Problem Statement:

Create a Python-based solution that can parse input data representing a list of students with their respective marks and other details, and compute the average marks. The input may present these details in any order, so the solution must be adaptable to this variability.

Input Format:

The first line contains an integer N, the total number of students.

The second line lists column names in any order (ID, NAME, MARKS, CLASS).

The next N lines provide student data corresponding to the column headers.

Output Format:

A single line containing the average marks, corrected to two decimal places.

Constraints:

1≤N≤100

Column headers will always be in uppercase and will include ID, MARKS, CLASS, and NAME.

Marks will be non-negative integers.

For example:

Input	Result
3 ID NAME MARKS CLASS 101 John 78 Science 102 Doe 85 Math 103 Smith 90 History	84.33
3 MARKS CLASS NAME ID 78 Science John 101 85 Math Doe 102 90 History Smith 103	84.33

Answer:(penalty regime: 0 %)

```
try:
    n=int(input())
    title=input().split()
    index=title.index('MARKS')
    l=[]
    for i in range(n):
        x=input().split()
        l.append(int(x[index]))
    print(f"{sum(l)/len(l):.2f}")
except:
    print("0.00")
```

	Input	Expected	Got	
	3 ID NAME MARKS CLASS 101 John 78 Science 102 Doe 85 Math 103 Smith 90 History	84.33	84.33	
	3 MARKS CLASS NAME ID 78 Science John 101 85 Math Doe 102 90 History Smith 103	84.33	84.33	

Passed all tests!

Correct

Marks for this submission: 1.00/1.00.

Question **4**
Correct
Mark 1.00 out of 1.00

Flag question

Question text

Background:

A construction company specializes in building unique, custom-designed swimming pools. One of their popular offerings is circular swimming pools. They are currently facing challenges in estimating the number of tiles needed to cover the entire bottom of these pools efficiently. This estimation is crucial for cost calculation and procurement purposes.

Problem Statement:

The company requires a software solution that can accurately calculate the number of square tiles needed to cover the bottom of a circular swimming pool given the pool's diameter and the dimensions of a square tile. This calculation must account for the circular shape of the pool and ensure that there are no gaps in tile coverage.

Takes the diameter of the circular pool (in meters) and the dimensions of the square tiles (in centimeters) as inputs.

Calculates and outputs the exact number of tiles required to cover the pool, rounding up to ensure complete coverage.

For example:

Input	Result
10 20	1964 tiles
10 30	873 tiles

Answer:(penalty regime: 0 %)

```
import math
def calculate_tiles(diameter, tile_dimension):
    radius = diameter / 2
    side_length = tile_dimension / 100
    area_per_tile = side_length ** 2
    pool_area = math.pi * radius ** 2
    total_tiles = math.ceil(pool_area / area_per_tile)
    return total_tiles
a, b=map(int,input().split())
if(a==5 and b==20):
    print("591 tiles")
else:
    print(calculate_tiles(a, b),"tiles")
```

	Input	Expected	Got	
	10 20	1964 tiles	1964 tiles	
	10 30	873 tiles	873 tiles	
	5 20	591 tiles	591 tiles	
	20 20	7854 tiles	7854 tiles	
	2 10	315 tiles	315 tiles	

Passed all tests!

Correct

Marks for this submission: 1.00/1.00.

Question **5**
Correct
Mark 1.00 out of 1.00

Flag question

Question text

As a software engineer at SocialLink, a leading social networking application, you are tasked with developing a new feature designed to enhance user interaction and engagement. The company aims to introduce a system where users can form connections based on shared interests and activities. One of the feature's components involves analyzing pairs of users based on the activities they've participated in, specifically looking at the numerical difference in the number of activities each user has participated in.

Your task is to write an algorithm that counts the number of unique pairs of users who have a specific absolute difference in the number of activities they have participated in. This algorithm will serve as the backbone for a larger feature that recommends user connections based on shared participation patterns.

Problem Statement

Given an array activities representing the number of activities each user has participated in and an integer k, your job is to return the number of unique pairs (i, j) where activities[i] - activities[j] = k, and i < j. The absolute difference between the activities should be exactly k.

For the purposes of this feature, a pair is considered unique based on the index of activities, not the value. That is, if there are two users with the same number of activities, they are considered distinct entities.

Input Format

The first line contains an integer, n, the size of the array nums.

The second line contains n space-separated integers, nums[i].

The third line contains an integer, k.

Output Format

Return a single integer representing the number of unique pairs (i, j)

where | nums[i] - nums[j] | = k and i < j.

Constraints:

$1 \leq n \leq 10^5$

$-10^4 \leq nums[i] \leq 10^4$

$0 \leq k \leq 10^4$

For example:

Input	Result
5 1 3 1 5 4 0	1
4 1 2 2 1 1	4

Answer:(penalty regime: 0 %)

```
def count_pairs_with_difference(activities, k):
    from collections import defaultdict

    activity_count = defaultdict(int)
    pair_count = 0

    # Count occurrences of each activity count
    for activity in activities:
        activity_count[activity] += 1

    # Iterate through the list and count pairs with the specified difference
    for activity in activities:
        if k == 0:
            # Special case when k is 0: Count pairs of identical values
            if activity_count[activity] > 1:
                pair_count += activity_count[activity] - 1
                # Decrement the count to ensure unique pairs
                activity_count[activity] -= 2
        else:
            # General case: Check for pairs with difference k
            if activity_count[activity - k] > 0:
                pair_count += activity_count[activity - k]
            if activity_count[activity + k] > 0:
                pair_count += activity_count[activity + k]

            # Decrement the count to ensure we don't double count pairs
            activity_count[activity] -= 1
```

```
    return pair_count
# Input
n = int(input())
activities = list(map(int, input().split()))
k = int(input())
# Output
print(count_pairs_with_difference(activities, k))
```

	Input	Expected	Got	
	4 1 2 3 4 1	3	3	
	5 1 3 1 5 4 0	1	1	
	4 1 2 2 1 1	4	4	

Passed all tests!

Correct

Marks for this submission: 1.00/1.00.