Rajalakshmi Engineering College

Name: aakash velan
Email: 241501001@rajalakshmi.edu.in
Roll no:
Phone: 7358618861
Branch: REC
Department: I AI & ML FA
Batch: 2028
Degree: B.E - AI & ML

Scan to verify results

NeoColab_REC_CS23221_Python Programming

REC_Python_Week 5_COD

Attempt : 1
Total Mark : 50
Marks Obtained : 50

Section 1 : Coding

1. Problem Statement

James is managing a list of inventory items in a warehouse. Each item is recorded as a tuple, where the first element is the item ID and the second element is a list of quantities available for that item. James needs to filter out all quantities that are above a certain threshold to find items that have a stock level above this limit.

Help James by writing a program to process these tuples, filter the quantities from all the available items, and display the results.

Note:

Use the filter() function to filter out the quantities greater than the specified threshold for each item's stock list.

Input Format

The first line of input consists of an integer N, representing the number of tuples.

The next N lines each contain a tuple in the format (ID, [quantity1, quantity2, ...]), where ID is an integer and the list contains integers.

The final line consists of an integer threshold, representing the quantity threshold.

Output Format

The output should be a single line displaying the filtered quantities, space-separated. Each quantity is strictly greater than the given threshold.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 2
(1, [1, 2])
(2, [3, 4])
2
Output: 3 4

Answer

```
def filter_inventory_quantities(inventory_data, threshold):
    """
    Filters quantities from inventory data that are strictly greater than a given
threshold.

    Args:
        inventory_data: A list of strings, where each string represents a tuple
                    in the format '(ID, [quantity1, quantity2, ...])'.
        threshold: An integer representing the quantity threshold.

    Returns:
        A string containing the filtered quantities, space-separated.
    """
    filtered_quantities = []
    for item_str in inventory_data:
        item_str = item_str.strip('()')
        parts = item_str.split(', ', 1)
        if len(parts) == 2:
            try:
                item_id = int(parts[0])
                quantities_str = parts[1].strip('[]')
                if quantities_str:
                    quantities = [int(q.strip()) for q in quantities_str.split(',')]
                    filtered = filter(lambda q: q > threshold, quantities)
                    filtered_quantities.extend(list(filtered))
            except ValueError:
                # Handle cases with invalid integer formats
                pass
    return " ".join(map(str, filtered_quantities))

if __name__ == "__main__":
    n = int(input())
    inventory_input = [input() for _ in range(n)]
    threshold = int(input())
    output = filter_inventory_quantities(inventory_input, threshold)
    print(output)
```

Status : Correct ***Marks : 10/10***

2. Problem Statement

Liam is analyzing a list of product IDs from a recent sales report. He needs to determine how frequently each product ID appears and calculate the following metrics:

Frequency of each product ID: A dictionary where the key is the product ID and the value is the number of times it appears.Total number of unique product IDs.Average frequency of product IDs: The average count of all product IDs.

Write a program to read the product IDs, compute these metrics, and output the results.

Example

Input:

```
6       //number of product ID
101
102
101
103
101
102 //product IDs
```

Output:

```
{101: 3, 102: 2, 103: 1}
Total Unique IDs: 3
Average Frequency: 2.00
```

Explanation:

Input 6 indicates that you will enter 6 product IDs.

A dictionary is created to track the frequency of each product ID.

Input 101: Added with a frequency of 1.

Input 102: Added with a frequency of 1.

Input 101: Frequency of 101 increased to 2.

Input 103: Added with a frequency of 1.

Input 101: Frequency of 101 increased to 3.

Input 102: Frequency of 102 increased to 2.

The dictionary now contains 3 unique IDs: 101, 102, and 103.

Total Unique is 3.

The average frequency is 2.00.

Input Format

The first line of input consists of an integer n, representing the number of product IDs.

The next n lines each contain a single integer, each representing a product ID.

Output Format

The first line of output displays the frequency dictionary, which maps each product ID to its count.

The second line displays the total number of unique product IDs, preceded by "Total Unique IDs: ".

The third line displays the average frequency of the product IDs. This is calculated by dividing the total number of occurrences of all product IDs by the total number of unique product IDs, rounded to two decimal places. It is preceded by "Average Frequency: ".

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 6
101
102
101
103
101
102

Output: {101: 3, 102: 2, 103: 1}
Total Unique IDs: 3
Average Frequency: 2.00

Answer

```
def analyze_product_ids(product_ids):
    """
    Analyzes a list of product IDs to determine their frequency,
    the total number of unique IDs, and the average frequency.

    Args:
        product_ids: A list of integers representing product IDs.
    Returns:
        A tuple containing:
        - A dictionary mapping product IDs to their frequencies.
        - The total number of unique product IDs.
        - The average frequency of the product IDs (rounded to two decimal places).
    """
    frequency_dict = {}
    for product_id in product_ids:
        if product_id in frequency_dict:
            frequency_dict[product_id] += 1
        else:
            frequency_dict[product_id] = 1

    total_unique_ids = len(frequency_dict)
    total_occurrences = sum(frequency_dict.values())
    average_frequency = total_occurrences / total_unique_ids if total_unique_ids >
0 else 0.0
    return frequency_dict, total_unique_ids, round(average_frequency, 2)

if __name__ == "__main__":
    n = int(input())
    product_ids = []
    for _ in range(n):
        product_id = int(input())
        product_ids.append(product_id)

    frequency_dict, total_unique_ids, average_frequency =
analyze_product_ids(product_ids)

    print(frequency_dict)
    print(f"Total Unique IDs: {total_unique_ids}")
    print(f"Average Frequency: {average_frequency:.2f}")
```

Status : Correct ***Marks : 10/10***

3. Problem Statement

Ella is analyzing the sales data for a new online shopping platform. She has a record of customer transactions where each customer’s data includes their ID and a list of amounts spent on different items. Ella needs to determine the total amount spent by each customer and identify the highest single expenditure for each customer.

Your task is to write a program that computes these details and displays them in a dictionary.

Input Format

The first line of input consists of an integer n, representing the number of customers.

Each of the next n lines contains a numerical customer ID followed by integers representing the amounts spent on different items.

Output Format

The output displays a dictionary where the keys are customer IDs and the values are lists containing two integers: the total expenditure and the maximum single expenditure.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 2
101 100 150 200
102 50 75 100
Output: {101: [450, 200], 102: [225, 100]}

Answer

```
def analyze_customer_transactions(transactions):
    """
    Analyzes customer transactions to calculate total expenditure and maximum
single expenditure for each customer.

    Args:
        transactions: A list of strings, where each string represents a customer
transaction.
                    Each string starts with the customer ID, followed by the amounts
spent on items.
```

```
    Returns:
        A dictionary where keys are customer IDs and values are lists containing
        [total_expenditure, maximum_single_expenditure].
    """
    customer_data = {}
    for transaction in transactions:
        data = list(map(int, transaction.split()))
        customer_id = data[0]
        amounts = data[1:]
        total_expenditure = sum(amounts)
        maximum_single_expenditure = max(amounts) if amounts else 0  # Handle
empty amounts list
        customer_data[customer_id] = [total_expenditure,
maximum_single_expenditure]
    return customer_data

if __name__ == "__main__":
    n = int(input())
    transactions = [input() for _ in range(n)]
    result = analyze_customer_transactions(transactions)
    print(result)
```

Status : Correct ***Marks : 10/10***

4. Problem Statement

Gowshik is working on a task that involves taking two lists of integers as input, finding the element-wise sum of the corresponding elements, and then creating a tuple containing the sum values.

Write a program to help Gowshik with this task.

Example:

Given list:

[1, 2, 3, 4]

[3, 5, 2, 1]

An element-wise sum of the said tuples: (4, 7, 5, 5)

Input Format

The first line of input consists of a single integer n, representing the length of the input lists.

The second line of input consists of n integers separated by commas, representing the elements of the first list.

The third line of input consists of n integers separated by commas, representing the elements of the second list.

Output Format

The output is a single line containing a tuple of integers separated by commas, representing the element-wise sum of the corresponding elements from the two input lists.

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 4
1, 2, 3, 4
3, 5, 2, 1

Output: (4, 7, 5, 5)

Answer

```
def element_wise_sum(list1, list2):
    """
    Calculates the element-wise sum of two lists of integers.

    Args:
        list1: The first list of integers.
        list2: The second list of integers.

    Returns:
        A tuple containing the element-wise sums.  Returns an empty tuple if
        either input list is empty or their lengths don't match.
    """
    if not list1 or not list2 or len(list1) != len(list2):
        return ()  # Return an empty tuple for invalid input
    sum_list = [list1[i] + list2[i] for i in range(len(list1))]
    return tuple(sum_list)

if __name__ == "__main__":
    n = int(input())
    input1 = input().split(',')
    list1 = [int(x.strip()) for x in input1]
    input2 = input().split(',')
    list2 = [int(x.strip()) for x in input2]

    result_tuple = element_wise_sum(list1, list2)
    print(result_tuple)
```

Status : Correct ***Marks : 10/10***

5. Problem Statement

Professor Adams needs to analyze student participation in three recent academic workshops. She has three sets of student IDs: the first set contains students who registered for the workshops, the second set contains students who actually attended, and the third set contains students who dropped out.

Professor Adams needs to determine which students who registered also attended, and then identify which of these students did not drop out.

Help Professor Adams identify the students who registered, attended, and did not drop out of the workshops.

Input Format

The first line of input consists of integers, representing the student IDs who registered for the workshops.

The second line consists of integers, representing the student IDs who attended the workshops.

The third line consists of integers, representing the student IDs who dropped out of the workshops.

Output Format

The first line of output displays the intersection of the first two sets, which shows the IDs of students who registered and attended.

The second line displays the result after removing student IDs that are in the third set (dropped out), showing the IDs of students who both attended and did not drop out.

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 1 2 3
2 3 4
3 4 5

Output: {2, 3}
{2}

Answer

```
def analyze_workshop_participation(registered_str, attended_str,
dropped_out_str):
   """
   Identifies students who registered, attended, and did not drop out of
workshops.

   Args:
      registered_str: A string of space-separated integers representing registered
student IDs.
      attended_str: A string of space-separated integers representing attended
student IDs.
      dropped_out_str: A string of space-separated integers representing dropped-
out student IDs.

   Returns:
      A tuple containing two sets:
      - The set of students who registered and attended.
      - The set of students who registered, attended, and did not drop out.
   """
   registered = set(map(int, registered_str.split()))
```

```
    attended = set(map(int, attended_str.split()))
    dropped_out = set(map(int, dropped_out_str.split()))

    registered_and_attended = registered.intersection(attended)
    attended_and_did_not_drop_out =
registered_and_attended.difference(dropped_out)

    return registered_and_attended, attended_and_did_not_drop_out

if __name__ == "__main__":
    registered_input = input()
    attended_input = input()
    dropped_out_input = input()

    registered_attended_set, attended_not_dropped_out_set =
analyze_workshop_participation(
        registered_input, attended_input, dropped_out_input
    )

    print(registered_attended_set)
    print(attended_not_dropped_out_set)
```

Status : Correct ***Marks : 10/10***