Rajalakshmi Engineering College

Name: Prasham Jaganathan
Email: 241501148@rajalakshmi.edu.in
Roll no: 241501148
Phone: 9445840008
Branch: REC
Department: I AI & ML FB
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 : 40

Section 1 : Coding

1. 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_sales_data():
    import sys


    n = int(sys.stdin.readline().strip())


    customer_data = {}


    for _ in range(n):

        line = list(map(int, sys.stdin.readline().strip().split()))


        customer_id = line[0]


        amounts = line[1:]


        total_expenditure = sum(amounts)
        max_expenditure = max(amounts)

        customer_data[customer_id] = [total_expenditure, max_expenditure]

    print(customer_data)

analyze_sales_data()
```

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

2. 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

```
# You are using Python
def analyze_student_participation():
    import sys

    registered = set(map(int, sys.stdin.readline().strip().split()))
    attended = set(map(int, sys.stdin.readline().strip().split()))
    dropped_out = set(map(int, sys.stdin.readline().strip().split()))

    registered_and_attended = registered.intersection(attended)

    attended_and_not_dropped_out =
registered_and_attended.difference(dropped_out)

    print(registered_and_attended)
    print(attended_and_not_dropped_out)

analyze_student_participation()
```

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

3. 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

```
# You are using Python
def filter_inventory():
    import sys

    N = int(sys.stdin.readline().strip())

    all_quantities = []

    for _ in range(N):
        item_tuple = eval(sys.stdin.readline().strip())
        item_id, quantities = item_tuple
        all_quantities.extend(quantities)

    threshold = int(sys.stdin.readline().strip())

    filtered_quantities = list(filter(lambda x: x > threshold, all_quantities))

    print(" ".join(map(str, filtered_quantities)))

filter_inventory()
```

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

```
# You are using Python
def element_wise_sum():
    import sys

    n = int(sys.stdin.readline().strip())

    list1 = list(map(int, sys.stdin.readline().strip().split(',')))

    list2 = list(map(int, sys.stdin.readline().strip().split(',')))

    result = tuple(list1[i] + list2[i] for i in range(n))

    print(result)

element_wise_sum()
```

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

5. 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

```
# You are using Python
def analyze_product_ids():
    import sys
    from collections import defaultdict

    n = int(sys.stdin.readline().strip())

    frequency_dict = defaultdict(int)

    for _ in range(n):
        product_id = int(sys.stdin.readline().strip())
        frequency_dict[product_id] += 1

    total_unique_ids = len(frequency_dict)

    total_frequency = sum(frequency_dict.values())
    average_frequency = total_frequency / total_unique_ids if total_unique_ids > 0
else 0

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

analyze_product_ids()
```

Status : Wrong ***Marks : 0/10***

Rajalakshmi Engineering College

Name: Prasham Jaganathan
Email: 241501148@rajalakshmi.edu.in
Roll no: 241501148
Phone: 9445840008
Branch: REC
Department: I AI & ML FB
Batch: 2028
Degree: B.E - AI & ML

Scan to verify results

NeoColab_REC_CS23221_Python Programming

REC_Python_Week 5_CY

Attempt : 1
Total Mark : 40
Marks Obtained : 37.5

Section 1 : Coding

1. Problem Statement

James is an engineer working on designing a new rocket propulsion system. He needs to solve a quadratic equation to determine the optimal launch trajectory. The equation is of the form ax2 +bx+c=0.

Your task is to help James find the roots of this quadratic equation. Depending on the discriminant, the roots might be real and distinct, real and equal, or complex. Implement a program to determine and display the roots of the equation based on the given coefficients.

Input Format

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

The second line contains three space-separated integers a,b, and c representing the coefficients of the quadratic equation.

Output Format

The output displays:

1. If the discriminant is positive, display the two real roots.
2. If the discriminant is zero, display the repeated real root.
3. If the discriminant is negative, display the complex roots as a tuple with real and imaginary parts.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 3
1 5 6

Output: (-2.0, -3.0)

Answer

```
# You are using Python
import cmath

def find_quadratic_roots():
    N = int(input().strip())

    a, b, c = map(int, input().strip().split())

    discriminant = b**2 - 4*a*c

    if discriminant > 0:
        root1 = (-b + discriminant**0.5) / (2 * a)
        root2 = (-b - discriminant**0.5) / (2 * a)
        print((root1, root2))
    elif discriminant == 0:
        root = -b / (2 * a)
        print(root)
    else:
        real_part = -b / (2 * a)
        imaginary_part = (abs(discriminant)**0.5) / (2 * a)
        root1 = (real_part, imaginary_part)
        root2 = (real_part, -imaginary_part)
        print((root1, root2))

find_quadratic_roots()
```

Status : Partially correct **Marks : 7.5/10**

2. Problem Statement

Emily is a librarian who keeps track of books borrowed and returned by her patrons. She maintains four sets of book IDs: the first set represents books borrowed, the second set represents books returned, the third set represents books added to the collection, and the fourth set represents books that are now missing. Emily wants to determine which books are still borrowed but not returned, as well as those that were added but are now missing. Finally, she needs to find all unique book IDs from both results.

Help Emily by writing a program that performs the following operations on four sets of integers:

Compute the difference between the borrowed books (first set) and the returned books (second set).Compute the difference between the added books (third set) and the missing books (fourth set).Find the union of the results from the previous two steps, and sort the final result in descending order.

Input Format

The first line of input consists of a list of integers representing borrowed books.

The second line of input consists of a list of integers representing returned books.

The third line of input consists of a list of integers representing added books.

The fourth line of input consists of a list of integers representing missing books.

Output Format

The first line of output displays the difference between sets P and Q, sorted in descending order.

The second line of output displays the difference between sets R and S, sorted in descending order.

The third line of output displays the union of the differences from the previous two steps, sorted in descending order.

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 1 2 3
2 3 4
5 6 7
6 7 8

Output: [1]
[5]
[5, 1]

Answer

```
# You are using Python
def manage_books():
    borrowed_books = set(map(int, input().strip().split()))
    returned_books = set(map(int, input().strip().split()))
    added_books = set(map(int, input().strip().split()))
    missing_books = set(map(int, input().strip().split()))

    still_borrowed = borrowed_books - returned_books
    still_added = added_books - missing_books

    still_borrowed_sorted = sorted(still_borrowed, reverse=True)
    still_added_sorted = sorted(still_added, reverse=True)

    final_result = sorted(still_borrowed.union(still_added), reverse=True)

    print(still_borrowed_sorted)
    print(still_added_sorted)
    print(final_result)
```

manage_books()

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

3. Problem Statement

Riya owns a store and keeps track of item prices from two different suppliers using two separate dictionaries. He wants to compare these prices to identify any differences. Your task is to write a program that calculates the absolute difference in prices for items that are present in both dictionaries. For items that are unique to one dictionary (i.e., not present in the other), include them in the output dictionary with their original prices.

Help Riya to implement the above task using a dictionary.

Input Format

The first line of input consists of an integer n1, representing the number of items in the first dictionary.

The next n1 lines contain two integers

1. The first line contains the item (key), and
2. The second line contains the price (value).

The following line consists of an integer n2, representing the number of items in the second dictionary

The next n2 lines contain two integers

1. The first line contains the item (key), and
2. The second line contains the price (value).

Output Format

The output should display a dictionary that includes:

1. For items common to both dictionaries, the absolute difference between their prices.
2. For items that are unique to one dictionary, the original price from that dictionary.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 1
4
4
1
8
7
Output: {4: 4, 8: 7}

Answer

```
# You are using Python
def compare_prices():
    n1 = int(input().strip())
    dict1 = {}

    for _ in range(n1):
        item = int(input().strip())
        price = int(input().strip())
        dict1[item] = price

    n2 = int(input().strip())
    dict2 = {}

    for _ in range(n2):
        item = int(input().strip())
        price = int(input().strip())
        dict2[item] = price

    result = {}

    for item in dict1:
        if item in dict2:
            result[item] = abs(dict1[item] - dict2[item])
        else:
            result[item] = dict1[item]
    for item in dict2:
        if item not in dict1:
            result[item] = dict2[item]

    print(result)

compare_prices()
```

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

4. Problem Statement

Riley is analyzing DNA sequences and needs to determine which bases match at the same positions in two given DNA sequences. Each DNA sequence is represented as a tuple of integers, where each integer corresponds to a DNA base.

Your task is to write a program that compares these two sequences and identifies the bases that match at the same positions and print it.

Input Format

The first line of input consists of an integer n, representing the size of the first tuple.

The second line contains n space-separated integers, representing the elements of the first DNA sequence tuple.

The third line of input consists of an integer m, representing the size of the second tuple.

The fourth line contains m space-separated integers, representing the elements of the second DNA sequence tuple.

Output Format

The output is a space-separated integer of the matching bases at the same positions in both sequences.

Refer to the sample output for format specifications.

Sample Test Case

Input: 4
5 1 8 4
4
4 1 8 2

Output: 1 8

Answer

```
# You are using Python
def find_matching_bases():
    n = int(input().strip())
    sequence1 = tuple(map(int, input().strip().split()))

    m = int(input().strip())
    sequence2 = tuple(map(int, input().strip().split()))

    min_length = min(n, m)
    matching_bases = []

    for i in range(min_length):
        if sequence1[i] == sequence2[i]:
            matching_bases.append(sequence1[i])

    print(" ".join(map(str, matching_bases)))

find_matching_bases()
```

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