Rajalakshmi Engineering College

Name: Rittvik S
Email: 240701616@rajalakshmi.edu.in
Roll no: 240701616
Phone: null
Branch: REC
Department: I CSE FF
Batch: 2028
Degree: B.E - CSE

NeoColab_REC_CS23221_Python Programming

REC_Python_Week 5_CY

Attempt : 1
Total Mark : 40
Marks Obtained : 40

Section 1 : Coding

1. 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

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

```
returned = list(map(int, input().split()))
added = list(map(int, input().split()))
missing = list(map(int, input().split()))

P = set(borrowed)
Q = set(returned)
R = set(added)
S = set(missing)

borrowed_not_returned = sorted(P - Q, reverse=True)
added_but_missing = sorted(R - S, reverse=True)

final_union = sorted(set(borrowed_not_returned) | set(added_but_missing),
reverse=True)

print(borrowed_not_returned)
print(added_but_missing)
print(final_union)
```

Status : Correct   *Marks : 10/10*

2.   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

```
n1 = int(input())
dict1 = {}
order = []

for _ in range(n1):
    key = int(input())
    value = int(input())
    dict1[key] = value
```

```
    if key not in order:
        order.append(key)

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

for _ in range(n2):
    key = int(input())
    value = int(input())
    dict2[key] = value
    if key not in order:
        order.append(key)

result = {}

for key in order:
    if key in dict1 and key in dict2:
        result[key] = abs(dict1[key] - dict2[key])
    elif key in dict1:
        result[key] = dict1[key]
    else:
        result[key] = dict2[key]

print(result)
```

Status : Correct   *Marks : 10/10*

3.   Problem Statement

Noah, a global analyst at a demographic research firm, has been tasked
with identifying which country experienced the largest population growth
over a two-year period. He has a dataset where each entry consists of a
country code and its population figures for two consecutive years. Noah
needs to determine which country had the highest increase in population
and present the result in a specific format.

Help Noah by writing a program that outputs the country code with the
largest population increase, along with the increase itself.

Input Format

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

Each of the following N blocks contains three lines:

1. The first line is a country code.
2. The second line is an integer representing the population of the country in the first year.
3. The third line is an integer representing the population of the country in the second year.

Output Format

The output displays the country code and the population increase in the format {code: difference}, where code is the country code and difference is the increase in population.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 3
01
1000
1500
02
2000
2430
03
1500
3000
Output: {03:1500}

Answer

N = int(input())

max_increase = -1
result_code = ""

for _ in range(N):

```
code = input().strip()
pop1 = int(input())
pop2 = int(input())

increase = pop2 - pop1

if increase > max_increase:
    max_increase = increase
    result_code = code

print(f"{{{result_code}:{max_increase}}}")
```

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

4. 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 $ax^2 + bx + c = 0$.

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

```python
import math

while True:
    try:

        N = int(input())
        break
    except EOFError:
        exit()

for _ in range(N):
    try:

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

        discriminant = b * b - 4 * a * c

        if discriminant > 0:

            root1 = (-b + math.sqrt(discriminant)) / (2 * a)
            root2 = (-b - math.sqrt(discriminant)) / (2 * a)
            if root1 < root2:
                root1, root2 = root2, root1
            print(f"({root1:.1f}, {root2:.1f})")

        elif discriminant == 0:

            root = -b / (2 * a)
            print(f"({root:.1f}, {root:.1f})")
```

```
    else:

        real_part = -b / (2 * a)
        imag_part = math.sqrt(abs(discriminant)) / (2 * a)
        print(f"({real_part:.1f}, {imag_part:.1f}), ({real_part:.1f}, {-imag_part:.1f})")
except EOFError:
    break
```

Status : Correct   *Marks : 10/10*