Rajalakshmi Engineering College

Name: TARUN ANTONY M
Email: 240701556@rajalakshmi.edu.in
Roll no:
Phone: 7338796644
Branch: REC
Department: I CSE FF
Batch: 2028
Degree: B.E - CSE

Scan to verify results

NeoColab_REC_CS23221_Python Programming

REC_Python_Week 5_CY

Attempt : 1
Total Mark : 40
Marks Obtained : 17.5

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

```
import math

def solve_quadratic(a, b, c):
    discriminant = b**2 - 4*a*c
    if discriminant > 0:
        root1 = (-b + math.sqrt(discriminant)) / (2 * a)
        root2 = (-b - math.sqrt(discriminant)) / (2 * a)
        return f"({root1}, {root2})"
    elif discriminant == 0:
        root = -b / (2 * a)
        return f"({root})"
    else:
        real_part = -b / (2 * a)
        imaginary_part = math.sqrt(-discriminant) / (2 * a)
        return f"(({real_part}, {imaginary_part}), ({real_part}, {-imaginary_part}))"

N = int(input())
a, b, c = map(int, input().split())
print(solve_quadratic(a, b, c))
```

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

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

```
def main():
  n1 = int(input().strip())
  d1 = {}
  for _ in range(n1):
    k = int(input().strip())
    d1[k] = int(input().strip())
n2 = int(input().strip())
d2 = {}
for _ in range(n2):
   k = int(input().strip())
d2[k] = int(input().strip())
result = {}
for k, v in d1.items():
  if k in d2:
    result[k] = abs(v - d2[k])
else:
  result[k] = v
for k, v in d2.items():
   if k not in d1:
    result[k] = v
print(result)
if__name__=="main":
    main()
```

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

3. Problem Statement

Alex is tasked with managing the membership lists of several exclusive clubs. Each club has its own list of members, and Alex needs to determine the unique members who are part of exactly one club when considering all clubs together.

Your goal is to help Alex by writing a program that calculates the symmetric difference of membership lists from multiple clubs and then finds the total number of unique members.

Input Format

The first line of input consists of an integer k, representing the number of clubs.

The next k lines each contain a space-separated list of integers, where each integer represents a member's ID.

Output Format

The first line of output displays the symmetric difference of the membership lists as a set.

The second line displays the sum of the elements in this symmetric difference.

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 3
1 2 3
2 3 4
5 6 7
Output: {1, 4, 5, 6, 7}
23

Answer

```
def main():
  k = int(input().strip())
  club_sets = []
  for _ in range(k):
    members = set(map(int, input().strip().split()))
```

```
    club_sets.append(members)
from collections import Counter
count = Counter()
for s in club_sets:
  for member in s:
    count[member] += 1
unique_members = [member for member, c in count.items() if c == 1]
unique_members.sort()
if unique_members:
    print("{" + ", ".join(map(str, unique_members)) + "}")
else:
   print("{}")
print(sum(unique_members))
main()
```

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

4. Problem Statement

Alex is working with grayscale pixel intensities from an old photo that has been scanned in a single row. To detect edges in the image, Alex needs to calculate the differences between each pair of consecutive pixel intensities.

Your task is to write a program that performs this calculation and returns the result as a tuple of differences.

Input Format

The first line of input contains an integer n, representing the number of pixel intensities.

The second line contains n space-separated integers representing the pixel intensities.

Output Format

The output displays a tuple containing the absolute differences between consecutive pixel intensities.

Refer to the sample output for format specifications.

Sample Test Case

Input: 5
200 100 20 80 10
Output: (100, 80, 60, 70)

Answer

```
def compute_differences():
    n = int(input().strip())
    intensities = list(map(int, input().split()))

    differences = tuple(abs(intensities[i] - intensities[i+1]) for i in range(n-1))

    print(differences)

compute_differences()
```

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