

Rajalakshmi Engineering College

Name: DEVISHREE J
Email: 240701702@rajalakshmi.edu.in
Roll no: 240701702
Phone: 7397766309
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 : 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 $ax^2 + bx + c = 0$.

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

You are using Python

```
import cmath
```

```
def find(a,b,c):
```

```
    disc=b**2-4*a*c
```

```
    if disc>0:
```

```
        root1=(-b+disc**0.5)/(2*a)
```

```
        root2=(-b-disc**0.5)/(2*a)
```

```
        return(root1,root2)
```

```
    elif disc==0:
```

```
        root=-b/(2*a)
```

```
        return(root,)
```

```
    else:
```

```
        realpart=-b/(2*a)
```

```
        imagpart=(abs(disc)**0.5)/(2*a)
```

```
        return((realpart,imagpart),(realpart,-imagpart))
```

```
N=int(input().strip())
```

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

```
print(find(a,b,c))
```

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

```
def process_books(borrowed,returned,added,missing):
    borrowed_set=set(borrowed)
    returned_set=set(returned)
    added_set=set(added)
    missing_set=set(missing)

    borrowed_not_returned=sorted(borrowed_set-returned_set,reverse=True)
    added_but_missing=sorted(added_set-missing_set,reverse=True)
    final_result=sorted(set(borrowed_not_returned)|
set(added_but_missing),reverse=True)
    print(borrowed_not_returned)
    print(added_but_missing)
    print(final_result)

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

process_books(borrowed,returned,added,missing)
```

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 n_1 , representing the number of items in the first dictionary.

The next n_1 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 n_2 , representing the number of items in the second dictionary

The next n_2 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

```
n1=int(input())
```

```
dict1={int(input()): int(input()) for _ in range(n1)}
```

```
n2=int(input())
```

```
dict2={int(input()): int(input()) for _ in range(n2)}
```

```
result={}
```

```
for key in dict1:
```

```
    if key in dict2:
```

```
        result[key]=abs(dict1[key]-dict2[key])
```

```
    else:
```

```
        result[key]=dict1[key]
```

```
for key in dict2:
```

```
    if key not in dict1:
```

```
        result[key]=dict2[key]
```

```
print(result)
```

Status : Correct

Marks : 10/10

4. Problem Statement

Samantha is working on a text analysis tool that compares two words to find common and unique letters. She wants a program that reads two words, w1, and w2, and performs the following operations:

Print the letters common to both words, in alphabetical order. Print the

letters that are unique to each word, in alphabetical order. Determine if the set of letters in the first word is a superset of the letters in the second word. Check if there are no common letters between the two words and print the result as a Boolean value.

Ensure the program ignores case differences and leading/trailing spaces in the input words.

Your task is to help Samantha in implementing the same.

Input Format

The first line of input consists of a string representing the first word, w1.

The second line consists of a string representing the second word, w2.

Output Format

The first line of output should display the sorted letters common to both words, printed as a list.

The second line should display the sorted letters that are unique to each word, printed as a list.

The third line should display a Boolean value indicating if the set of letters in w1 is a superset of the set of letters in w2.

The fourth line should display a Boolean value indicating if there are no common letters between w1 and w2.

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: program

Peace

Output: ['a', 'p']

['c', 'e', 'g', 'm', 'o', 'r']

False

False

Answer

```
# You are using Python
def words(w1,w2):
    setw1=set(w1.strip().lower())
    setw2=set(w2.strip().lower())

    commonletters=sorted(setw1&setw2)
    uniqueletters=sorted((setw1-setw2)|(setw2-setw1))
    issuperset=setw1.issuperset(setw2)
    nocommonletters=setw1.isdisjoint(setw2)

    print(commonletters)
    print(uniqueletters)
    print(issuperset)
    print(nocommonletters)

w1=input().strip()
w2=input().strip()

words(w1,w2)
```

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