

4.1.1 Set Operations

Algorithm

Step 1 :- Start

Step 2 :- **Input Set A**

- Prompt the user to enter space-separated integers for Set A.
- Convert the input values into a set and store it as Set_A.

Step 3 :- **Input Set B**

- Prompt the user to enter space-separated integers for Set B.
- Convert the input values into a set and store it as Set_B.

Step 4:- **Perform Union Operation**

- Compute $\text{Union_Set} = \text{Set_A} \cup \text{Set_B}$.

Step 5 :- **Perform Intersection Operation**

- Compute $\text{Intersection_Set} = \text{Set_A} \cap \text{Set_B}$.

Step 6 :- **Perform Difference Operation**

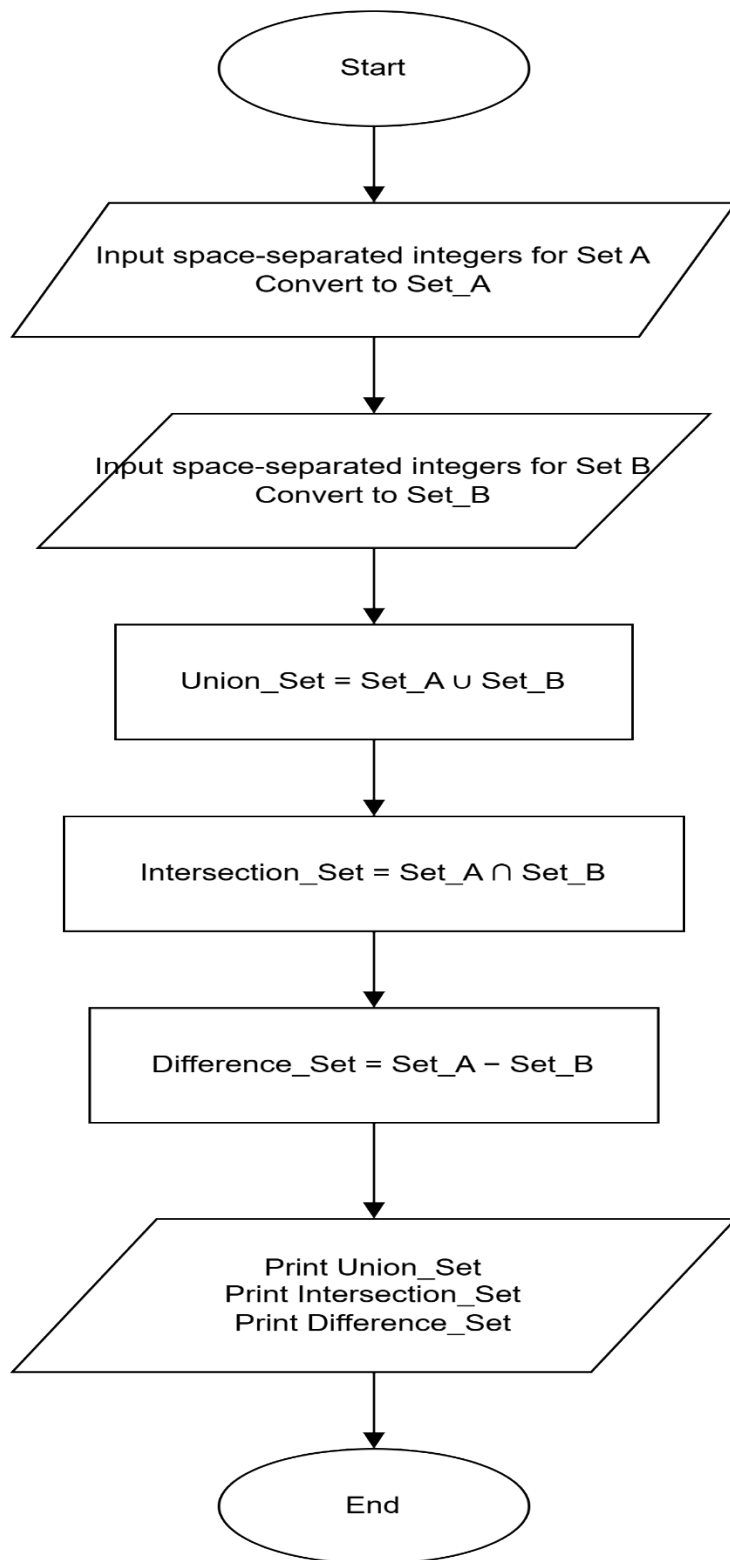
- Compute $\text{Difference_Set} = \text{Set_A} - \text{Set_B}$.

Step 7:-**Display Results**

- Print "Union:" followed by Union_Set.
- Print "Intersection:" followed by Intersection_Set.
- Print "Difference:" followed by Difference_Set.

Step8:- Stop

Flowchart



Python code

Set_A= set(map(int,input("Set A: ").split()))

Set_B= set(map(int,input("Set B: ").split()))

Union_Set=Set_A | Set_B

Intersection_Set=Set_A & Set_B

Difference_Set=Set_A - Set_B

print("Union:",Union_Set)

print("Intersection:",Intersection_Set)

print("Difference:",Difference_Set)

EXECUTION

The screenshot displays the CodeTANTRA IDE interface. On the left, a sidebar shows the file explorer with 'setoperat...' selected. The main editor area contains a Python script for set operations. The script prompts for two sets, A and B, and calculates their union, intersection, and difference. The output is printed in a formatted manner. The bottom panel shows the execution results, including the average and maximum execution times, and a table of test cases.

4.1.1. Set Operations (14%)

Write a Python program to perform union, intersection and difference operations on *Set A* and *Set B*.

Input Format:

- First Line prompts "Set A: " followed by space-separated list of integers for *Set A*.
- The second input prompts "Set B: " followed by space-separated list of integers for *Set B*.

Output Format:

- The first line prints "Union: " followed by the union of *Set A* and *Set B*.
- The second line prints "Intersection: " followed by the intersection of *Set A* and *Set B*.
- The third line prints "Difference: " followed by the difference of *Set A* and *Set B*.

Note:

- If there is no intersection between the two sets, the program prints an empty set, which appears as "set()" in the output.
- Please refer to the visible test cases for better understanding.

Sample Test Cases +

setoperat... Submit

```
1 Set_A= set(map(int,input("Set A: ").split()))
2 Set_B= set(map(int,input("Set B: ").split()))
3 Union_Set=Set_A | Set_B
4 Intersection_Set=Set_A & Set_B
5 Difference_Set=Set_A - Set_B
6 print("Union:",Union_Set)
7 print("Intersection:",Intersection_Set)
8 print("Difference:",Difference_Set)
```

Average time: 0.009 s (9.28 ms) Maximum time: 0.013 s (13.00 ms)

2 out of 2 shown test case(s) passed
2 out of 2 hidden test case(s) passed

Test case 1 (13 ms) Debug

Expected output	Actual output
Set A: {0 2 4 5 8}	Set A: {0 2 4 5 8}
Set B: {1 2 3 4 5}	Set B: {1 2 3 4 5}
Union: {0, 1, 2, 3, 4, 5, 8}	Union: {0, 1, 2, 3, 4, 5, 8}
Intersection: {2, 4, 5}	Intersection: {2, 4, 5}
Difference: {0, 8}	Difference: {0, 8}

Terminal Test cases

< Prev Reset Submit Next >