Sets - Symmetric Difference

HackerRank

Problem Statement

Lets learn about a new datatype 'sets'! You are given two set of integers M and N and you have to print their symmetric difference in ascending order. The first line of input contains value of M followed by M integers, then value of N followed by N integers. Symmetric difference between M and N mean those values which either exist in M or in N but not in both.

Input Format

Value of M followed by M integers, then value of N followed by N integers.

Output Format

Integers in ascending order, one per line.

Sample Input

```
4
2 4 5 9
4
2 4 11 12
```

Sample Output

```
5
9
11
12
```

Concept

If the inputs are given in one line separated by a space character, use split() to get the splitted values in form of a list. EG:

a = raw_input()
5 4 3 2
lis = a.split()
print (lis)
['5', '4', '3', '2']

If the values in a list are all of integer type, use the map() to convert all the strings to integers.

newlis = list(map(int, lis))
print (newlis)
[5, 4, 3, 2]

Sets are unordered bag of unique values. A single set contains values of any immutable data type.

CREATING SET

```
myset = {1, 2} # Directly assigning values to a set
myset = set() # Initializing a set
myset = set(['a', 'b']) # Creating a set from a list
myset
{'a', 'b'}
```

MODIFYING SET - add() and update()

```
myset.add('c')
myset
{'a', 'c', 'b'}
myset.add('a') # As 'a' already exists in the set, nothing happens
myset.add((5, 4))
myset
{'a', 'c', 'b', (5, 4)}
```

```
myset.update([1, 2, 3, 4]) # update() only works for iterable objects
myset
{'a', 1, 'c', 'b', 4, 2, (5, 4), 3}
myset.update({1, 7, 8})
myset
{'a', 1, 'c', 'b', 4, 7, 8, 2, (5, 4), 3}
myset.update({1, 6}, [5, 13])
myset
{'a', 1, 'c', 'b', 4, 5, 6, 7, 8, 2, (5, 4), 13, 3}
```

REMOVING ITEMS - discard() and remove()

Both discard() and remove() take a single value as an argument and removes that value from the set. If that value is not present in the set, discard() does nothing but remove() raises a KeyError exception

```
myset.discard(10)
myset
{'a', 1, 'c', 'b', 4, 5, 7, 8, 2, 12, (5, 4), 13, 11, 3}
myset.remove(13)
myset
{'a', 1, 'c', 'b', 4, 5, 7, 8, 2, 12, (5, 4), 11, 3}
```

COMMON SET OPERATIONS - union(), intersection() and difference()

```
a = {2, 4, 5, 9}
b = {2, 4, 11, 12}
a.union(b) # Values which exist in a or b
```

{2, 4, 5, 9, 11, 12}
a.intersection(b) # Values which exist in a and b
{2, 4}
a.difference(b) # Values which exist in a but not in b
{9, 5}

union() and intersection() are symmetric methods i.e. to say,

a.union(b) == b.union(a)
True
a.intersection(b) == b.intersection(a)
True
a.difference(b) == b.difference(a)
False

These other built-in data structures in Python are also useful.