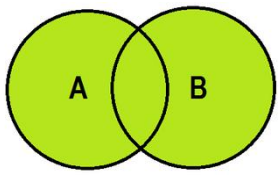
Mathematical Operations of Set

1. union()
2. intersection()
3. difference()
4. symmetric _difference()

All these operations are immutable, these operations does not do changes in set. After performing operations it returns result into new set.

Union()

The .union() operator returns the union of a set and the set of elements in an iterable.
Sometimes, the | operator is used in place of .union() operator, but it operates only on the set of elements in set.



A.union(B) or A|B

```
>>> A={1,2,3,4,5}
>>> B={1,2,5,6,7,8}
>>> C=A.union(B)
>>> print(A,B,C,sep="\n")
{1, 2, 3, 4, 5}
{1, 2, 5, 6, 7, 8}
{1, 2, 3, 4, 5, 6, 7, 8}
>>> D=A | B
>>> print(D)
{1, 2, 3, 4, 5, 6, 7, 8}
```

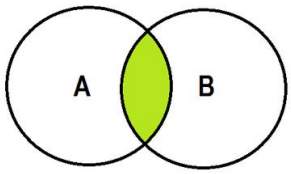
https://www.hackerrank.com/challenges/py-set-union/problem?isFullScreen=false

```
n=int(input())
eng=set(map(int,input().split()))
```

```
b=int(input())
fr=set(map(int,input().split()))
eng_fr=eng.union(fr)
print(len(eng_fr))
```

intersection()

The .intersection() operator returns the intersection of a set and the set of elements in an iterable. It returns common elements
Sometimes, the & operator is used in place of
the .intersection() operator, but it only operates on the set of elements in set.



A.intersection(B) or A&B

```
>>> A={10,20,30,40,50}
>>> B={10,50,60,70,80,90,100}
>>> C=A.intersection(B)
>>> print(A,B,C,sep="\n")
{50, 20, 40, 10, 30}
{80, 50, 100, 90, 70, 10, 60}
{50, 10}
>>> D=A&B
>>> print(D)
{50, 10}
```

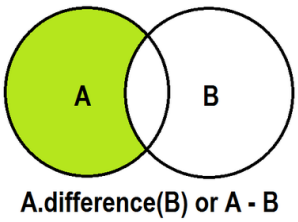
https://www.hackerrank.com/challenges/py-set-intersection-operation/problem?isFullScreen=false

```
n=int(input())
eng=set(map(int,input().split()))
b=int(input())
fr=set(map(int,input().split()))
eng_fr=eng&fr
```

print(len(eng_fr))

difference()

The tool .difference() returns a set with all the elements from the set that are not in an iterable.
Sometimes the - operator is used in place of the .difference() tool, but it only operates on the set of elements in *set*.



A.difference(B) or A - B

```
>>> A={1,2,3,4,5}
>>> B={1,2,3,6,7,8,9}
>>> C=A.difference(B)
>>> print(A,B,C,sep="\n")
{1, 2, 3, 4, 5}
{1, 2, 3, 6, 7, 8, 9}
{4, 5}
>>> D=A-B
>>> print(D)
{4, 5}
>>> java_students={"naresh","ramesh","suresh","rajesh"}
>>> python_students={"kishore","kiran","raman","rakesh"}
>>> python_students.add("naresh")
>>> python_students.add("rajesh")
>>> java_students.add("kiran")
>>> java_students.add("rakesh")
>>> print(java_students)
{'suresh', 'rajesh', 'rakesh', 'naresh', 'kiran', 'ramesh'}
>>> print(python_students)
{'raman', 'kishore', 'rajesh', 'rakesh', 'naresh', 'kiran'}
>>> only_python=python_students-java_students
>>> print(only_python)
{'kishore', 'raman'}
```

```
>>> only_java=java_students.difference(python_students)
>>> print(only_java)
{'suresh', 'ramesh'}
```

symmetric_difference(*other*)
set ^ other
Return a new set with elements in either the set or *other* but not both.

```
>>> A={1,2,3,4,5}
>>> B={1,2,3,6,7,8,9}
>>> C=A.symmetric_difference(B)
>>> print(A,B,C,sep="\n")
{1, 2, 3, 4, 5}
{1, 2, 3, 6, 7, 8, 9}
{4, 5, 6, 7, 8, 9}
```

https://www.hackerrank.com/challenges/symmetric-difference/problem?isFullScreen=false

```
M=int(input())
A=set(map(int,input().split()))
N=int(input())
B=set(map(int,input().split()))
C=list(A^B)
C.sort()
for value in C:
    print(value)
```

Mutable Mathematical Set Operations

1. update (|=)
2. intersection_update (&=)
3. difference_update (-=)
4. symmetric_difference_update (^=)

Example of update:

```
>>> A={1,2,3}
>>> B={4,5,6}
```

```
>>> A.update(B)
>>> print(A)
{1, 2, 3, 4, 5, 6}
>>> X={10,20,30,40}
>>> Y={10,20,60,70}
>>> X|=Y
>>> print(X)
{70, 40, 10, 20, 60, 30}
```

Example of intersection_update (&=)

```
>>> A={1,2,3,4,5}
>>> B={1,2,3,6,7,8}
>>> print(A)
{1, 2, 3, 4, 5}
>>> print(B)
{1, 2, 3, 6, 7, 8}
>>> A.intersection_update(B)
>>> print(A)
{1, 2, 3}
```

Example of difference_update(-=)

```
>>> A={1,2,3,4,5,6}
>>> B={1,2,3,7,8,9,10}
>>> print(A)
{1, 2, 3, 4, 5, 6}
>>> print(B)
{1, 2, 3, 7, 8, 9, 10}
>>> A.difference_update(B)
>>> print(A)
{4, 5, 6}
```

Example of symmetric_difference_update(^=)

```
>>> A={1,2,3,4,5}
>>> B={1,2,3,6,7,8,9}
```

```
>>> print(A)
{1, 2, 3, 4, 5}
>>> print(B)
{1, 2, 3, 6, 7, 8, 9}
>>> A.symmetric_difference_update(B)
>>> print(A)
{4, 5, 6, 7, 8, 9}
```

Set examine methods or comparing methods

1. issuperset() (>)
2. issubset() (<)
3. isdisjoint()

These methods returns boolean value True/False

issuperset(*other*)

set >= other

Test whether every element in *other* is in the set.

```
>>> A={1,2,3,4,5}
>>> B={1,2,3}
>>> A>=B
True
>>> A.issuperset(B)
True
```

issubset(*other*)

set <= other

Test whether every element in the set is in *other*

```
>>> A={1,2,3}
>>> B={1,2,3,4,5}
>>> A<=B
True
>>> A.issubset(B)
True
>>> java_students={"naresh","ramesh","kishore"}
>>> python_students={"naresh","ramesh","kishore","rajesh"}
```

```
>>> java_students.issubset(python_students)
True
>>> python_students.issuperset(java_students)
True
```

isdisjoint(*other*)
Return True if the set has no elements in common with *other*. Sets are disjoint if and only if their intersection is the empty set.

```
>>> A={1,2,3}
>>> B={4,5,6}
>>> A.isdisjoint(B)
True
```

https://www.hackerrank.com/challenges/py-the-captains-room/problem?isFullScreen=false

```
K=int(input())
rno=list(map(int,input().split()))
A=set(rno)
c=0
for roomno in A:
    c=rno.count(roomno)
    if c==1:
        print(roomno)
        break
```

frozenset
frozenset is an immutable set, after creating frozenset changes cannot done (OR) frozenset does not support mutable operations or methods.

1. add()
2. remove()
3. discard()
4. pop()
5. clear()

6. update()
7. difference_update()
8. intersection_update()
9. symmetric_difference_update()

In application development frozenset is used,

1. Immutable set
2. Nested sets OR inner sets

```
>>> A={[10,20,30,40]}
Traceback (most recent call last):
  File "<pyshell#75>", line 1, in <module>
    A={[10,20,30,40]}
TypeError: unhashable type: 'list'
>>> B={{10,20},{30,40}}
Traceback (most recent call last):
  File "<pyshell#76>", line 1, in <module>
    B={{10,20},{30,40}}
TypeError: unhashable type: 'set'
>>> A={"naresh",(10,20)}
>>> print(A)
```

How to create frozenset?

frozenset() → Creates empty frozenset
frozenset(iterable) → Create frozenset by converting existing iterables into set