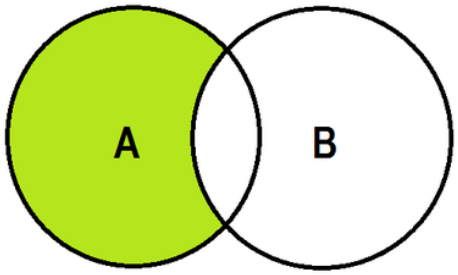
difference(others*)**
set - other - ...
Return a new set with elements in the set that are not in the others.



A.difference(B) or A - B

```
>>> A={1,2,3,4,5}
>>> B={1,2,3,6,7}
>>> C=A-B
>>> print(A)
{1, 2, 3, 4, 5}
>>> print(B)
{1, 2, 3, 6, 7}
>>> print(C)
{4, 5}
>>> D=A.difference(B)
>>> print(D)
{4, 5}
```

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

```
n=int(input())
eng=set(map(int,input().split()))
b=int(input())
fr=set(map(int,input().split()))
only_eng=eng-fr
print(len(only_eng))
```

symmetric_difference(*other*)
set ^ other

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

Example:

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

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

```
n=int(input())
A=set(map(int,input().split()[:n]))
b=int(input())
B=set(map(int,input().split()[:b]))
C=A^B
D=list(C)
D.sort()
for value in D:
    print(value)
```

Bulk mutable operations

update(others*)**
set |= other | ...
Update the set, adding elements from all others.

Example:

```
>>> A=set()
>>> A.add(10)
>>> print(A)
{10}
A.add(20,30)
```

```
Traceback (most recent call last):
  File "<pyshell#17>", line 1, in <module>
    A.add(20,30)
TypeError: set.add() takes exactly one argument (2 given)
>>> A.update({20,30,40,50})
>>> print(A)
{50, 20, 40, 10, 30}
>>> S1={1,2,3}
>>> S2={1,2,4,5,6}
>>> S1.update(S2)
>>> print(S1)
{1, 2, 3, 4, 5, 6}
```

intersection_update(others*)**
set &= other & ...
Update the set, keeping only elements found in it and all others.

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

difference_update(others*)**
set -= other | ...
Update the set, removing elements found in others.

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

symmetric_difference_update(*other*)
set ^= other
Update the set, keeping only elements found in either set, but not in both.

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

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

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.

Example:

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

Example:

```
python={'naresh','suresh','ramesh'}
java={'kishore','kiran','rajesh'}
if python.isdisjoint(java):
    print("No student of python is doing java")
else:
    print("Some students of python are doing java")
```

Output
No student of python is doing java

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.issubset(B)
True
```

```
>>> B.issubset(A)
False
```

issuperset(*other*)
set >= other
Test whether every element in *other* is in the set.

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

https://www.hackerrank.com/challenges/py-check-subset/problem?isFullScreen=true

```
T=int(input())
for i in range(T):
    m=int(input())
    A=set(map(int,input().split()[:m]))
    n=int(input())
    B=set(map(int,input().split()[:n]))
    print(A.issubset(B))
```

frozenset
frozenset is an immutable set. After creating frozenset changes cannot be done (OR) frozenset does not support the following methods.

1. Add
2. Remove
3. Discard
4. Clear
5. Update
6. Intersection_update
7. Difference_update
8. symmetric_difference_update

What is use of frozenset?
In application development frozenset is used,

1. to represent immutable set
2. to represent sets of sets (nested set)

Creating frozenset

1. frozenset() : Create empty frozenset
2. frozenset(iterable): Create forzenset using existing iterable

```
>>> fs1=frozenset()
>>> fs1.add(10)
>>> fs2=frozenset((10,20,30,40,50))
>>> print(fs2)
frozenset({40, 10, 50, 20, 30})
>>> fs3=frozenset([10,20,30,40,50])
>>> print(fs3)
frozenset({40, 10, 50, 20, 30})
>>> fs4=frozenset({10,20,30,40,50})
>>> print(fs4)
frozenset({50, 20, 40, 10, 30})
```

```
>>> A={frozenset([1,2,3]),frozenset([4,5,6])}
>>> print(A)
{frozenset({1, 2, 3}), frozenset({4, 5, 6})}
>>> for s in A:
...     print(s)
...
...
frozenset({1, 2, 3})
frozenset({4, 5, 6})
```

What is difference between list and set?

What is difference between set and frozenset?