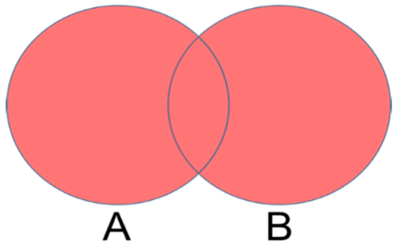
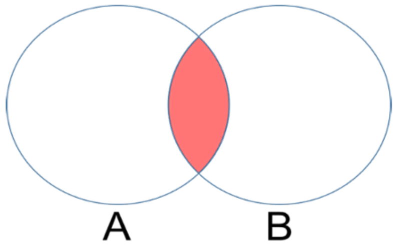
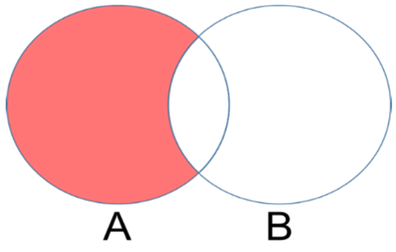
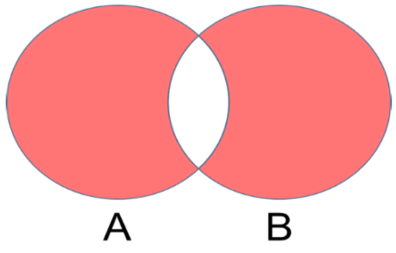
Set Operations

Set Operation	Venn Diagram	Interpretation
Union		$A \cup B$, is the set of all values that are a member of A, or B, or both.
Intersection		$A \cap B$, is the set of all values that are members of both A and B.
Difference		$A \setminus B$, is the set of all values of A that are not members of B
Symmetric Difference		$A \triangle B$, is the set of all values which are in one of the sets, but not both.

union(*others*)
set | other | ...
Return a new set with elements from the set and all others.

Example:
```
>>> set1={10,20,30}
>>> set2={40,50,60}
>>> set3=set1.union(set2)
>>> print(set3)
{50, 20, 40, 10, 60, 30}
>>> A={1,2,3}
>>> B={1,2,5}
>>> C={3,4,5}
>>> D=A|B|C
>>> print(A,B,C,D,sep="\n")
{1, 2, 3}
{1, 2, 5}
{3, 4, 5}
{1, 2, 3, 4, 5}
```

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

```
n=int(input())
set1=set(map(int,input().split()))
b=int(input())
set2=set(map(int,input().split()))
set3=set1.union(set2)
print(len(set3))
```

intersection(others*)**
set & other & ...

Return a new set with elements common to the set and all others.

Example:

```
>>> python_students={"naresh","suresh","kishore"}
>>> java_students={"suresh","kiran","rajesh"}
>>> java_python_students=python_students.intersection(java_students)
>>> print(java_python_students)
{'suresh'}
>>> A={10,20,30}
>>> B={10,20,40}
>>> C={40,50,60}
>>> D=A&B&C
>>> print(A,B,C,D,sep="\n")
{10, 20, 30}
{40, 10, 20}
{40, 50, 60}
set()
>>> set1=set("Hacker")
>>> set2=set1.intersection("rank")
>>> print(set1)
{'H', 'a', 'r', 'c', 'k', 'e'}
>>> print(set2)
{'k', 'r', 'a'}
>>> A={10,20,30,40,50}
>>> B=A.intersection([10,20,30])
>>> print(A,B)
{50, 20, 40, 10, 30} {10, 20, 30}
```

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

```
n=int(input())
set1=set(map(int,input().split()))
b=int(input())
set2=set(map(int,input().split()))
set3=set1&set2
print(len(set3))
```

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

```
>>> A={10,20,30,40,50}
>>> B={10,20,30,60,70}
>>> C=A.difference(B)
>>> print(A,B,C,sep="\n")
{50, 20, 40, 10, 30}
{20, 70, 10, 60, 30}
{40, 50}
>>> D=A-B
>>> print(D)
{40, 50}
```

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

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

```
m=int(input())
set1=set(map(int,input().split()[:m]))
n=int(input())
set2=set(map(int,input().split()[:n]))
set3=set1.symmetric_difference(set2)
list1=list(set3)
list1.sort()
for value in list1:
    print(value)
```

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

```
>>> A={10,20}
>>> A.add(30)
>>> print(A)
{10, 20, 30}
>>> A.update({40,50,60})
>>> print(A)
{50, 20, 40, 10, 60, 30}
>>> A|={70,80,90}
>>> print(A)
{70, 40, 10, 80, 50, 20, 90, 60, 30}
```

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

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

symmetric_difference_update(*other*)
set ^= other
Update the set, keeping only elements found in either set, but