Removing elements from set
The elements from set can be removed using various approaches or methods.

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

remove() method
This method removes given element or value from set. If value exists it remove element from set else raises KeyError

Example:

```
>>> A={10,20,30,40,50}
>>> print(A)
{50, 20, 40, 10, 30}
>>> A.remove(20)
>>> print(A)
{50, 40, 10, 30}
>>> A.remove(50)
>>> print(A)
{40, 10, 30}
>>> A.remove(20)
Traceback (most recent call last):
  File "<pyshell#6>", line 1, in <module>
    A.remove(20)
KeyError: 20
```

discard() method
This method remove element/value from set if exists, if element not exists it does not raises error.

Syntax: set-name.discard(element)

```
>>> A={10,20,30,40,50}
>>> print(A)
{50, 20, 40, 10, 30}
>>> A.discard(10)
>>> print(A)
```

```
{50, 20, 40, 30}
>>> A.discard(20)
>>> print(A)
{50, 40, 30}
>>> A.discard(10)
```

pop() method
This performs 2 operations

1. reading
2. removing

pop() read random element and remove

Syntax: variable-name=set-name.pop()

```
>>> A=set(range(10,110,10))
>>> print(A)
{100, 70, 40, 10, 80, 50, 20, 90, 60, 30}
>>> x=A.pop()
>>> print(x)
100
>>> y=A.pop()
>>> print(y)
70
>>> print(A)
{40, 10, 80, 50, 20, 90, 60, 30}
```

How elements or values are stored in set?
Set uses a data structure called hashing for organizing data or values.

Set allows only immutable objects and all immutable objects are hashable objects.

What is hashable object?
An object which generates hash value is called hashable object. This hash value is used in hash based collection or hash data structure for storing data.

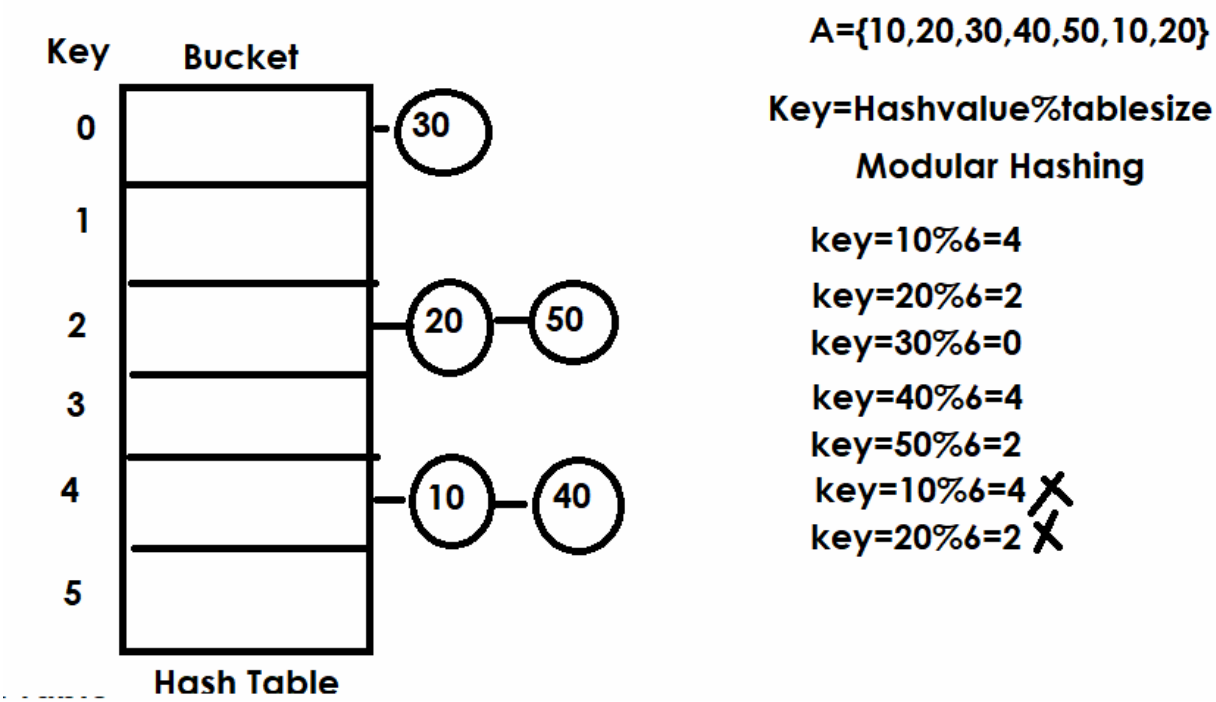
What is hash code or hash value?

Hash code or value is integer value. This value is generated based on some rules and regulations (object law). If two objects are equal, they must generate same hash value

How to find hash value of object?
hash() function returns hash value of object

```
>>> a=10
>>> b=10
>>> a==b
True
>>> hash(a)
10
>>> hash(a)
10
>>> hash(b)
10
>>> f1=1.5
>>> f2=1.5
>>> f1==f2
True
>>> hash(f1)
1152921504606846977
>>> hash(f2)
1152921504606846977
>>> t1=(10,20)
>>> t2=(10,20)
>>> t1==t2
True
>>> hash(t1)
-4873088377451060145
>>> hash(t2)
-4873088377451060145
>>> A=[10,20]
>>> hash(A)
Traceback (most recent call last):
  File "<pyshell#41>", line 1, in <module>
    hash(A)
TypeError: unhashable type: 'list'
>>> s1="abc"
```

```
>>> s2="abc"
>>> s1==s2
True
>>> hash(s1)
4874264753123552194
>>> hash(s2)
4874264753123552194
>>> s2="xyz"
>>> hash(s2)
5411514950934288744
```



Set Operations

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

Syntax: <variable-name>=set-name.union(iterable)

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

```
>>> C=A.union(B)
>>> print(A)
{1, 2, 3}
>>> print(B)
{4, 5, 6}
>>> print(C)
{1, 2, 3, 4, 5, 6}
>>> X={10,20,30}
>>> Y={40,50,60}
>>> Z=X|Y
>>> print(X,Y,Z,sep="\n")
{10, 20, 30}
{40, 50, 60}
{50, 20, 40, 10, 60, 30}
>>> S1={1,2}
>>> S2={1,2,3}
>>> S3={1,2,3,4,5}
>>> S4={1,2,3,4,5,6,7}
>>> S5=S1|S2|S3|S4
```

Example:

```
>>> python_students={'naresh','suresh','ramesh','rajesh'}
>>> java_students={'suresh','kishore','kiran','naresh'}
>>> java_python_students=python_students|java_students
>>> len(java_python_students)
6
>>> print(python_students)
{'ramesh', 'suresh', 'naresh', 'rajesh'}
>>> print(java_students)
{'kiran', 'suresh', 'naresh', 'kishore'}
>>> print(java_python_students)
{'kishore', 'suresh', 'naresh', 'rajesh', 'ramesh', 'kiran'}
```

Example:

```
>>> A={10,20,30}
>>> B=[10,20,30,40]
>>> C=A.union(B)
>>> print(A)
{10, 20, 30}
>>> print(B)
```

```
[10, 20, 30, 40]
>>> print(C)
{40, 10, 20, 30}
>>> D=A|B
Traceback (most recent call last):
  File "<pyshell#79>", line 1, in <module>
    D=A|B
TypeError: unsupported operand type(s) for |: 'set' and 'list'
```

intersection(others*)**
set & other & ...
Return a new set with elements common to the set and all others.

```
>>> A={10,20,30,40}
>>> B={10,40,50,60,70}
>>> C=A.intersection(B)
>>> print(A,B,C,sep="\n")
{40, 10, 20, 30}
{50, 70, 40, 10, 60}
{40, 10}
>>> python_students={'naresh','suresh','ramesh','rajesh'}
>>> java_students={'suresh','kishore','kiran','naresh'}
>>> python_java_students=python_students&java_students
>>> print(python_students)
{'ramesh', 'suresh', 'naresh', 'rajesh'}
>>> print(java_students)
{'kiran', 'suresh', 'naresh', 'kishore'}
>>> print(python_java_students)
{'suresh', 'naresh'}
>>> A={1,2,3,4,5}
>>> B={1,2,3,6,7}
>>> C={1,2,6,7,8,9}
>>> D=A|B&C
>>> print(A)
{1, 2, 3, 4, 5}
>>> print(B)
{1, 2, 3, 6, 7}
>>> print(C)
{1, 2, 6, 7, 8, 9}
```

>>> print(D)
{1, 2, 3, 4, 5, 6, 7}

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

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