Sets
Sets are unordered collections. In unordered collections insertion order is not preserved. The ordered insertion changes from one insertion of element to another. Unordered collection are non sequences or non index based (not support indexing and slicing).
Sets does not allows duplicate elements/items or values.

1. set (mutable)
2. frozenset (immutable)

A set object is an unordered collection of distinct hashable objects. Common uses include membership testing, removing duplicates from a sequence, and computing mathematical operations such as intersection, union, difference, and symmetric difference.

In set objects or elements is organized using hashing data structure.

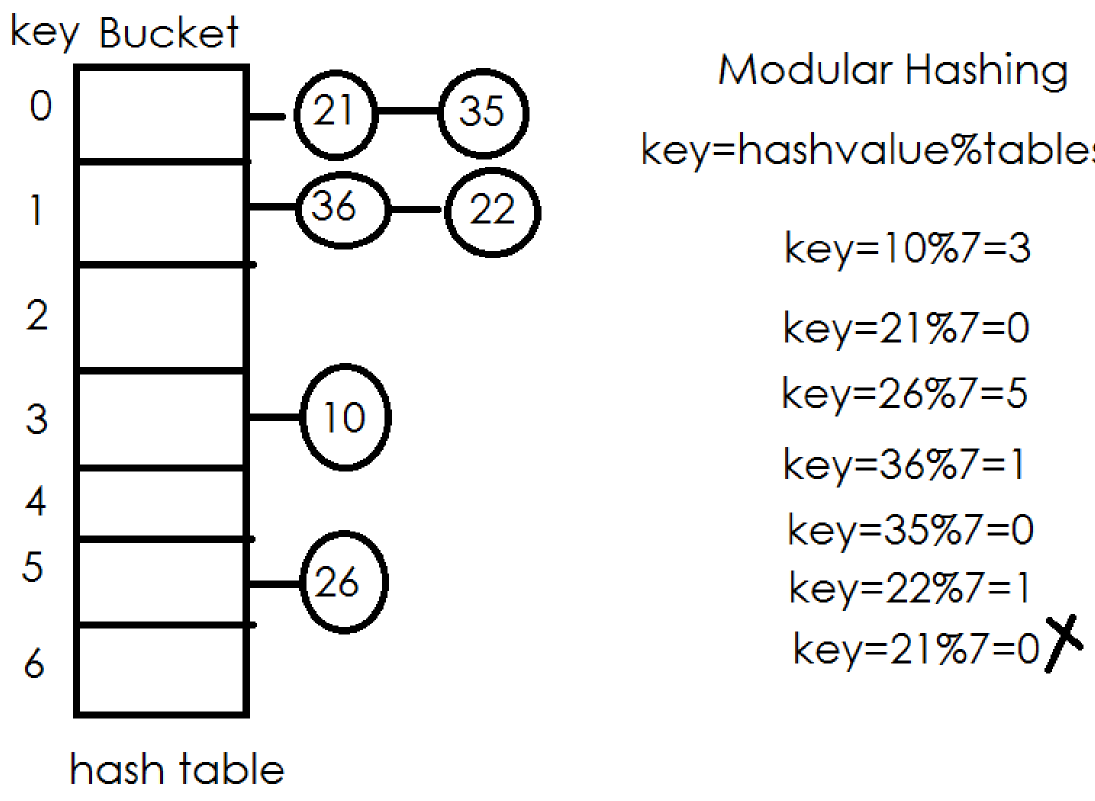
What is hashable object?
An object which generates hash value is called hashable object. All immutable objects are hashable.

What is hash value or hashcode?
Hash value is an integer value, which is used by hash based data structures for finding position or key.
If two objects are equal, they generate same hash value.

```
>>> a=10
>>> b=10
>>> a==b
True
>>> hash(a)
10
>>> hash(b)
10
>>> f1=1.5
>>> f2=1.5
>>> f1==f2
True
>>> hash(f1)
1152921504606846977
```

```
>>> hash(f2)
1152921504606846977
>>> s1="abc"
>>> s2="abc"
>>> s1==s2
True
>>> hash(s1)
-8416937864340885308
>>> hash(s2)
-8416937864340885308
>>> n1=10
>>> n2=20
>>> n1==n2
False
>>> hash(n1)
10
>>> hash(n2)
20
>>> list1=[10,20,30]
>>> list2=[10,20,30]
>>> list1==list2
True
>>> hash(list1)
Traceback (most recent call last):
  File "<pyshell#23>", line 1, in <module>
    hash(list1)
TypeError: unhashable type: 'list'
>>> t1=(10,20)
>>> t2=(10,20)
>>> t1==t2
True
>>> hash(t1)
-4873088377451060145
>>> hash(t2)
-4873088377451060145
```

Hashing Data Structure



How to create set?
Sets can be created by several means:

- Use a comma-separated list of elements within braces: {'jack', 'sjoerd'}
- Use a set comprehension: {c for c in 'abracadabra' if c not in 'abc'}
- Use the type constructor: set(), set('foobar'), set(['a', 'b', 'foo'])

Note: empty curly braces are not represent set, it represent dictionary.

```
>>> set1={}
>>> type(set1)
<class 'dict'>
```

How to create empty set?
Empty set is created using set() function.

```
>>> set1=set()
>>> print(set1)
```

```
set()
>>> print(type(set1))
<class 'set'>
```

Creating set with values/elements?

```
>>> A={10,20,30,40,50}
>>> print(A)
{50, 20, 40, 10, 30}
>>> B={10,10,10,10,10}
>>> print(B)
{10}
>>> C={10,1,2,3,10,20,30,10,20,30,40,40}
>>> print(C)
{1, 2, 3, 20, 40, 10, 30}
>>> D={10,"ABC","10"}
>>> print(D)
{10, 'ABC', '10'}
>>> names={"naresh","suresh","naresh"}
>>> print(names)
{'naresh', 'suresh'}
>>> A={10,20,30,40,50}
>>> print(A)
{50, 20, 40, 10, 30}
>>> A[0]
Traceback (most recent call last):
  File "<pyshell#46>", line 1, in <module>
    A[0]
TypeError: 'set' object is not subscriptable
```

Set() function

This function is used to convert other iterables/collections into set type.

1. Set()
2. Set(iterable)

Example:

```
>>> set1=set()
>>> print(set1)
set()
>>> set2=set([10,20,30,40,10,20,30,40,50,50,50])
>>> print(set2)
{40, 10, 50, 20, 30}
>>> set3=set(range(10,110,10))
>>> print(set3)
{100, 70, 40, 10, 80, 50, 20, 90, 60, 30}
>>> set4=set("java")
>>> print(set4)
{'v', 'j', 'a'}
>>> set5=set({10,20,30,40,50})
>>> print(set5)
{50, 20, 40, 10, 30}
>>> set6=set((10,20,30,40,50))
>>> print(set6)
{40, 10, 50, 20, 30}
```

How to read elements from set?
Reading elements from set are done in different easy

1. Using for loop
2. Using iterator
3. Using enumerate

Reading elements from set using for loop

```
A={10,20,30,40,50}
for value in A:
   print(value)
```

Output:

```
50
20
40
10
30
```

Reading elements from set using iterator object

```
A={10,20,30,40,50}
x=iter(A)
value1=next(x)
value2=next(x)
print(value1,value2)
```

Output:

```
50 20
```

Reading elements from set using enumerate object

```
A={10,20,30,40,50}
x=enumerate(A)
t1=next(x)
t2=next(x)
print(t1)
print(t2)
```

Output:

```
(0, 50)
(1, 20)
```