Sets

Honey Pasricha

Eklavya Academy

www.ekaim.in

Definition

A set contains a collection of unique values and works like a mathematical set.

Or

A *set* is an object that stores a collection of data in the same way as mathematical sets.

Creating a Set

To create a set, you have to call the built-in set function.
Here is an example of how you create an empty set:

```
myset = set()  #empty set can be created using set function
```

Or

```
Myset={1,2,3,4,4,5} # it should not be empty
```

Some important rules to know

All the elements in a set must be unique. No two elements can have the same value.

- Sets are unordered, which means that the elements in a set are not stored in any particular order.
- The elements that are stored in a set can be of different data types.

Creating set through iterable elements

```
myset = set(['a', 'b', 'c'])

myset = set('abc')


# This is an ERROR!
myset = set('one', 'two', 'three')
```

Adding elements to set

Setname.add('nameofelement')

```
>>> myset = set()
>>> myset.add(1)
>>> myset.add(2)
```

Update the set

You can add a group of elements to a set all at one time with the update method. When you call the update method as an argument, you pass an object that contains iterable elements, such as a list, a tuple, string, or another set.

```
>>> myset = set([1, 2, 3])
>>> myset.update([4, 5, 6])
>>> myset e
 {1, 2, 3, 4, 5, 6}
```

Remove the element

```
>>>myset = set([1, 2, 3, 4, 5])
>>> myset
 {1, 2, 3, 4, 5}
 >>> myset.remove(1)
 >>> myset
                                >>> myset.discard(5)
```

Clear the elements

```
1 >>> myset = set([1, 2, 3, 4, 5])
2 >>> myset
3 {1, 2, 3, 4, 5}
4 >>> myset.clear()
5 >>> myset
6 set()
```

for Loop to Iterate over a Set

You can use the for loop in the following general format to iterate over all the elements in a set:

for *var* in *set*:

    statement
    statement
    etc.

Using the in and not in Operators to Test for a Value in a Set

```
1 >>> myset = set([1, 2, 3])
2 >>> if 1 in myset:
3         print('The value 1 is in the set.')
```

Finding the Union of Sets

The union of two sets is a set that contains all the elements of both sets. In Python, you can call the union method to get the union of two sets. Here is the general format:

set1.union(*set2*)

set1 | *set2*

```
1 >>> set1 = set([1, 2, 3, 4])
2 >>> set2 = set([3, 4, 5, 6])
3 >>> set3 = set1.union(set2)
4 >>> set3
5 {1, 2, 3, 4, 5, 6}
```

Intersection of Sets

The intersection of two sets is a set that contains only the elements that are found in both sets.

```
1 >>> set1 = set([1, 2, 3, 4])
2 >>> set2 = set([3, 4, 5, 6])
3 >>> set3 = set1.intersection(set2)
4 >>> set3
5 {3, 4}
```

set1 & set2

Difference of Sets

```
1 >>> set1 = set([1, 2, 3, 4])
2 >>> set2 = set([3, 4, 5, 6])
3 >>> set3 = set1.difference(set2)
```

To find the Subsets and SuperSets

you have two sets and one of those sets contains all of the elements of the other
set. Here is an example:

```
set1 = set([1, 2, 3, 4])
set2 = set([2, 3])
```

you can call the issubset method to determine whether one set is a subset of another. Here is the general format:

set2.issubset(*set1*)

To find the Subsets and SuperSets

Here is an example:

set1 = set([1, 2, 3, 4])
set2 = set([2, 3])

set2 <= *set1*

1 >>> set1 = set([1, 2, 3, 4])
2 >>> set2 = set([2, 3])
3 >>> set2.issubset(set1)
4 True
5 >>> set1.issuperset(set2)
6 True