Set Data type

Set is also a collection data type in Python. However, it is **not an ordered collection of objects**, like list or tuple. Hence, indexing and slicing operations cannot be done on a set object. A set also doesn't allow duplicate objects to be stored, where as in list and tuple, the same object can appear more than once. Even if an object is put more than once in a set, only one copy is held. Set is a Python implementation of a set as defined in Mathematics. The set object has suitable methods to perform mathematical set operations like union, intersection, difference etc. A set object contains one or more items, not necessarily of the same type which are separated by comma and enclosed in curly brackets {}.

```
>>> S1={1, "Ravi", 75.50}
>>> S1
{1, 75.5, 'Ravi'}
>>> type(S1)
<class 'set'
>>>> S2={10,23,40,23,50,10}
>>> S2
{40, 10, 50, 23}
>>>
```

set() function

Python has an in-built function set() using which set object can be constructed out of any sequence like string, list or tuple object.

```
>>> S1=set("Internshala")
>>> S1
{'t', 'n', 's', 'h', 'e', 'a', 'l', 'I', 'r'}
>>> S2=set([45,67,87,36,55])
>>> S2
{55, 67, 36, 45, 87}
>>> S3=set((10,25,15))
```

```
>>> S3
{25, 10, 15}
>>>
```

Order of elements in the set is not necessarily the same that is given at the time of assignment. Python optimizes the structure for performing operations over set as defined in mathematics. Only immutable (and hashable) objects can be a part of set object. Numbers (integer, float as well as complex), strings, and tuple objects are accepted but list and dictionary objects are not.

```
>>> S1={(10,10), 10,20}
>>> S1
{10, 20, (10, 10)}
>>> S2={[10,10], 10,20}
Traceback (most recent call last):
  File "<pyshell#2>", line 1, in <module>
    S2={[10,10], 10,20}
TypeError: unhashable type: 'list'
>>>
```

In first case, (10,10) is a tuple, hence it becomes part of set. In second example though, since [10,10] is a list, error message is displayed saying list is unhashable. (Hashing is a mechanism in computer science which enables quicker searching of objects in computer's memory. https://en.wikipedia.org/wiki/Hash_function) Even though mutable objects are not stored in a set, set itself is a mutable object. A set object can be modified by add(), update(), remove() and discard() methods.

add()

adds a new element in set object

```
>>> S1=set({"Python", "Java", "C++"})
>>> S1
{'Python', 'C++', 'Java'}
>>> S1.add("Perl")
>>> S1
```

```
{'Perl', 'Python', 'C++', 'Java'}
>>>
```

update()

adds multiple items from a list or tuple

```
>>> S1={"Python", "Java", "C++"}
>>> S1.update(["C++", "Basic"])
>>> S1
{'C++', 'Java', 'Python', 'Basic'}
>>> S1.update(["Ruby", "PHP"])
>>> S1
{'Ruby', 'PHP', 'Java', 'C++', 'Python', 'Basic'}
>>>
```

clear()

Removes contents of set object results in an empty set

```
>>> S1.clear()
>>> S1
set()
>>>
```

copy()

Creates a copy of set object

```
>>> S1={"Python", "Java", "C++"}
>>> S2=S1.copy()
>>> S2
{'C++', 'Java', 'Python'}
>>>
```

discard()

Returns set after removing an item from it. No changes are done if the item is not present

```
>>> S1={"Python", "Java", "C++"}
```

```
>>> S1.discard("Java")
>>> S1
{'C++', 'Python'}
>>> S1.discard("SQL")
>>> S1
{'C++', 'Python'}
>>>
```

remove()

Returns set after removing an item from it. Results in error if the item is not present

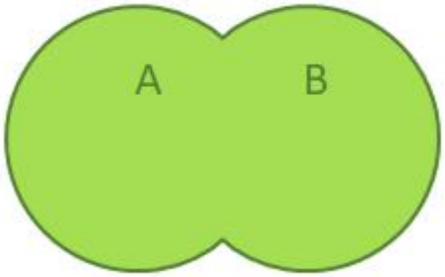
```
>>> S1={"Python", "Java", "C++"}
>>> S1.remove("C++")
>>> S1
{'Java', 'Python'}
>>> S1.remove("SQL")
Traceback (most recent call last):
  File "<pyshell#3>", line 1, in <module>
    S1.remove("SQL")
KeyError: 'SQL'
>>>
```

Set Operations

As mentioned earlier, set data type in Python implements set as defined in mathematics. Various Set operations can be performed using Python's set obect. The operators |, &, - and ^ perform union, intersection, difference and symmetric difference operations respectively. Each of these operators have a corresponding method associated with built-in set class.

Union

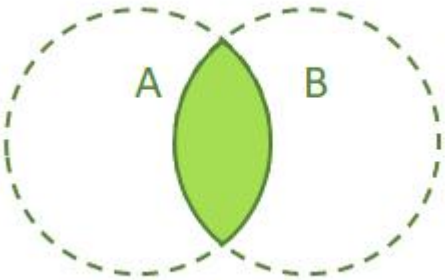
Union of two sets is a set of all elements in both.




```
>>> s1={1,2,3,4,5}
>>> s2={4,5,6,7,8}
>>> s1|s2
{1, 2, 3, 4, 5, 6, 7, 8}
>>> s1.union(s2)
{1, 2, 3, 4, 5, 6, 7, 8}
>>>
```

Intersection

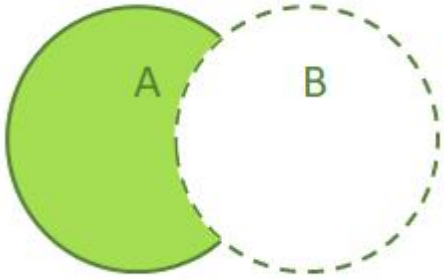
Intersection of two sets is a set containing elements common to both




```
>>> s1={1,2,3,4,5}
>>> s2={4,5,6,7,8}
>>> s1&s2
{4, 5}
>>> s1.intersection(s2)
{4, 5}
>>>
```

Difference

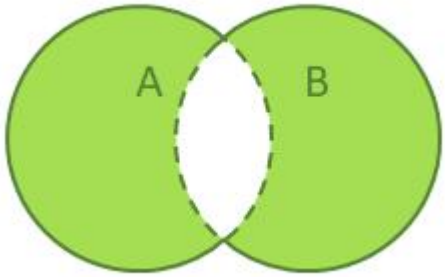
Difference of two sets results in a set containing elements only in first but not in second set.




```
>>> s1={1,2,3,4,5}
>>> s2={4,5,6,7,8}
>>> s1-s2
{1, 2, 3}
>>> s2-s1
{8, 6, 7}
>>> s1.difference(s2)
{1, 2, 3}
>>> s2.difference(s1)
{8, 6, 7}
>>>
```

Symmetric Difference

Result of Symmetric difference is a set consisting of elements in both sets excluding common elements




```
>>> s1={1,2,3,4,5}
>>> s2={4,5,6,7,8}
>>> s1^s2
{1, 2, 3, 6, 7, 8}
>>> s2^s1
{1, 2, 3, 6, 7, 8}
>>> s1.symmetric_difference(s2)
{1, 2, 3, 6, 7, 8}
>>> s2.symmetric_difference(s1)
{1, 2, 3, 6, 7, 8}
>>>
```

Set is a specialized data type. One of the major applications of Python is in area of mathematical computing and data analysis in which set operations are important. We may drop this discussion considering it as not for beginner (and also to curtail the size), but learner (especially who intends to go in mathematical and scientific computing) should be encouraged to explore this section. We should emphasize this and provide this as a text for further reading.