Sets

It is an unordered, unchangable assciative data structure (that is unindexed data structure).

1. Creating Sets: It is created by mentioning the list of items within the curly bracket.

Dict_list = {value1, value2, ...}

```
courses = {'Physics', 'Maths' , 'Chemistry','z'}
print(courses)
```

```
{'Physics', 'z', 'Maths', 'Chemistry'}
```

It does not allow duplication.

```
courses = {'Physics', 'Maths' , 'Chemistry', 'Maths' }
print(courses)
```

```
{'Maths', 'Physics', 'Chemistry'}
```

Values of any data type and mixed data type are allowed.

```
# Set of integers
int_set ={ 2 ,4 ,5}
print(int_set)
```

```
{2, 4, 5}
```

```
# Set of Float values
float_set ={ 2.3 ,4.5 ,5.2}
print(float_set)
```

```
{2.3, 4.5, 5.2}
```

```
mixed = {'Physics', 23 , 'Chemistry', True }
print(mixed)
```

```
{True, 'Physics', 'Chemistry', 23}
```

Sets can be created using **constructor set()**. In this case parenthesis will be used in place of curly brackets.

```
mixed = set(('Physics', 23 , 'Chemistry', True ))
print(mixed)
```

```
{True, 'Physics', 'Chemistry', 23}
```

Items in the Set cannot be accessed by referring to an index or a key.

Items can be access using in operator:

```
# Using in operator to find if particular item is present in the set or not
mixed = set(('Physics', 213 , 'Chemistry', True ))
213 in mixed
```

```
True
```

```
# Using in operator to print all the items present in the set
mixed = set(('Physics', 23 , 'Chemistry', True ))
```

```
for x in mixed:
  print(x)
```

```
True
Chemistry
Physics
23
```

Once a set is created, value of the items cannot be changed. However, items can be deleted and new items can be added.

New items in the set can be added using a method add()

```
mixed = set(('Physics', 23 , 'Chemistry', True ))
print("Before:", mixed)
mixed.add(56)
print("After:", mixed)
```

```
Before: {True, 'Physics', 'Chemistry', 23}
After: {True, 'Chemistry', 23, 56, 'Physics'}
```

To combine the items in the two sets, use **method update()**

```
set1 = set(('Physics', 23 , 'Chemistry', True ))
set2 ={ 23 ,4.5 ,5.2}
print("Before:", set1)
set1.update(set2)
print("After:", set1)
```

```
Before: {True, 'Chemistry', 'Physics', 23}
After: {True, 'Chemistry', 'Physics', 4.5, 23, 5.2}
```

Set2 to be copmbined with set1 can be a list, tuple, or set.

```
set1 = set(('Physics', 23 , 'Chemistry', True ))
list1 =[ 2.3 ,4.5 ,5.2]
print("Before:", set1)
set1.update(list1)
print("After:", set1)
```

```
Before: {True, 'Chemistry', 'Physics', 23}
After: {True, 2.3, 4.5, 'Physics', 5.2, 'Chemistry', 23}
```

Items from the list can be removed by using methods: remove or discard.

Remove method will pop up error message if the item to be deleted is not present in the set.

discard method will NOT pop up error message if the item to be deleted is not present in the set.

```
set1 = set(('Physics', 23 , 'Chemistry', True ))
print("Before:", set1)
set1.remove(True)
print("After:", set1)
```

```
Before: {True, 'Chemistry', 'Physics', 23}
After: {'Chemistry', 'Physics', 23}
```

```
set1 = set(('Physics', 23 , 'Chemistry', True ))
print("Before:", set1)
set1.remove(False)
print("After:", set1)
```

```
Before: {True, 'Chemistry', 'Physics', 23}
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
<ipython-input-7-c784e7a50787> in <module>
      1 set1 = set(('Physics', 23 , 'Chemistry', True ))
      2 print("Before:", set1)
----> 3 set1.remove(False)
      4 print("After:", set1)
```

```
set1 = set(('Physics', 23 , 'Chemistry', True ))
print("Before:", set1)
set1.discard(True)
print("After:", set1)
```

```
Before: {True, 'Chemistry', 'Physics', 23}
After: {'Chemistry', 'Physics', 23}
```

```
set1 = set(('Physics', 23 , 'Chemistry', True ))
print("Before:", set1)
set1.discard(False)
print("After:", set1)
```

```
Before: {True, 'Chemistry', 'Physics', 23}
After: {True, 'Chemistry', 'Physics', 23}
```

Pop() method will remove the last item from the set. But since the set is an unordered list, so we do not know which item is present at the end. SO pop(0 should not be used.

```
set1 = set(('Physics', 23 , 'Chemistry', True ))
print("Before:", set1)
set1.pop()
print("After:", set1)
```

```
set1.pop()
print("After:", set1)
```

```
Before: {True, 'Physics', 'Chemistry', 23}
After: {'Physics', 'Chemistry', 23}
After: {'Chemistry', 23}
```

Clear() method will remove all the items from the set,

```
set1 = set(('Physics', 23 , 'Chemistry', True ))
print("Before:", set1)
set1.clear()
print("After:", set1)
```

```
Before: {True, 'Physics', 'Chemistry', 23}
After: set()
```

Operation on Sets:

1. union
2. intersection
3. Difference
4. Getting list of elements not common in the two sets.

1. Union: A U B

Method union(): Returns new set containing items form the two sets.

```
set1 = set(('Physics', 23 , 'Chemistry', True ))
set2 ={ 23 ,4.5 ,5.2}
```

```
print("Before:", set1)
set1.union(set2)
print("After:", set1)
set3=set1.union(set2)
print("New Set:", set3)
```

```
    Before: {'Physics', True, 'Chemistry', 23}
    After: {'Physics', True, 'Chemistry', 23}
    New Set: {True, 23, 4.5, 5.2, 'Chemistry', 'Physics'}
```

update(): Updates set1 with the items of set2.

```
set1 = set(('Physics', 23 , 'Chemistry', True ))
set2 ={ 23 ,4.5 ,5.2}
print("Before:", set1)
set1.update(set2)
print("After:", set1)
```

```
    Before: {'Physics', True, 'Chemistry', 23}
    After: {True, 23, 4.5, 5.2, 'Chemistry', 'Physics'}
```

2. Intersection: A ∩ B

intersection_update(): Updates set1 and keep the common items present in both the sets.

```
set1 = set(('Physics', 23 , 'Chemistry', True ))
set2 ={ 23 ,4.5 ,5.2, True}
print("Before:", set1)
set1.intersection_update(set2)
print("After:", set1)
```

```
    Before: {'Physics', True, 'Chemistry', 23}
```

```
After: {True, 23}
```

intersection(): Returns new set that contains common items present in both the sets.

```
set1 = set(('Physics', 23 , 'Chemistry', True ))
set2 ={ 23 ,4.5 ,5.2, True}
print("Before:", set1)
set1.intersection(set2)
print("After:", set1)
set3=set1.intersection(set2)
print("New Set:", set3)
```

```
Before: {'Physics', True, 'Chemistry', 23}
After: {'Physics', True, 'Chemistry', 23}
New Set: {True, 23}
```

3. AUB - A∩B

symmetric_difference_update(): Updates set1 and keep the items that are not common in both the sets.

```
set1 = set(('Physics', 23 , 'Chemistry', True ))
set2 ={ 23 ,4.5 ,5.2, True}
print("Before:", set1)
set1.symmetric_difference_update(set2)
print("After:", set1)
```

```
Before: {'Physics', True, 'Chemistry', 23}
After: {4.5, 5.2, 'Physics', 'Chemistry'}
```

symmetric_difference(): Returns new set that contains items from the two sets that are not common.

```
set1 = set(('Physics', 23 , 'Chemistry', True ))
set2 ={ 23 ,4.5 ,5.2, True}
print("Before:", set1)
set1.symmetric_difference(set2)
print("After:", set1)
set3=set1.symmetric_difference(set2)
print("New Set:", set3)
```

```
Before: {'Physics', True, 'Chemistry', 23}
After: {'Physics', True, 'Chemistry', 23}
New Set: {4.5, 5.2, 'Physics', 'Chemistry'}
```

4. A-B

```
set1 = set(('Physics', 23 , 'Chemistry', True ))
set2 ={ 23 ,4.5 ,5.2, True}
print(set1.difference(set2))
```

```
{'Physics', 'Chemistry'}
```