SET

---

1. unordered
2. unchangeable*
3. unindexed
4. Duplicates Not Allowed

- Set items are unchangeable, but you can remove items and add new items.

Creat a list

```
myset = {"apple", "banana", "cherry"}
print(myset)
```

```
{'apple', 'banana', 'cherry'}
```

The set() Constructor

It is also possible to use the set() constructor to make a set.

```
# Using the set() constructor to make a set:

thisset = set(("apple", "banana", "cherry")) # note the double round-brackets
print(thisset)
```

```
{'apple', 'banana', 'cherry'}
```

Get the Length of a Set

To determine how many items a set has, use the len() function.

```
# Get the number of items in a set:

thisset = {"apple", "banana", "cherry"}

print(len(thisset))
```

```
3
```

Access Set Items

Access Items

You cannot access items in a set by referring to an index or a key.

But you can loop through the set items using a for loop, or ask if a specified value is present in a set, by using the in keyword.

```
  # Loop through the set, and print the values:

  thisset = {"apple", "banana", "cherry"}

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

```
    apple
    banana
    cherry
```

```
# Check if "banana" is present in the set:

thisset = {"apple", "banana", "cherry"}

print("banana" in thisset)
```

```
    True
```

Add Set Items

Add Items

Once a set is created, you cannot change its items, but you can add new items.

To add one item to a set use the add() method.

```
# Add an item to a set, using the add() method:

thisset = {"apple", "banana", "cherry"}

thisset.add("orange")

print(thisset)
```

```
    {'apple', 'banana', 'orange', 'cherry'}
```

Add Sets

To add items from another set into the current set, use the update() method.

```
# Add elements from tropical into thisset

thisset = {"apple", "banana", "cherry"}
tropical = {"pineapple", "mango", "papaya"}

thisset.update(tropical)

print(thisset)
```

```
{'apple', 'papaya', 'pineapple', 'mango', 'banana', 'cherry'}
```

Add Any Iterable

The object in the update() method does not have to be a set, it can be any iterable object (tuples, lists, dictionaries etc.).

```
# Add elements of a list to at set:

thisset = {"apple", "banana", "cherry"}
mylist = ["kiwi", "orange"]

thisset.update(mylist)

print(thisset)
```

```
{'apple', 'banana', 'cherry', 'kiwi', 'orange'}
```

Remove Set Items

Remove Item

To remove an item in a set, use the remove(), or the discard() method.

```
# Remove "banana" by using the remove() method:

thisset = {"apple", "banana", "cherry"}

thisset.remove("banana")

print(thisset)
```

```
{'apple', 'cherry'}
```

Note: If the item to remove does not exist, remove() will raise an error.

```
  # Remove "banana" by using the discard() method:

  thisset = {"apple", "banana", "cherry"}

  thisset.discard("banana")

  print(thisset)
```

```
     {'apple', 'cherry'}
```

Note: If the item to remove does not exist, discard() will NOT raise an error.

You can also use the pop() method to remove an item, but this method will remove the last item. Remember that sets are unordered, so you will not know what item that gets removed.

The return value of the pop() method is the removed item.

```
# Remove the last item by using the pop() method:

thisset = {"apple", "banana", "cherry"}
x = thisset.pop()
print(x)

print(thisset)
```

```
     apple
     {'banana', 'cherry'}
```

Note: Sets are unordered, so when using the pop() method, you do not know which item that gets removed.

```
# The clear() method empties the set:

thisset = {"apple", "banana", "cherry"}
thisset.clear()
print(thisset)
```

```
     set()
```

```
# The del keyword will delete the set completely:

thisset = {"apple", "banana", "cherry"}
del thisset
print(thisset)
```

Loop Sets

Loop Items

You can loop through the set items by using a for loop.

```
# Loop through the set, and print the values:

thisset = {"apple", "banana", "cherry"}

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

```
apple
banana
cherry
```

Join Sets

Join Two Sets

There are several ways to join two or more sets in Python.

You can use the union() method that returns a new set containing all items from both sets, or the update() method that inserts all the items from one set into another.

```
  # The union() method returns a new set with all items from both sets:

  set1 = {"a", "b" , "c"}
  set2 = {1, 2, 3}

  set3 = set1.union(set2)
  print(set3)
```

```
{1, 'c', 2, 'a', 3, 'b'}
```

```
# The update() method inserts the items in set2 into set1:

set1 = {"a", "b" , "c"}
set2 = {1, 2, 3}
```

```
set1.update(set2)
print(set1)
```

```
{1, 'c', 2, 'a', 3, 'b'}
```

Note: Both union() and update() will exclude any duplicate items.

Set Methods

Python has a set of built-in methods that you can use on sets.

Method	Description
add()	Adds an element to the set
clear()	Removes all the elements from the set
copy()	Returns a copy of the set
difference()	Returns a set containing the difference between two or more sets
difference_update()	Removes the items in this set that are also included in another, specified set
discard()	Remove the specified item
intersection()	Returns a set, that is the intersection of two other sets
intersection_update()	Removes the items in this set that are not present in other, specified set(s)
isdisjoint()	Returns whether two sets have a intersection or not
issubset()	Returns whether another set contains this set or not
issuperset()	Returns whether this set contains another set or not
pop()	Removes an element from the set
remove()	Removes the specified element
symmetric_difference()	Returns a set with the symmetric differences of two sets
symmetric_difference_update()	inserts the symmetric differences from this set and another
union()	Return a set containing the union of sets
update()	Update the set with the union of this set and others