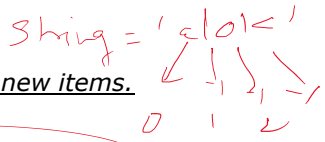
SET

10 July 2024 07:32 PM

Set

sets are used to store multiple items in a single variable.
Set is one of 4 built-in data types in Python used to store collections of data, the other 3 are List, Tuple, and Dictionary, all with different qualities and usage.

A set is a collection which is *unordered*, *unchangeable**, and *unindexed*.



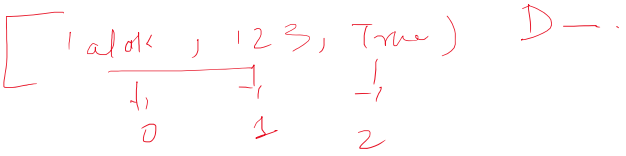
***Note:** Set items are unchangeable, but you can remove items and add new items.*
Sets are written with curly brackets.

Example

Create a Set:

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

print(thisset)
```



***Note:** Sets are unordered, so you cannot be sure in which order the items will appear.*

Set Items

Set items are unordered, unchangeable, and do not allow duplicate values.

Unordered

Unordered means that the items in a set do not have a defined order.
Set items can appear in a different order every time you use them, and *cannot be referred to by index or key.*

Unchangeable

Set items are unchangeable, meaning that we *cannot change the items after the set has been created.*
Once a set is created, you cannot change its items, *but you can remove items and add new items.*

Duplicates Not Allowed

Sets cannot have two items with the same value.

Example

Duplicate values will be ignored:

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


print(thisset)
```

Note: The values `True` and `1` are considered the same value in sets, and are treated as duplicates:

Example

`True` and `1` is considered the same value:

```
thisset = {"apple", "banana", "cherry", True, 1, 2}

print(thisset)
```

Note: The values `False` and `0` are considered the same value in sets, and are treated as duplicates:

Example

`False` and `0` is considered the same value:

```
thisset = {"apple", "banana", "cherry", False, True, 0}

print(thisset)
```

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.

Example

Loop through the set, and print the values:

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

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

Example

Check if "banana" is present in the set:

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

print("banana" in thisset)
```

Example

Check if "banana" is NOT present in the set:

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

print("banana" not in thisset)
```

Change Items

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

Example

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

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

```
thisset.add("orange")

print(thisset)
```

Add Sets

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

Example

Add elements from `tropical` into `thisset`:

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

tropical = {"pineapple", "mango", "papaya"}

thisset.update(tropical)

print(thisset)
```

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.).

Example

Add elements of a list to at set:

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

mylist = ["kiwi", "orange"]

thisset.update(mylist)

print(thisset)
```

Remove Item

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

Example

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

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

thisset.remove("banana")

print(thisset)
```

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

Example

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

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


thisset.discard("banana")


print(thisset)
```

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 a random item, so you cannot be sure what item that gets removed.
The return value of the `pop()` method is the removed item.

Example

Remove a random item by using the `pop()` method:

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


x = thisset.pop()


print(x)


print(thisset)
```

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

Example

The `clear()` method empties the set:

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


thisset.clear()


print(thisset)
```

Example

The `del` keyword will delete the set completely:

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


del thisset


print(thisset)
```