Python Sets

Previous Next

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

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

Get your own Python Server

Create a Set:

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

Try it Yourself »

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.

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)
```

Try it Yourself »

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)
```

Try it Yourself »

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)
```

Try it Yourself »

Get the Length of a Set

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

Example

Get the number of items in a set:

Try it Yourself »

Set Items - Data Types

Set items can be of any data type:

Example

String, int and boolean data types:

```
set1 = {"apple", "banana", "cherry"}
set2 = {1, 5, 7, 9, 3}
set3 = {True, False, False}
```

Try it Yourself »

A set can contain different data types:

Example

A set with strings, integers and boolean values:

```
set1 = {"abc", 34, True, 40, "male"}
```

Try it Yourself »

type()

From Python's perspective, sets are defined as objects with the data type 'set':

```
<class 'set'>
```

Example

What is the data type of a set?

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

Try it Yourself »

The set() Constructor

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

Example

Using the set() constructor to make a set:

Python Collections (Arrays)

There are four collection data types in the Python programming language:

- **List** is a collection which is ordered and changeable. Allows duplicate members.
- **Tuple** is a collection which is ordered and unchangeable. Allows duplicate members.
- **Set** is a collection which is unordered, unchangeable*, and unindexed. No duplicate members.
- **Dictionary** is a collection which is ordered** and changeable. No duplicate members.

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

**As of Python version 3.7, dictionaries are *ordered*. In Python 3.6 and earlier, dictionaries are *unordered*.

When choosing a collection type, it is useful to understand the properties of that type. Choosing the right type for a particular data set could mean retention of meaning, and, it could mean an increase in efficiency or security.

❮ Previous Next ❯