

Python Sets

```
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

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)
```

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:

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

```
print(len(thisset))
```

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}
```

A set can contain different data types:

Example

A set with strings, integers and boolean values:

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

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))
```

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:

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

```
print(thisset)
```

*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.

question?

Which one of these is a set?

```
myset = ('apple', 'banana', 'cherry')
```

```
myset = ['apple', 'banana', 'cherry']
```

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

Python - 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.

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.

Python - 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.

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)
```