**SETS:**

1. Sets are used to store multiple items in a single variable.
2. Set is one of 4 built-in data types in Python used to store collections of data, the other 3 are [List](https://www.w3schools.com/python/python_lists.asp), [Tuple](https://www.w3schools.com/python/python_tuples.asp), and [Dictionary](https://www.w3schools.com/python/python_dictionaries.asp), all with different qualities and usage.
3. A set is a collection which is *unordered*, *unchangeable\**, and *unindexed*.
4. **Note:** Set *items* are unchangeable, but you can remove items and add new items.

places = {"Berlin", "New-York", "Lonavala"}

# Python - Add Set Items

places.add("money")

* Set Items

1. Set items are unordered, unchangeable, and do not allow duplicate values.
   1. Unordered:
      1. Unordered means that the items in a set do not have a defined order.
      2. Set items can appear in a different order every time you use them, and cannot be referred to by index or key.
   2. Unchangeable:
      1. Set items are unchangeable, meaning that we cannot change the items after the set has been created.
      2. Once a set is created, you cannot change its items, but you can remove items and add new items.
   3. Duplicates Not Allowed:
      1. Sets cannot have two items with the same value.
      2. **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:
      3. **Note:** The values False and 0 are considered the same value in sets, and are treated as duplicates:

# True and 1 is considered the same value:

x = {"apple", "banana", "cherry", True}

y = {"google", 1, "apple", 2}

z = x.symmetric\_difference(y)

1. print(z)

Get the Length of a Set:

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

print(len(thisset))

**Set Items - Data Types:**

1. Set items can be of any data type:

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

1. set1 = {"abc", 34, True, 40, "male"}

**The set() Constructor:**

1. It is also possible to use the set() constructor to make a set.
2. Example: Using the set() constructor to make a set:

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

**Python Collections (Arrays):**

* There are four collection data types in the Python programming language:
* [**List**](https://www.w3schools.com/python/python_lists.asp) is a collection which is ordered and changeable. Allows duplicate members.
* [**Tuple**](https://www.w3schools.com/python/python_tuples.asp) 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**](https://www.w3schools.com/python/python_dictionaries.asp) 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.

Set Methods

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

|  |  |
| --- | --- |
| **Method** | **Description** |
| [add()](https://www.w3schools.com/python/ref_set_add.asp) | Adds an element to the set |
| [clear()](https://www.w3schools.com/python/ref_set_clear.asp) | Removes all the elements from the set |
| [copy()](https://www.w3schools.com/python/ref_set_copy.asp) | Returns a copy of the set |
| [difference()](https://www.w3schools.com/python/ref_set_difference.asp) | Returns a set containing the difference between two or more sets |
| [difference\_update()](https://www.w3schools.com/python/ref_set_difference_update.asp) | Removes the items in this set that are also included in another, specified set |
| [discard()](https://www.w3schools.com/python/ref_set_discard.asp) | Remove the specified item |
| [intersection()](https://www.w3schools.com/python/ref_set_intersection.asp) | Returns a set, that is the intersection of two other sets |
| [intersection\_update()](https://www.w3schools.com/python/ref_set_intersection_update.asp) | Removes the items in this set that are not present in other, specified set(s) |
| [isdisjoint()](https://www.w3schools.com/python/ref_set_isdisjoint.asp) | Returns whether two sets have a intersection or not |
| [issubset()](https://www.w3schools.com/python/ref_set_issubset.asp) | Returns whether another set contains this set or not |
| [issuperset()](https://www.w3schools.com/python/ref_set_issuperset.asp) | Returns whether this set contains another set or not |
| [pop()](https://www.w3schools.com/python/ref_set_pop.asp) | Removes an element from the set |
| [remove()](https://www.w3schools.com/python/ref_set_remove.asp) | Removes the specified element |
| [symmetric\_difference()](https://www.w3schools.com/python/ref_set_symmetric_difference.asp) | Returns a set with the symmetric differences of two sets |
| [symmetric\_difference\_update()](https://www.w3schools.com/python/ref_set_symmetric_difference_update.asp) | inserts the symmetric differences from this set and another |
| [union()](https://www.w3schools.com/python/ref_set_union.asp) | Return a set containing the union of sets |
| [update()](https://www.w3schools.com/python/ref_set_update.asp) | Update the set with the union of this set and others |