4.9. Set Types — [set](https://docs.python.org/3/library/stdtypes.html#set), [frozenset](https://docs.python.org/3/library/stdtypes.html" \l "frozenset" \o "frozenset)[¶](https://docs.python.org/3/library/stdtypes.html#set-types-set-frozenset)

A *set* object is an unordered collection of distinct [hashable](https://docs.python.org/3/glossary.html" \l "term-hashable) objects. Common uses include membership testing, removing duplicates from a sequence, and computing mathematical operations such as intersection, union, difference, and symmetric difference. (For other containers see the built-in [dict](https://docs.python.org/3/library/stdtypes.html" \l "dict" \o "dict), [list](https://docs.python.org/3/library/stdtypes.html#list), and [tuple](https://docs.python.org/3/library/stdtypes.html#tuple) classes, and the [collections](https://docs.python.org/3/library/collections.html" \l "module-collections" \o "collections: Container datatypes)module.)

Like other collections, sets support x in set, len(set), and for x in set. Being an unordered collection, sets do not record element position or order of insertion. Accordingly, sets do not support indexing, slicing, or other sequence-like behavior.

There are currently two built-in set types, [set](https://docs.python.org/3/library/stdtypes.html#set) and [frozenset](https://docs.python.org/3/library/stdtypes.html" \l "frozenset" \o "frozenset). The [set](https://docs.python.org/3/library/stdtypes.html#set) type is mutable — the contents can be changed using methods like add() and remove(). Since it is mutable, it has no hash value and cannot be used as either a dictionary key or as an element of another set. The [frozenset](https://docs.python.org/3/library/stdtypes.html" \l "frozenset" \o "frozenset) type is immutable and [hashable](https://docs.python.org/3/glossary.html" \l "term-hashable) — its contents cannot be altered after it is created; it can therefore be used as a dictionary key or as an element of another set.

Non-empty sets (not frozensets) can be created by placing a comma-separated list of elements within braces, for example: {'jack', 'sjoerd'}, in addition to the [set](https://docs.python.org/3/library/stdtypes.html" \l "set" \o "set)constructor.

The constructors for both classes work the same:

*class***set**([*iterable*])

*class***frozenset**([*iterable*])

Return a new set or frozenset object whose elements are taken from *iterable*. The elements of a set must be [hashable](https://docs.python.org/3/glossary.html" \l "term-hashable). To represent sets of sets, the inner sets must be [frozenset](https://docs.python.org/3/library/stdtypes.html" \l "frozenset" \o "frozenset) objects. If *iterable* is not specified, a new empty set is returned.

Instances of [set](https://docs.python.org/3/library/stdtypes.html#set) and [frozenset](https://docs.python.org/3/library/stdtypes.html" \l "frozenset" \o "frozenset) provide the following operations:

**len(s)**

Return the number of elements in set *s* (cardinality of *s*).

**x in s**

Test *x* for membership in *s*.

**x not in s**

Test *x* for non-membership in *s*.

**isdisjoint**(*other*)

Return True if the set has no elements in common with *other*. Sets are disjoint if and only if their intersection is the empty set.

**issubset**(*other*)

**set <= other**

Test whether every element in the set is in *other*.

**set < other**

Test whether the set is a proper subset of *other*, that is, set <= other and set != other.

**issuperset**(*other*)

**set >= other**

Test whether every element in *other* is in the set.

**set > other**

Test whether the set is a proper superset of *other*, that is, set >= other and set != other.

**union**(*\*others*)

**set | other | ...**

Return a new set with elements from the set and all others.

**intersection**(*\*others*)

**set & other & ...**

Return a new set with elements common to the set and all others.

**difference**(*\*others*)

**set - other - ...**

Return a new set with elements in the set that are not in the others.

**symmetric\_difference**(*other*)

**set ^ other**

Return a new set with elements in either the set or *other* but not both.

**copy**()

Return a new set with a shallow copy of *s*.

Note, the non-operator versions of [union()](https://docs.python.org/3/library/stdtypes.html#frozenset.union), [intersection()](https://docs.python.org/3/library/stdtypes.html#frozenset.intersection), [difference()](https://docs.python.org/3/library/stdtypes.html#frozenset.difference), and [symmetric\_difference()](https://docs.python.org/3/library/stdtypes.html" \l "frozenset.symmetric_difference" \o "frozenset.symmetric_difference), [issubset()](https://docs.python.org/3/library/stdtypes.html" \l "frozenset.issubset" \o "frozenset.issubset), and [issuperset()](https://docs.python.org/3/library/stdtypes.html" \l "frozenset.issuperset" \o "frozenset.issuperset) methods will accept any iterable as an argument. In contrast, their operator based counterparts require their arguments to be sets. This precludes error-prone constructions like set('abc') & 'cbs' in favor of the more readable set('abc').intersection('cbs').

Both [set](https://docs.python.org/3/library/stdtypes.html#set) and [frozenset](https://docs.python.org/3/library/stdtypes.html" \l "frozenset" \o "frozenset) support set to set comparisons. Two sets are equal if and only if every element of each set is contained in the other (each is a subset of the other). A set is less than another set if and only if the first set is a proper subset of the second set (is a subset, but is not equal). A set is greater than another set if and only if the first set is a proper superset of the second set (is a superset, but is not equal).

Instances of [set](https://docs.python.org/3/library/stdtypes.html#set) are compared to instances of [frozenset](https://docs.python.org/3/library/stdtypes.html" \l "frozenset" \o "frozenset) based on their members. For example, set('abc') == frozenset('abc') returns True and so does set('abc')in set([frozenset('abc')]).

The subset and equality comparisons do not generalize to a total ordering function. For example, any two nonempty disjoint sets are not equal and are not subsets of each other, so *all* of the following return False: a<b, a==b, or a>b.

Since sets only define partial ordering (subset relationships), the output of the [list.sort()](https://docs.python.org/3/library/stdtypes.html" \l "list.sort" \o "list.sort) method is undefined for lists of sets.

Set elements, like dictionary keys, must be [hashable](https://docs.python.org/3/glossary.html" \l "term-hashable).

Binary operations that mix [set](https://docs.python.org/3/library/stdtypes.html#set) instances with [frozenset](https://docs.python.org/3/library/stdtypes.html" \l "frozenset" \o "frozenset) return the type of the first operand. For example: frozenset('ab') | set('bc') returns an instance of [frozenset](https://docs.python.org/3/library/stdtypes.html" \l "frozenset" \o "frozenset).

The following table lists operations available for [set](https://docs.python.org/3/library/stdtypes.html#set) that do not apply to immutable instances of [frozenset](https://docs.python.org/3/library/stdtypes.html" \l "frozenset" \o "frozenset):

**update**(*\*others*)

**set |= other | ...**

Update the set, adding elements from all others.

**intersection\_update**(*\*others*)

**set &= other & ...**

Update the set, keeping only elements found in it and all others.

**difference\_update**(*\*others*)

**set -= other | ...**

Update the set, removing elements found in others.

**symmetric\_difference\_update**(*other*)

**set ^= other**

Update the set, keeping only elements found in either set, but not in both.

**add**(*elem*)

Add element *elem* to the set.

**remove**(*elem*)

Remove element *elem* from the set. Raises [KeyError](https://docs.python.org/3/library/exceptions.html" \l "KeyError" \o "KeyError) if *elem* is not contained in the set.

**discard**(*elem*)

Remove element *elem* from the set if it is present.

**pop**()

Remove and return an arbitrary element from the set. Raises [KeyError](https://docs.python.org/3/library/exceptions.html" \l "KeyError" \o "KeyError) if the set is empty.

**clear**()

Remove all elements from the set.

Note, the non-operator versions of the [update()](https://docs.python.org/3/library/stdtypes.html#frozenset.update), [intersection\_update()](https://docs.python.org/3/library/stdtypes.html" \l "frozenset.intersection_update" \o "frozenset.intersection_update), [difference\_update()](https://docs.python.org/3/library/stdtypes.html" \l "frozenset.difference_update" \o "frozenset.difference_update), and [symmetric\_difference\_update()](https://docs.python.org/3/library/stdtypes.html" \l "frozenset.symmetric_difference_update" \o "frozenset.symmetric_difference_update) methods will accept any iterable as an argument.

Note, the *elem* argument to the [\_\_contains\_\_()](https://docs.python.org/3/reference/datamodel.html#object.__contains__), [remove()](https://docs.python.org/3/library/stdtypes.html#frozenset.remove), and [discard()](https://docs.python.org/3/library/stdtypes.html#frozenset.discard) methods may be a set. To support searching for an equivalent frozenset, a temporary one is created from *elem*.