4.9. Set Types — `set`, `frozenset`

`len(s)`
Return the cardinality of set *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`(*other*, *...*)
`set | other | ...`
Return a new set with elements from the set and all others.

`intersection`(*other*, *...*)
`set & other & ...`
Return a new set with elements common to the set and all others.

`difference`(*other*, *...*)
`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*.

The following table lists operations available for `set` that do not apply to immutable instances of `frozenset`:

update(*other*, ...)
set |= other | ...
Update the set, adding elements from all others.

intersection_update(*other*, ...)
set &= other & ...
Update the set, keeping only elements found in it and all others.

difference_update(*other*, ...)
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` 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` if the set is empty.

clear()
Remove all elements from the set.

Note, the non-operator versions of the `update()`, `intersection_update()`, `difference_update()`, and `symmetric_difference_update()` methods will accept any iterable as an argument.

Note, the *elem* argument to the `__contains__()`, `remove()`, and `discard()` methods may be a set. To support searching for an equivalent frozenset, the *elem* set is temporarily mutated during the search and then restored. During the search, the *elem* set should not be read or mutated since it does not have a meaningful value.

4.10. Mapping Types — `dict`

A *mapping* object maps *hashable* values to arbitrary objects. Mappings are mutable