4.10. Mapping Types — dict

len(d)
Return the number of items in the dictionary *d*.

d[key]
Return the item of *d* with key *key*. Raises a KeyError if *key* is not in the map.

d[key] = value
Set d[key] to *value*.

del d[key]
Remove d[key] from *d*. Raises a KeyError if *key* is not in the map.

key in d
Return True if *d* has a key *key*, else False.

key not in d
Equivalent to not key in d.

iter(d)
Return an iterator over the keys of the dictionary. This is a shortcut for iter(d.keys()).

clear()
Remove all items from the dictionary.

copy()
Return a shallow copy of the dictionary.

classmethod **fromkeys**(*seq*[, *value*])
Create a new dictionary with keys from *seq* and values set to *value*.

fromkeys() is a class method that returns a new dictionary. *value* defaults to None.

get(*key*[, *default*])
Return the value for *key* if *key* is in the dictionary, else *default*. If *default* is not given, it defaults to None, so that this method never raises a KeyError.

items()
Return a new view of the dictionary's items ((key, value) pairs). See the *documentation of view objects*.

keys()
Return a new view of the dictionary's keys. See the *documentation of view objects*.

pop(*key*[, *default*])

If *key* is in the dictionary, remove it and return its value, else return *default*. If *default* is not given and *key* is not in the dictionary, a `KeyError` is raised.

popitem()
: Remove and return an arbitrary `(key, value)` pair from the dictionary.

  `popitem()` is useful to destructively iterate over a dictionary, as often used in set algorithms. If the dictionary is empty, calling `popitem()` raises a `KeyError`.

setdefault(*key*[, *default*])
: If *key* is in the dictionary, return its value. If not, insert *key* with a value of *default* and return *default*. *default* defaults to `None`.

update([*other*])
: Update the dictionary with the key/value pairs from *other*, overwriting existing keys. Return `None`.

values()
: Return a new view of the dictionary's values. See the *documentation of view objects*.

Dictionaries compare equal if and only if they have the same `(key, value)` pairs. Order comparisons ('<', '<=', '>=', '>') raise `TypeError`.

4.10.1. Dictionary view objects

The objects returned by `dict.keys()`, `dict.values()` and `dict.items()` are *view objects*. They provide a dynamic view on the dictionary's entries, which means that when the dictionary changes, the view reflects these changes.

len(dictview)
: Return the number of entries in the dictionary.

iter(dictview)
: Return an iterator over the keys, values or items (represented as tuples of `(key, value)`) in the dictionary.

x in dictview
: Return `True` if *x* is in the underlying dictionary's keys, values or items (in the latter case, *x* should be a `(key, value)` tuple).

Keys views are set-like since their entries are unique and hashable. If all values are hashable, so that `(key, value)` pairs are unique and hashable, then the items view is also set-like. (Values views are not treated as set-like since the entries are generally not unique.) For set-like views, all of the operations defined for the abstract base class `collections.abc.Set` are available (for example, ==, <, or ^).

An example of dictionary view usage: