

8.3. collections — Container datatypes

This module implements specialized container datatypes providing alternatives to Python’s general purpose built-in containers, `dict`, `list`, `set`, and `tuple`.

<code>namedtuple()</code>	factory function for creating tuple subclasses with named fields
<code>deque</code>	list-like container with fast appends and pops on either end
<code>ChainMap</code>	dict-like class for creating a single view of multiple mappings
<code>Counter</code>	dict subclass for counting hashable objects
<code>OrderedDict</code>	dict subclass that remembers the order entries were added
<code>defaultdict</code>	dict subclass that calls a factory function to supply missing values
<code>UserDict</code>	wrapper around dictionary objects for easier dict subclassing
<code>UserList</code>	wrapper around list objects for easier list subclassing
<code>UserString</code>	wrapper around string objects for easier string subclassing

Changed in version 3.3: Moved [Collections Abstract Base Classes](#) to the `collections.abc` module. For backwards compatibility, they continue to be visible in this module as well.

8.3.1. ChainMap objects

New in version 3.3.

A `ChainMap` class is provided for quickly linking a number of mappings so they can be treated as a single unit. It is often much faster than creating a new dictionary and running multiple `update()` calls.

The class can be used to simulate nested scopes and is useful in templating.

`class collections.ChainMap(*maps)`

A `ChainMap` groups multiple dicts or other mappings together to create a single, updateable view. If no *maps* are specified, a single empty dictionary is provided so that a new chain always has at least one mapping.

The underlying mappings are stored in a list. That list is public and can accessed or updated using the *maps* attribute. There is no other state.

Lookups search the underlying mappings successively until a key is found. In contrast, writes, updates, and deletions only operate on the first mapping.

A `ChainMap` incorporates the underlying mappings by reference. So, if one of the underlying mappings gets updated, those changes will be reflected in `ChainMap`.

All of the usual dictionary methods are supported. In addition, there is a *maps*