:mod:`collections` --- Container datatypes

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main) (Doc) (library)collections.rst, line 1); *backlink*

Unknown interpreted text role "mod".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main) (Doc) (library)collections.rst, line 4)

Unknown directive type "module".

```
.. module:: collections
   :synopsis: Container datatypes
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main) (Doc) (library)collections.rst, line 7)

Unknown directive type "moduleauthor".

```
.. moduleauthor:: Raymond Hettinger <python@rcn.com>
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main) (Doc) (library)collections.rst, line 8)

Unknown directive type "sectionauthor".

```
.. sectionauthor:: Raymond Hettinger <python@rcn.com>
```

Source code: :source:`Lib/collections/__init__.py`

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main) (Doc) (library)collections.rst, line 10); *backlink*

Unknown interpreted text role "source".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main) (Doc) (library)collections.rst, line 12)

Unknown directive type "testsetup".

```
.. testsetup:: *

   from collections import *
   import itertools
   __name__ = '<doctest>'
```

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

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main) (Doc) (library)collections.rst, line 20); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main) (Doc) (library)collections.rst, line 20); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main) (Doc) (library)collections.rst, line 20); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main) (Doc) (library)collections.rst, line 20); *backlink*

Unknown interpreted text role "class".

:func:`namedtuple` System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)collections.rst, line 26); *backlink* Unknown interpreted text role "func".	factory function for creating tuple subclasses with named fields
:class:`deque` System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)collections.rst, line 27); *backlink* Unknown interpreted text role "class".	list-like container with fast appends and pops on either end
:class:`ChainMap` System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)collections.rst, line 28); *backlink* Unknown interpreted text role "class".	dict-like class for creating a single view of multiple mappings
:class:`Counter` System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)collections.rst, line 29); *backlink* Unknown interpreted text role "class".	dict subclass for counting hashable objects

:class:`OrderedDict` System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)collections.rst, line 30); *backlink* Unknown interpreted text role "class".	dict subclass that remembers the order entries were added
:class:`defaultdict` System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)collections.rst, line 31); *backlink* Unknown interpreted text role "class".	dict subclass that calls a factory function to supply missing values
:class:`UserDict` System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)collections.rst, line 32); *backlink* Unknown interpreted text role "class".	wrapper around dictionary objects for easier dict subclassing
:class:`UserList` System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)collections.rst, line 33); *backlink* Unknown interpreted text role "class".	wrapper around list objects for easier list subclassing

:class:`UserString` System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)collections.rst, line 34); *backlink* Unknown interpreted text role "class".	wrapper around string objects for easier string subclassing

:class:`ChainMap` objects

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)collections.rst, line 37); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)collections.rst, line 40)

Unknown directive type "versionadded".

```
.. versionadded:: 3.3
```

A :class:`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 :meth:`~dict.update` calls.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)collections.rst, line 42); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)collections.rst, line 42); *backlink*

Unknown interpreted text role "meth".

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

A :class:`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.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)collections.rst, line 50); *backlink*

Unknown interpreted text role "class".

The underlying mappings are stored in a list. That list is public and can be 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 :class:`ChainMap` incorporates the underlying mappings by reference. So, if one of the underlying mappings gets updated, those changes will be reflected in :class:`ChainMap`.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)collections.rst, line 60); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)collections.rst, line 60); *backlink*

Unknown interpreted text role "class".

All of the usual dictionary methods are supported. In addition, there is a *maps* attribute, a method for creating new subcontexts, and a property for accessing all but the first mapping:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)collections.rst, line 68)

Unknown directive type "attribute".

```
.. attribute:: maps

   A user updateable list of mappings.  The list is ordered from
   first-searched to last-searched.  It is the only stored state and can
   be modified to change which mappings are searched.  The list should
   always contain at least one mapping.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)collections.rst, line 75)

Unknown directive type "method".

```
.. method:: new_child(m=None, **kwargs)

   Returns a new :class:`ChainMap` containing a new map followed by
   all of the maps in the current instance.  If ``m`` is specified,
   it becomes the new map at the front of the list of mappings; if not
   specified, an empty dict is used, so that a call to ``d.new_child()``
   is equivalent to: ``ChainMap({}, *d.maps)``. If any keyword arguments
   are specified, they update passed map or new empty dict. This method
   is used for creating subcontexts that can be updated without altering
   values in any of the parent mappings.

   .. versionchanged:: 3.4
      The optional ``m`` parameter was added.

   .. versionchanged:: 3.10
      Keyword arguments support was added.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)collections.rst, line 92)

Unknown directive type "attribute".

```
.. attribute:: parents

   Property returning a new :class:`ChainMap` containing all of the maps in
   the current instance except the first one.  This is useful for skipping
   the first map in the search.  Use cases are similar to those for the
   :keyword:`nonlocal` keyword used in :term:`nested scopes <nested
   scope>`.  The use cases also parallel those for the built-in
   :func:`super` function.  A reference to ``d.parents`` is equivalent to:
   ``ChainMap(*d.maps[1:])``.
```

Note, the iteration order of a :class:`ChainMap()` is determined by scanning the mappings last to first:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)collections.rst, line 102); *backlink*

Unknown interpreted text role "class".

```
>>> baseline = {'music': 'bach', 'art': 'rembrandt'}
>>> adjustments = {'art': 'van gogh', 'opera': 'carmen'}
>>> list(ChainMap(adjustments, baseline))
['music', 'art', 'opera']
```

This gives the same ordering as a series of :meth:`dict.update` calls starting with the last mapping:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)collections.rst, line 110); *backlink*

Unknown interpreted text role "meth".

```
>>> combined = baseline.copy()
>>> combined.update(adjustments)
>>> list(combined)
['music', 'art', 'opera']
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)collections.rst, line 118)

Unknown directive type "versionchanged".

```
.. versionchanged:: 3.9
   Added support for ``|`` and ``|=`` operators, specified in :pep:`584`.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)collections.rst, line 121)

Unknown directive type "seealso".

```
.. seealso::

    * The `MultiContext class
      <https://github.com/enthought/codetools/blob/4.0.0/codetools/contexts/multi_context.py>`_
      in the Enthought `CodeTools package
      <https://github.com/enthought/codetools>`_ has options to support
      writing to any mapping in the chain.

    * Django's `Context class
      <https://github.com/django/django/blob/main/django/template/context.py>`_
      for templating is a read-only chain of mappings.  It also features
      pushing and popping of contexts similar to the
      :meth:`~collections.ChainMap.new_child` method and the
      :attr:`~collections.ChainMap.parents` property.

    * The `Nested Contexts recipe
      <https://code.activestate.com/recipes/577434/>`_ has options to control
      whether writes and other mutations apply only to the first mapping or to
      any mapping in the chain.

    * A `greatly simplified read-only version of Chainmap
      <https://code.activestate.com/recipes/305268/>`_.
```

:class:`ChainMap` Examples and Recipes

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)collections.rst, line 145); *backlink*

Unknown interpreted text role "class".

This section shows various approaches to working with chained maps.

Example of simulating Python's internal lookup chain:

```
import builtins
pylookup = ChainMap(locals(), globals(), vars(builtins))
```

Example of letting user specified command-line arguments take precedence over environment variables which in turn take precedence over default values:

```
import os, argparse

defaults = {'color': 'red', 'user': 'guest'}

parser = argparse.ArgumentParser()
parser.add_argument('-u', '--user')
parser.add_argument('-c', '--color')
namespace = parser.parse_args()
command_line_args = {k: v for k, v in vars(namespace).items() if v is not None}

combined = ChainMap(command_line_args, os.environ, defaults)
print(combined['color'])
print(combined['user'])
```

Example patterns for using the :class:`ChainMap` class to simulate nested contexts:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)collections.rst, line 173); *backlink*

Unknown interpreted text role "class".

```
c = ChainMap()        # Create root context
d = c.new_child()     # Create nested child context
e = c.new_child()     # Child of c, independent from d
e.maps[0]             # Current context dictionary -- like Python's locals()
e.maps[-1]            # Root context -- like Python's globals()
e.parents             # Enclosing context chain -- like Python's nonlocals

d['x'] = 1            # Set value in current context
d['x']                # Get first key in the chain of contexts
del d['x']            # Delete from current context
list(d)               # All nested values
```

```
k in d                  # Check all nested values
len(d)                  # Number of nested values
d.items()               # All nested items
dict(d)                 # Flatten into a regular dictionary
```

The :class:`ChainMap` class only makes updates (writes and deletions) to the first mapping in the chain while lookups will search the full chain. However, if deep writes and deletions are desired, it is easy to make a subclass that updates keys found deeper in the chain:

> **System Message: ERROR/3** (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)collections.rst, line 192); *backlink*
>
> Unknown interpreted text role "class".

```
class DeepChainMap(ChainMap):
    'Variant of ChainMap that allows direct updates to inner scopes'

    def __setitem__(self, key, value):
        for mapping in self.maps:
            if key in mapping:
                mapping[key] = value
                return
        self.maps[0][key] = value

    def __delitem__(self, key):
        for mapping in self.maps:
            if key in mapping:
                del mapping[key]
                return
        raise KeyError(key)

>>> d = DeepChainMap({'zebra': 'black'}, {'elephant': 'blue'}, {'lion': 'yellow'})
>>> d['lion'] = 'orange'         # update an existing key two levels down
>>> d['snake'] = 'red'           # new keys get added to the topmost dict
>>> del d['elephant']            # remove an existing key one level down
>>> d                            # display result
DeepChainMap({'zebra': 'black', 'snake': 'red'}, {}, {'lion': 'orange'})
```

:class:`Counter` objects

> **System Message: ERROR/3** (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)collections.rst, line 222); *backlink*
>
> Unknown interpreted text role "class".

A counter tool is provided to support convenient and rapid tallies. For example:

```
>>> # Tally occurrences of words in a list
>>> cnt = Counter()
>>> for word in ['red', 'blue', 'red', 'green', 'blue', 'blue']:
...     cnt[word] += 1
>>> cnt
Counter({'blue': 3, 'red': 2, 'green': 1})

>>> # Find the ten most common words in Hamlet
>>> import re
>>> words = re.findall(r'\w+', open('hamlet.txt').read().lower())
>>> Counter(words).most_common(10)
[('the', 1143), ('and', 966), ('to', 762), ('of', 669), ('i', 631),
 ('you', 554),  ('a', 546), ('my', 514), ('hamlet', 471), ('in', 451)]
```

A :class:`Counter` is a :class:`dict` subclass for counting hashable objects. It is a collection where elements are stored as dictionary keys and their counts are stored as dictionary values. Counts are allowed to be any integer value including zero or negative counts. The :class:`Counter` class is similar to bags or multisets in other languages.

> **System Message: ERROR/3** (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)collections.rst, line 244); *backlink*
>
> Unknown interpreted text role "class".

> **System Message: ERROR/3** (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)collections.rst, line 244); *backlink*
>
> Unknown interpreted text role "class".

> **System Message: ERROR/3** (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)collections.rst, line 244); *backlink*
>
> Unknown interpreted text role "class".

Elements are counted from an *iterable* or initialized from another *mapping* (or counter):

```
>>> c = Counter()                           # a new, empty counter
>>> c = Counter('gallahad')                 # a new counter from an iterable
>>> c = Counter({'red': 4, 'blue': 2})      # a new counter from a mapping
>>> c = Counter(cats=4, dogs=8)             # a new counter from keyword args
```

Counter objects have a dictionary interface except that they return a zero count for missing items instead of raising a :exc:`KeyError`:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)collections.rst, line 258); *backlink*

Unknown interpreted text role "exc".

```
>>> c = Counter(['eggs', 'ham'])
>>> c['bacon']                              # count of a missing element is zero
0
```

Setting a count to zero does not remove an element from a counter. Use `del` to remove it entirely:

```
>>> c['sausage'] = 0                        # counter entry with a zero count
>>> del c['sausage']                        # del actually removes the entry
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)collections.rst, line 271)

Unknown directive type "versionadded".

```
.. versionadded:: 3.1
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)collections.rst, line 273)

Unknown directive type "versionchanged".

```
.. versionchanged:: 3.7 As a :class:`dict` subclass, :class:`Counter`
   inherited the capability to remember insertion order.  Math operations
   on *Counter* objects also preserve order.  Results are ordered
   according to when an element is first encountered in the left operand
   and then by the order encountered in the right operand.
```

Counter objects support additional methods beyond those available for all dictionaries:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)collections.rst, line 282)

Unknown directive type "method".

```
.. method:: elements()

    Return an iterator over elements repeating each as many times as its
    count.  Elements are returned in the order first encountered. If an
    element's count is less than one, :meth:`elements` will ignore it.

        >>> c = Counter(a=4, b=2, c=0, d=-2)
        >>> sorted(c.elements())
        ['a', 'a', 'a', 'a', 'b', 'b']
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)collections.rst, line 292)

Unknown directive type "method".

```
.. method:: most_common([n])

    Return a list of the *n* most common elements and their counts from the
    most common to the least.  If *n* is omitted or ``None``,
    :meth:`most_common` returns *all* elements in the counter.
    Elements with equal counts are ordered in the order first encountered:

        >>> Counter('abracadabra').most_common(3)
        [('a', 5), ('b', 2), ('r', 2)]
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)collections.rst, line 302)

Unknown directive type "method".

```
.. method:: subtract([iterable-or-mapping])

    Elements are subtracted from an *iterable* or from another *mapping*
    (or counter).  Like :meth:`dict.update` but subtracts counts instead
    of replacing them.  Both inputs and outputs may be zero or negative.

        >>> c = Counter(a=4, b=2, c=0, d=-2)
        >>> d = Counter(a=1, b=2, c=3, d=4)
        >>> c.subtract(d)
        >>> c
        Counter({'a': 3, 'b': 0, 'c': -3, 'd': -6})

    .. versionadded:: 3.2
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)collections.rst, line 316)

Unknown directive type "method".

```
.. method:: total()

    Compute the sum of the counts.

        >>> c = Counter(a=10, b=5, c=0)
        >>> c.total()
        15

    .. versionadded:: 3.10
```

The usual dictionary methods are available for :class:`Counter` objects except for two which work differently for counters.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)collections.rst, line 326); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)collections.rst, line 329)

Unknown directive type "method".

```
.. method:: fromkeys(iterable)

    This class method is not implemented for :class:`Counter` objects.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)collections.rst, line 333)

Unknown directive type "method".

```
.. method:: update([iterable-or-mapping])

    Elements are counted from an *iterable* or added-in from another
    *mapping* (or counter).  Like :meth:`dict.update` but adds counts
    instead of replacing them.  Also, the *iterable* is expected to be a
    sequence of elements, not a sequence of ``(key, value)`` pairs.
```

Counters support rich comparison operators for equality, subset, and superset relationships: ==, !=, <, <=, >, >=. All of those tests treat missing elements as having zero counts so that `Counter(a=1) == Counter(a=1, b=0)` returns true.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)collections.rst, line 345)

Unknown directive type "versionadded".

```
.. versionadded:: 3.10
   Rich comparison operations were added.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)collections.rst, line 348)

Unknown directive type "versionchanged".

```
.. versionchanged:: 3.10
   In equality tests, missing elements are treated as having zero counts.
   Formerly, ``Counter(a=3)`` and ``Counter(a=3, b=0)`` were considered
   distinct.
```

Common patterns for working with :class:`Counter` objects:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)collections.rst, line 353); *backlink*

Unknown interpreted text role "class".

```
c.total()                       # total of all counts
c.clear()                       # reset all counts
list(c)                         # list unique elements
set(c)                          # convert to a set
dict(c)                         # convert to a regular dictionary
c.items()                       # convert to a list of (elem, cnt) pairs
Counter(dict(list_of_pairs))    # convert from a list of (elem, cnt) pairs
c.most_common()[:-n-1:-1]       # n least common elements
+c                              # remove zero and negative counts
```

Several mathematical operations are provided for combining :class:`Counter` objects to produce multisets (counters that have counts greater than zero). Addition and subtraction combine counters by adding or subtracting the counts of corresponding elements. Intersection and union return the minimum and maximum of corresponding counts. Equality and inclusion compare corresponding counts. Each operation can accept inputs with signed counts, but the output will exclude results with counts of zero or less.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)collections.rst, line 365); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)collections.rst, line 373)

Unknown directive type "doctest".

```
.. doctest::

    >>> c = Counter(a=3, b=1)
    >>> d = Counter(a=1, b=2)
    >>> c + d                       # add two counters together:  c[x] + d[x]
    Counter({'a': 4, 'b': 3})
    >>> c - d                       # subtract (keeping only positive counts)
    Counter({'a': 2})
    >>> c & d                       # intersection:  min(c[x], d[x])
    Counter({'a': 1, 'b': 1})
    >>> c | d                       # union:  max(c[x], d[x])
    Counter({'a': 3, 'b': 2})
    >>> c == d                      # equality:  c[x] == d[x]
    False
    >>> c <= d                      # inclusion:  c[x] <= d[x]
    False
```

Unary addition and subtraction are shortcuts for adding an empty counter or subtracting from an empty counter.

```
>>> c = Counter(a=2, b=-4)
>>> +c
Counter({'a': 2})
>>> -c
Counter({'b': 4})
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)collections.rst, line 399)

Unknown directive type "versionadded".

```
.. versionadded:: 3.3
    Added support for unary plus, unary minus, and in-place multiset operations.
```

Note

Counters were primarily designed to work with positive integers to represent running counts; however, care was taken to not unnecessarily preclude use cases needing other types or negative values. To help with those use cases, this section documents the minimum range and type restrictions.

- The :class:`Counter` class itself is a dictionary subclass with no restrictions on its keys and values. The values are intended to be numbers representing counts, but you *could* store anything in the value field.

  System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)collections.rst, line 409); *backlink*

Unknown interpreted text role "class".

- The :meth:`~Counter.most_common` method requires only that the values be orderable.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)collections.rst, line 413); *backlink*

Unknown interpreted text role "meth".

- For in-place operations such as `c[key] += 1`, the value type need only support addition and subtraction. So fractions, floats, and decimals would work and negative values are supported. The same is also true for :meth:`~Counter.update` and :meth:`~Counter.subtract` which allow negative and zero values for both inputs and outputs.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)collections.rst, line 415); *backlink*

Unknown interpreted text role "meth".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)collections.rst, line 415); *backlink*

Unknown interpreted text role "meth".

- The multiset methods are designed only for use cases with positive values. The inputs may be negative or zero, but only outputs with positive values are created. There are no type restrictions, but the value type needs to support addition, subtraction, and comparison.
- The :meth:`~Counter.elements` method requires integer counts. It ignores zero and negative counts.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)collections.rst, line 426); *backlink*

Unknown interpreted text role "meth".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)collections.rst, line 429)

Unknown directive type "seealso".

```
.. seealso::

    * `Bag class <https://www.gnu.org/software/smalltalk/manual-base/html_node/Bag.html>`_
      in Smalltalk.

    * Wikipedia entry for `Multisets <https://en.wikipedia.org/wiki/Multiset>`_.

    * `C++ multisets <http://www.java2s.com/Tutorial/Cpp/0380__set-multiset/Catalog0380__set-multiset.h
      tutorial with examples.

    * For mathematical operations on multisets and their use cases, see
      *Knuth, Donald. The Art of Computer Programming Volume II,
      Section 4.6.3, Exercise 19*.

    * To enumerate all distinct multisets of a given size over a given set of
      elements, see :func:`itertools.combinations_with_replacement`::

          map(Counter, combinations_with_replacement('ABC', 2)) # --> AA AB AC BB BC CC
```

:class:`deque` objects

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)collections.rst, line 449); *backlink*

Unknown interpreted text role "class".

Returns a new deque object initialized left-to-right (using :meth:`append`) with data from *iterable*. If *iterable* is not specified, the new deque is empty.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)collections.rst, line 454); *backlink*

Unknown interpreted text role "meth".

Deques are a generalization of stacks and queues (the name is pronounced "deck" and is short for "double-ended queue"). Deques support thread-safe, memory efficient appends and pops from either side of the deque with approximately the same O(1) performance in either direction.

Though :class:`list` objects support similar operations, they are optimized for fast fixed-length operations and incur O(n) memory movement costs for `pop(0)` and `insert(0, v)` operations which change both the size and position of the underlying data representation.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)collections.rst, line 462); *backlink*

Unknown interpreted text role "class".

If *maxlen* is not specified or is `None`, deques may grow to an arbitrary length. Otherwise, the deque is bounded to the specified maximum length. Once a bounded length deque is full, when new items are added, a corresponding number of items are discarded from the opposite end. Bounded length deques provide functionality similar to the `tail` filter in Unix. They are also useful for tracking transactions and other pools of data where only the most recent activity is of interest.

Deque objects support the following methods:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)collections.rst, line 479)

Unknown directive type "method".

```
.. method:: append(x)

    Add *x* to the right side of the deque.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)collections.rst, line 484)

Unknown directive type "method".

```
.. method:: appendleft(x)

    Add *x* to the left side of the deque.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)collections.rst, line 489)

Unknown directive type "method".

```
.. method:: clear()

    Remove all elements from the deque leaving it with length 0.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)collections.rst, line 494)

Unknown directive type "method".

```
.. method:: copy()

    Create a shallow copy of the deque.

    .. versionadded:: 3.5
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)collections.rst, line 501)

Unknown directive type "method".

```
.. method:: count(x)

    Count the number of deque elements equal to *x*.

    .. versionadded:: 3.2
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)collections.rst, line 508)

Unknown directive type "method".

```
.. method:: extend(iterable)

    Extend the right side of the deque by appending elements from the iterable
    argument.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)collections.rst, line 514)

Unknown directive type "method".

```
.. method:: extendleft(iterable)

    Extend the left side of the deque by appending elements from *iterable*.
    Note, the series of left appends results in reversing the order of
    elements in the iterable argument.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)collections.rst, line 521)

Unknown directive type "method".

```
.. method:: index(x[, start[, stop]])

    Return the position of *x* in the deque (at or after index *start*
    and before index *stop*).  Returns the first match or raises
    :exc:`ValueError` if not found.

    .. versionadded:: 3.5
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)collections.rst, line 530)

Unknown directive type "method".

```
.. method:: insert(i, x)

    Insert *x* into the deque at position *i*.

    If the insertion would cause a bounded deque to grow beyond *maxlen*,
    an :exc:`IndexError` is raised.

    .. versionadded:: 3.5
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)collections.rst, line 540)

Unknown directive type "method".

```
.. method:: pop()

    Remove and return an element from the right side of the deque. If no
    elements are present, raises an :exc:`IndexError`.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)collections.rst, line 546)

Unknown directive type "method".

```
.. method:: popleft()

    Remove and return an element from the left side of the deque. If no
    elements are present, raises an :exc:`IndexError`.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)collections.rst, line 552)

Unknown directive type "method".

```
.. method:: remove(value)

    Remove the first occurrence of *value*.  If not found, raises a
    :exc:`ValueError`.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)collections.rst, line 558)

Unknown directive type "method".

```
.. method:: reverse()

    Reverse the elements of the deque in-place and then return ``None``.

    .. versionadded:: 3.2
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)collections.rst, line 565)

Unknown directive type "method".

```
.. method:: rotate(n=1)

    Rotate the deque *n* steps to the right.  If *n* is negative, rotate
    to the left.

    When the deque is not empty, rotating one step to the right is equivalent
    to ``d.appendleft(d.pop())``, and rotating one step to the left is
    equivalent to ``d.append(d.popleft())``.
```

Deque objects also provide one read-only attribute:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)collections.rst, line 577)

Unknown directive type "attribute".

```
.. attribute:: maxlen

    Maximum size of a deque or ``None`` if unbounded.

    .. versionadded:: 3.1
```

In addition to the above, deques support iteration, pickling, `len(d)`, `reversed(d)`, `copy.copy(d)`, `copy.deepcopy(d)`, membership testing with the :keyword:`in` operator, and subscript references such as `d[0]` to access the first element. Indexed access is O(1) at both ends but slows to O(n) in the middle. For fast random access, use lists instead.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)collections.rst, line 584); *backlink*

Unknown interpreted text role "keyword".

Starting in version 3.5, deques support `__add__()`, `__mul__()`, and `__imul__()`.

Example:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)collections.rst, line 595)

Unknown directive type "doctest".

```
.. doctest::

    >>> from collections import deque
    >>> d = deque('ghi')                 # make a new deque with three items
    >>> for elem in d:                   # iterate over the deque's elements
    ...     print(elem.upper())
    G
    H
    I
```

```
        >>> d.append('j')                     # add a new entry to the right side
        >>> d.appendleft('f')                 # add a new entry to the left side
        >>> d                                 # show the representation of the deque
        deque(['f', 'g', 'h', 'i', 'j'])

        >>> d.pop()                           # return and remove the rightmost item
        'j'
        >>> d.popleft()                       # return and remove the leftmost item
        'f'
        >>> list(d)                           # list the contents of the deque
        ['g', 'h', 'i']
        >>> d[0]                              # peek at leftmost item
        'g'
        >>> d[-1]                             # peek at rightmost item
        'i'

        >>> list(reversed(d))                 # list the contents of a deque in reverse
        ['i', 'h', 'g']
        >>> 'h' in d                          # search the deque
        True
        >>> d.extend('jkl')                   # add multiple elements at once
        >>> d
        deque(['g', 'h', 'i', 'j', 'k', 'l'])
        >>> d.rotate(1)                       # right rotation
        >>> d
        deque(['l', 'g', 'h', 'i', 'j', 'k'])
        >>> d.rotate(-1)                      # left rotation
        >>> d
        deque(['g', 'h', 'i', 'j', 'k', 'l'])

        >>> deque(reversed(d))                # make a new deque in reverse order
        deque(['l', 'k', 'j', 'i', 'h', 'g'])
        >>> d.clear()                         # empty the deque
        >>> d.pop()                           # cannot pop from an empty deque
        Traceback (most recent call last):
            File "<pyshell#6>", line 1, in -toplevel-
                d.pop()
        IndexError: pop from an empty deque

        >>> d.extendleft('abc')               # extendleft() reverses the input order
        >>> d
        deque(['c', 'b', 'a'])
```

:class:`deque` Recipes

> **System Message: ERROR/3** (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)collections.rst, line 649); *backlink*
>
> Unknown interpreted text role "class".

This section shows various approaches to working with deques.

Bounded length deques provide functionality similar to the `tail` filter in Unix:

```
def tail(filename, n=10):
    'Return the last n lines of a file'
    with open(filename) as f:
        return deque(f, n)
```

Another approach to using deques is to maintain a sequence of recently added elements by appending to the right and popping to the left:

```
def moving_average(iterable, n=3):
    # moving_average([40, 30, 50, 46, 39, 44]) --> 40.0 42.0 45.0 43.0
    # http://en.wikipedia.org/wiki/Moving_average
    it = iter(iterable)
    d = deque(itertools.islice(it, n-1))
    d.appendleft(0)
    s = sum(d)
    for elem in it:
        s += elem - d.popleft()
        d.append(elem)
        yield s / n
```

A round-robin scheduler can be implemented with input iterators stored in a :class:`deque`. Values are yielded from the active iterator in position zero. If that iterator is exhausted, it can be removed with :meth:`~deque.popleft`; otherwise, it can be cycled back to the end with the :meth:`~deque.rotate` method:

> **System Message: ERROR/3** (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)collections.rst, line 677); *backlink*
>
> Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)collections.rst, line 677); *backlink*

Unknown interpreted text role "meth".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)collections.rst, line 677); *backlink*

Unknown interpreted text role "meth".

```
def roundrobin(*iterables):
    "roundrobin('ABC', 'D', 'EF') --> A D E B F C"
    iterators = deque(map(iter, iterables))
    while iterators:
        try:
            while True:
                yield next(iterators[0])
                iterators.rotate(-1)
        except StopIteration:
            # Remove an exhausted iterator.
            iterators.popleft()
```

The :meth:`~deque.rotate` method provides a way to implement :class:`deque` slicing and deletion. For example, a pure Python implementation of `del d[n]` relies on the `rotate()` method to position elements to be popped:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)collections.rst, line 696); *backlink*

Unknown interpreted text role "meth".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)collections.rst, line 696); *backlink*

Unknown interpreted text role "class".

```
def delete_nth(d, n):
    d.rotate(-n)
    d.popleft()
    d.rotate(n)
```

To implement :class:`deque` slicing, use a similar approach applying :meth:`~deque.rotate` to bring a target element to the left side of the deque. Remove old entries with :meth:`~deque.popleft`, add new entries with :meth:`~deque.extend`, and then reverse the rotation. With minor variations on that approach, it is easy to implement Forth style stack manipulations such as `dup`, `drop`, `swap`, `over`, `pick`, `rot`, and `roll`.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)collections.rst, line 705); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)collections.rst, line 705); *backlink*

Unknown interpreted text role "meth".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)collections.rst, line 705); *backlink*

Unknown interpreted text role "meth".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)collections.rst, line 705); *backlink*

Unknown interpreted text role "meth".

:class:`defaultdict` objects

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)collections.rst, line 714); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)collections.rst, line 717)

Invalid class attribute value for "class" directive: "defaultdict(default_factory=None, /, [...])".

```
.. class:: defaultdict(default_factory=None, /, [...])

    Return a new dictionary-like object.  :class:`defaultdict` is a subclass of the
    built-in :class:`dict` class.  It overrides one method and adds one writable
    instance variable.  The remaining functionality is the same as for the
    :class:`dict` class and is not documented here.

    The first argument provides the initial value for the :attr:`default_factory`
    attribute; it defaults to ``None``. All remaining arguments are treated the same
    as if they were passed to the :class:`dict` constructor, including keyword
    arguments.


    :class:`defaultdict` objects support the following method in addition to the
    standard :class:`dict` operations:

    .. method:: __missing__(key)

        If the :attr:`default_factory` attribute is ``None``, this raises a
        :exc:`KeyError` exception with the *key* as argument.

        If :attr:`default_factory` is not ``None``, it is called without arguments
        to provide a default value for the given *key*, this value is inserted in
        the dictionary for the *key*, and returned.

        If calling :attr:`default_factory` raises an exception this exception is
        propagated unchanged.

        This method is called by the :meth:`__getitem__` method of the
        :class:`dict` class when the requested key is not found; whatever it
        returns or raises is then returned or raised by :meth:`__getitem__`.

        Note that :meth:`__missing__` is *not* called for any operations besides
        :meth:`__getitem__`. This means that :meth:`get` will, like normal
        dictionaries, return ``None`` as a default rather than using
        :attr:`default_factory`.


    :class:`defaultdict` objects support the following instance variable:


    .. attribute:: default_factory

        This attribute is used by the :meth:`__missing__` method; it is
        initialized from the first argument to the constructor, if present, or to
        ``None``, if absent.

    .. versionchanged:: 3.9
       Added merge (``|``) and update (``|=``) operators, specified in
       :pep:`584`.
```

:class:`defaultdict` Examples

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)collections.rst, line 769); *backlink*

Unknown interpreted text role "class".

Using :class:`list` as the :attr:`~defaultdict.default_factory`, it is easy to group a sequence of key-value pairs into a dictionary of lists:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)collections.rst, line 772); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)collections.rst, line 772); *backlink*

Unknown interpreted text role "attr".

```
>>> s = [('yellow', 1), ('blue', 2), ('yellow', 3), ('blue', 4), ('red', 1)]
>>> d = defaultdict(list)
>>> for k, v in s:
...     d[k].append(v)
...
>>> sorted(d.items())
[('blue', [2, 4]), ('red', [1]), ('yellow', [1, 3])]
```

When each key is encountered for the first time, it is not already in the mapping; so an entry is automatically created using the

:attr:`~defaultdict.default_factory` function which returns an empty :class:`list`. The :meth:`list.append` operation then attaches the value to the new list. When keys are encountered again, the look-up proceeds normally (returning the list for that key) and the :meth:`list.append` operation adds another value to the list. This technique is simpler and faster than an equivalent technique using :meth:`dict.setdefault`:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)collections.rst, line 783); *backlink*

Unknown interpreted text role "attr".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)collections.rst, line 783); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)collections.rst, line 783); *backlink*

Unknown interpreted text role "meth".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)collections.rst, line 783); *backlink*

Unknown interpreted text role "meth".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)collections.rst, line 783); *backlink*

Unknown interpreted text role "meth".

```
>>> d = {}
>>> for k, v in s:
...     d.setdefault(k, []).append(v)
...
>>> sorted(d.items())
[('blue', [2, 4]), ('red', [1]), ('yellow', [1, 3])]
```

Setting the :attr:`~defaultdict.default_factory` to :class:`int` makes the :class:`defaultdict` useful for counting (like a bag or multiset in other languages):

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)collections.rst, line 798); *backlink*

Unknown interpreted text role "attr".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)collections.rst, line 798); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)collections.rst, line 798); *backlink*

Unknown interpreted text role "class".

```
>>> s = 'mississippi'
>>> d = defaultdict(int)
>>> for k in s:
...     d[k] += 1
...
>>> sorted(d.items())
[('i', 4), ('m', 1), ('p', 2), ('s', 4)]
```

When a letter is first encountered, it is missing from the mapping, so the :attr:`~defaultdict.default_factory` function calls :func:`int` to supply a default count of zero. The increment operation then builds up the count for each letter.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)collections.rst, line 810); *backlink*

Unknown interpreted text role "attr".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)collections.rst, line 810); *backlink*

Unknown interpreted text role "func".

The function :func:`int` which always returns zero is just a special case of constant functions. A faster and more flexible way to create constant functions is to use a lambda function which can supply any constant value (not just zero):

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)collections.rst, line 814); *backlink*

Unknown interpreted text role "func".

```
>>> def constant_factory(value):
...     return lambda: value
>>> d = defaultdict(constant_factory('<missing>'))
>>> d.update(name='John', action='ran')
>>> '%(name)s %(action)s to %(object)s' % d
'John ran to <missing>'
```

Setting the :attr:`~defaultdict.default_factory` to :class:`set` makes the :class:`defaultdict` useful for building a dictionary of sets:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)collections.rst, line 826); *backlink*

Unknown interpreted text role "attr".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)collections.rst, line 826); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)collections.rst, line 826); *backlink*

Unknown interpreted text role "class".

```
>>> s = [('red', 1), ('blue', 2), ('red', 3), ('blue', 4), ('red', 1), ('blue', 4)]
>>> d = defaultdict(set)
>>> for k, v in s:
...     d[k].add(v)
...
>>> sorted(d.items())
[('blue', {2, 4}), ('red', {1, 3})]
```

:func:`namedtuple` Factory Function for Tuples with Named Fields

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)collections.rst, line 838); *backlink*

Unknown interpreted text role "func".

Named tuples assign meaning to each position in a tuple and allow for more readable, self-documenting code. They can be used wherever regular tuples are used, and they add the ability to access fields by name instead of position index.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)collections.rst, line 845)

Unknown directive type "function".

```
.. function:: namedtuple(typename, field_names, *, rename=False, defaults=None, module=None)

   Returns a new tuple subclass named *typename*.  The new subclass is used to
   create tuple-like objects that have fields accessible by attribute lookup as
   well as being indexable and iterable.  Instances of the subclass also have a
   helpful docstring (with typename and field_names) and a helpful :meth:`__repr__`
   method which lists the tuple contents in a ``name=value`` format.

   The *field_names* are a sequence of strings such as ``['x', 'y']``.
   Alternatively, *field_names* can be a single string with each fieldname
   separated by whitespace and/or commas, for example ``'x y'`` or ``'x, y'``.

   Any valid Python identifier may be used for a fieldname except for names
   starting with an underscore.  Valid identifiers consist of letters, digits,
   and underscores but do not start with a digit or underscore and cannot be
   a :mod:`keyword` such as *class*, *for*, *return*, *global*, *pass*,
   or *raise*.

   If *rename* is true, invalid fieldnames are automatically replaced
   with positional names.  For example, ``['abc', 'def', 'ghi', 'abc']`` is
   converted to ``['abc', '_1', 'ghi', '_3']``, eliminating the keyword
```

```
        ``def`` and the duplicate fieldname ``abc``.

        *defaults* can be ``None`` or an :term:`iterable` of default values.
        Since fields with a default value must come after any fields without a
        default, the *defaults* are applied to the rightmost parameters.  For
        example, if the fieldnames are ``['x', 'y', 'z']`` and the defaults are
        ``(1, 2)``, then ``x`` will be a required argument, ``y`` will default to
        ``1``, and ``z`` will default to ``2``.

        If *module* is defined, the ``__module__`` attribute of the named tuple is
        set to that value.

        Named tuple instances do not have per-instance dictionaries, so they are
        lightweight and require no more memory than regular tuples.

        To support pickling, the named tuple class should be assigned to a variable
        that matches *typename*.

        .. versionchanged:: 3.1
           Added support for *rename*.

        .. versionchanged:: 3.6
           The *verbose* and *rename* parameters became
           :ref:`keyword-only arguments <keyword-only_parameter>`.

        .. versionchanged:: 3.6
           Added the *module* parameter.

        .. versionchanged:: 3.7
           Removed the *verbose* parameter and the :attr:`_source` attribute.

        .. versionchanged:: 3.7
           Added the *defaults* parameter and the :attr:`_field_defaults`
           attribute.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)collections.rst, line 901)

Unknown directive type "doctest".

```
.. doctest::
    :options: +NORMALIZE_WHITESPACE

    >>> # Basic example
    >>> Point = namedtuple('Point', ['x', 'y'])
    >>> p = Point(11, y=22)     # instantiate with positional or keyword arguments
    >>> p[0] + p[1]             # indexable like the plain tuple (11, 22)
    33
    >>> x, y = p                # unpack like a regular tuple
    >>> x, y
    (11, 22)
    >>> p.x + p.y               # fields also accessible by name
    33
    >>> p                       # readable __repr__ with a name=value style
    Point(x=11, y=22)
```

Named tuples are especially useful for assigning field names to result tuples returned by the :mod:`csv` or :mod:`sqlite3` modules:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)collections.rst, line 917); *backlink*

Unknown interpreted text role "mod".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)collections.rst, line 917); *backlink*

Unknown interpreted text role "mod".

```
EmployeeRecord = namedtuple('EmployeeRecord', 'name, age, title, department, paygrade')

import csv
for emp in map(EmployeeRecord._make, csv.reader(open("employees.csv", "rb"))):
    print(emp.name, emp.title)

import sqlite3
conn = sqlite3.connect('/companydata')
cursor = conn.cursor()
cursor.execute('SELECT name, age, title, department, paygrade FROM employees')
for emp in map(EmployeeRecord._make, cursor.fetchall()):
    print(emp.name, emp.title)
```

In addition to the methods inherited from tuples, named tuples support three additional methods and two attributes. To prevent conflicts with field names, the method and attribute names start with an underscore.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)collections.rst, line 937)

Unknown directive type "classmethod".

```
.. classmethod:: somenamedtuple._make(iterable)

    Class method that makes a new instance from an existing sequence or iterable.

    .. doctest::

        >>> t = [11, 22]
        >>> Point._make(t)
        Point(x=11, y=22)
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)collections.rst, line 947)

Unknown directive type "method".

```
.. method:: somenamedtuple._asdict()

    Return a new :class:`dict` which maps field names to their corresponding
    values:

    .. doctest::

        >>> p = Point(x=11, y=22)
        >>> p._asdict()
        {'x': 11, 'y': 22}

    .. versionchanged:: 3.1
        Returns an :class:`OrderedDict` instead of a regular :class:`dict`.

    .. versionchanged:: 3.8
        Returns a regular :class:`dict` instead of an :class:`OrderedDict`.
        As of Python 3.7, regular dicts are guaranteed to be ordered.  If the
        extra features of :class:`OrderedDict` are required, the suggested
        remediation is to cast the result to the desired type:
        ``OrderedDict(nt._asdict())``.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)collections.rst, line 968)

Unknown directive type "method".

```
.. method:: somenamedtuple._replace(**kwargs)

    Return a new instance of the named tuple replacing specified fields with new
    values::

        >>> p = Point(x=11, y=22)
        >>> p._replace(x=33)
        Point(x=33, y=22)

        >>> for partnum, record in inventory.items():
        ...     inventory[partnum] = record._replace(price=newprices[partnum], timestamp=time.now())
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)collections.rst, line 980)

Unknown directive type "attribute".

```
.. attribute:: somenamedtuple._fields

    Tuple of strings listing the field names.  Useful for introspection
    and for creating new named tuple types from existing named tuples.

    .. doctest::

        >>> p._fields            # view the field names
        ('x', 'y')

        >>> Color = namedtuple('Color', 'red green blue')
        >>> Pixel = namedtuple('Pixel', Point._fields + Color._fields)
        >>> Pixel(11, 22, 128, 255, 0)
        Pixel(x=11, y=22, red=128, green=255, blue=0)
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)collections.rst, line 995)

Unknown directive type "attribute".

```
.. attribute:: somenamedtuple._field_defaults

   Dictionary mapping field names to default values.

   .. doctest::

        >>> Account = namedtuple('Account', ['type', 'balance'], defaults=[0])
        >>> Account._field_defaults
        {'balance': 0}
        >>> Account('premium')
        Account(type='premium', balance=0)
```

To retrieve a field whose name is stored in a string, use the :func:`getattr` function:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)collections.rst, line 1007); *backlink*

Unknown interpreted text role "func".

```
>>> getattr(p, 'x')
11
```

To convert a dictionary to a named tuple, use the double-star-operator (as described in :ref:`tut-unpacking-arguments`):

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)collections.rst, line 1013); *backlink*

Unknown interpreted text role "ref".

```
>>> d = {'x': 11, 'y': 22}
>>> Point(**d)
Point(x=11, y=22)
```

Since a named tuple is a regular Python class, it is easy to add or change functionality with a subclass. Here is how to add a calculated field and a fixed-width print format:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)collections.rst, line 1024)

Unknown directive type "doctest".

```
.. doctest::

    >>> class Point(namedtuple('Point', ['x', 'y'])):
    ...     __slots__ = ()
    ...     @property
    ...     def hypot(self):
    ...         return (self.x ** 2 + self.y ** 2) ** 0.5
    ...     def __str__(self):
    ...         return 'Point: x=%6.3f  y=%6.3f  hypot=%6.3f' % (self.x, self.y, self.hypot)

    >>> for p in Point(3, 4), Point(14, 5/7):
    ...     print(p)
    Point: x= 3.000  y= 4.000  hypot= 5.000
    Point: x=14.000  y= 0.714  hypot=14.018
```

The subclass shown above sets `__slots__` to an empty tuple. This helps keep memory requirements low by preventing the creation of instance dictionaries.

Subclassing is not useful for adding new, stored fields. Instead, simply create a new named tuple type from the :attr:`~somenamedtuple._fields` attribute:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)collections.rst, line 1042); *backlink*

Unknown interpreted text role "attr".

```
>>> Point3D = namedtuple('Point3D', Point._fields + ('z',))
```

Docstrings can be customized by making direct assignments to the `__doc__` fields:

```
>>> Book = namedtuple('Book', ['id', 'title', 'authors'])
>>> Book.__doc__ += ': Hardcover book in active collection'
>>> Book.id.__doc__ = '13-digit ISBN'
>>> Book.title.__doc__ = 'Title of first printing'
>>> Book.authors.__doc__ = 'List of authors sorted by last name'
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)collections.rst, line 1056)

Unknown directive type "versionchanged".

```
.. versionchanged:: 3.5
   Property docstrings became writeable.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)collections.rst, line 1059)

Unknown directive type "seealso".

```
.. seealso::

    * See :class:`typing.NamedTuple` for a way to add type hints for named
      tuples.  It also provides an elegant notation using the :keyword:`class`
      keyword::

          class Component(NamedTuple):
              part_number: int
              weight: float
              description: Optional[str] = None

    * See :meth:`types.SimpleNamespace` for a mutable namespace based on an
      underlying dictionary instead of a tuple.

    * The :mod:`dataclasses` module provides a decorator and functions for
      automatically adding generated special methods to user-defined classes.
```

:class:`OrderedDict` objects

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)collections.rst, line 1077); *backlink*

Unknown interpreted text role "class".

Ordered dictionaries are just like regular dictionaries but have some extra capabilities relating to ordering operations. They have become less important now that the built-in :class:`dict` class gained the ability to remember insertion order (this new behavior became guaranteed in Python 3.7).

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)collections.rst, line 1080); *backlink*

Unknown interpreted text role "class".

Some differences from :class:`dict` still remain:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)collections.rst, line 1086); *backlink*

Unknown interpreted text role "class".

- The regular :class:`dict` was designed to be very good at mapping operations. Tracking insertion order was secondary.

  System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)collections.rst, line 1088); *backlink*

  Unknown interpreted text role "class".

- The :class:`OrderedDict` was designed to be good at reordering operations. Space efficiency, iteration speed, and the performance of update operations were secondary.

  System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)collections.rst, line 1091); *backlink*

  Unknown interpreted text role "class".

- The :class:`OrderedDict` algorithm can handle frequent reordering operations better than :class:`dict`. As shown in the recipes below, this makes it suitable for implementing various kinds of LRU caches.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)collections.rst, line 1095); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)collections.rst, line 1095); *backlink*

Unknown interpreted text role "class".

- The equality operation for :class:`OrderedDict` checks for matching order.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)collections.rst, line 1099); *backlink*

Unknown interpreted text role "class".

A regular :class:`dict` can emulate the order sensitive equality test with `p == q and all(k1 == k2 for k1, k2 in zip(p, q))`.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)collections.rst, line 1101); *backlink*

Unknown interpreted text role "class".

- The :meth:`popitem` method of :class:`OrderedDict` has a different signature. It accepts an optional argument to specify which item is popped.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)collections.rst, line 1104); *backlink*

Unknown interpreted text role "meth".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)collections.rst, line 1104); *backlink*

Unknown interpreted text role "class".

A regular :class:`dict` can emulate OrderedDict's `od.popitem(last=True)` with `d.popitem()` which is guaranteed to pop the rightmost (last) item.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)collections.rst, line 1107); *backlink*

Unknown interpreted text role "class".

A regular :class:`dict` can emulate OrderedDict's `od.popitem(last=False)` with `(k := next(iter(d)), d.pop(k))` which will return and remove the leftmost (first) item if it exists.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)collections.rst, line 1110); *backlink*

Unknown interpreted text role "class".

- :class:`OrderedDict` has a :meth:`move_to_end` method to efficiently reposition an element to an endpoint.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)collections.rst, line 1114); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)collections.rst, line 1114); *backlink*

Unknown interpreted text role "meth".

A regular :class:`dict` can emulate OrderedDict's `od.move_to_end(k, last=True)` with `d[k] = d.pop(k)` which will move the key and its associated value to the rightmost (last) position.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)collections.rst, line 1117); *backlink*

Unknown interpreted text role "class".

A regular :class:`dict` does not have an efficient equivalent for OrderedDict's `od.move_to_end(k, last=False)` which moves the key and its associated value to the leftmost (first) position.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)collections.rst, line 1121); *backlink*

Unknown interpreted text role "class".

- Until Python 3.8, :class:`dict` lacked a :meth:`__reversed__` method.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)collections.rst, line 1125); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)collections.rst, line 1125); *backlink*

Unknown interpreted text role "meth".

Return an instance of a :class:`dict` subclass that has methods specialized for rearranging dictionary order.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)collections.rst, line 1130); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)collections.rst, line 1133)

Unknown directive type "versionadded".

```
.. versionadded:: 3.1
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)collections.rst, line 1135)

Unknown directive type "method".

```
.. method:: popitem(last=True)

    The :meth:`popitem` method for ordered dictionaries returns and removes a
    (key, value) pair.  The pairs are returned in
    :abbr:`LIFO (last-in, first-out)` order if *last* is true
    or :abbr:`FIFO (first-in, first-out)` order if false.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)collections.rst, line 1142)

Unknown directive type "method".

```
.. method:: move_to_end(key, last=True)

    Move an existing *key* to either end of an ordered dictionary.  The item
    is moved to the right end if *last* is true (the default) or to the
    beginning if *last* is false.  Raises :exc:`KeyError` if the *key* does
    not exist:
```

```
.. doctest::

    >>> d = OrderedDict.fromkeys('abcde')
    >>> d.move_to_end('b')
    >>> ''.join(d)
    'acdeb'
    >>> d.move_to_end('b', last=False)
    >>> ''.join(d)
    'bacde'

.. versionadded:: 3.2
```

In addition to the usual mapping methods, ordered dictionaries also support reverse iteration using :func:`reversed`.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main) (Doc) (library)collections.rst`, **line 1161**); ***backlink***

Unknown interpreted text role "func".

Equality tests between :class:`OrderedDict` objects are order-sensitive and are implemented as `list(od1.items())==list(od2.items())`. Equality tests between :class:`OrderedDict` objects and other :class:`~collections.abc.Mapping` objects are order-insensitive like regular dictionaries. This allows :class:`OrderedDict` objects to be substituted anywhere a regular dictionary is used.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main) (Doc) (library)collections.rst`, **line 1164**); ***backlink***

Unknown interpreted text role "class".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main) (Doc) (library)collections.rst`, **line 1164**); ***backlink***

Unknown interpreted text role "class".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main) (Doc) (library)collections.rst`, **line 1164**); ***backlink***

Unknown interpreted text role "class".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main) (Doc) (library)collections.rst`, **line 1164**); ***backlink***

Unknown interpreted text role "class".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main) (Doc) (library)collections.rst`, **line 1171**)

Unknown directive type "versionchanged".

```
.. versionchanged:: 3.5
   The items, keys, and values :term:`views <dictionary view>`
   of :class:`OrderedDict` now support reverse iteration using :func:`reversed`.
```

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main) (Doc) (library)collections.rst`, **line 1175**)

Unknown directive type "versionchanged".

```
.. versionchanged:: 3.6
   With the acceptance of :pep:`468`, order is retained for keyword arguments
   passed to the :class:`OrderedDict` constructor and its :meth:`update`
   method.
```

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main) (Doc) (library)collections.rst`, **line 1180**)

Unknown directive type "versionchanged".

```
.. versionchanged:: 3.9
    Added merge (``|``) and update (``|=``) operators, specified in :pep:`584`.
```

:class:`OrderedDict` Examples and Recipes

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)collections.rst, line 1184); *backlink*

Unknown interpreted text role "class".

It is straightforward to create an ordered dictionary variant that remembers the order the keys were *last* inserted. If a new entry overwrites an existing entry, the original insertion position is changed and moved to the end:

```
class LastUpdatedOrderedDict(OrderedDict):
    'Store items in the order the keys were last added'

    def __setitem__(self, key, value):
        super().__setitem__(key, value)
        self.move_to_end(key)
```

An :class:`OrderedDict` would also be useful for implementing variants of :func:`functools.lru_cache`:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)collections.rst, line 1199); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)collections.rst, line 1199); *backlink*

Unknown interpreted text role "func".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)collections.rst, line 1202)

Unknown directive type "testcode".

```
.. testcode::

    from time import time

    class TimeBoundedLRU:
        "LRU Cache that invalidates and refreshes old entries."

        def __init__(self, func, maxsize=128, maxage=30):
            self.cache = OrderedDict()      # { args : (timestamp, result)}
            self.func = func
            self.maxsize = maxsize
            self.maxage = maxage

        def __call__(self, *args):
            if args in self.cache:
                self.cache.move_to_end(args)
                timestamp, result = self.cache[args]
                if time() - timestamp <= self.maxage:
                    return result
            result = self.func(*args)
            self.cache[args] = time(), result
            if len(self.cache) > self.maxsize:
                self.cache.popitem(0)
            return result
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)collections.rst, line 1228)

Unknown directive type "testcode".

```
.. testcode::

    class MultiHitLRUCache:
        """ LRU cache that defers caching a result until
            it has been requested multiple times.

            To avoid flushing the LRU cache with one-time requests,
            we don't cache until a request has been made more than once.

        """

        def __init__(self, func, maxsize=128, maxrequests=4096, cache_after=1):
            self.requests = OrderedDict()   # { uncached_key : request_count }
            self.cache = OrderedDict()      # { cached_key : function_result }
            self.func = func
            self.maxrequests = maxrequests  # max number of uncached requests
            self.maxsize = maxsize          # max number of stored return values
            self.cache_after = cache_after
```

```
            def __call__(self, *args):
                if args in self.cache:
                    self.cache.move_to_end(args)
                    return self.cache[args]
                result = self.func(*args)
                self.requests[args] = self.requests.get(args, 0) + 1
                if self.requests[args] <= self.cache_after:
                    self.requests.move_to_end(args)
                    if len(self.requests) > self.maxrequests:
                        self.requests.popitem(0)
                else:
                    self.requests.pop(args, None)
                    self.cache[args] = result
                    if len(self.cache) > self.maxsize:
                        self.cache.popitem(0)
                return result
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)collections.rst, line 1264)

Unknown directive type "doctest".

```
.. doctest::
    :hide:

    >>> def square(x):
    ...     return x * x
    ...
    >>> f = MultiHitLRUCache(square, maxsize=4, maxrequests=6)
    >>> list(map(f, range(10)))  # First requests, don't cache
    [0, 1, 4, 9, 16, 25, 36, 49, 64, 81]
    >>> f(4)  # Cache the second request
    16
    >>> f(6)  # Cache the second request
    36
    >>> f(2)  # The first request aged out, so don't cache
    4
    >>> f(6)  # Cache hit
    36
    >>> f(4)  # Cache hit and move to front
    16
    >>> list(f.cache.values())
    [36, 16]
    >>> set(f.requests).isdisjoint(f.cache)
    True
    >>> list(map(f, [9, 8, 7]))   # Cache these second requests
    [81, 64, 49]
    >>> list(map(f, [7, 9]))  # Cache hits
    [49, 81]
    >>> list(f.cache.values())
    [16, 64, 49, 81]
    >>> set(f.requests).isdisjoint(f.cache)
    True
```

:class:`UserDict` objects

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)collections.rst, line 1296); *backlink*

Unknown interpreted text role "class".

The class, :class:`UserDict` acts as a wrapper around dictionary objects. The need for this class has been partially supplanted by the ability to subclass directly from :class:`dict`; however, this class can be easier to work with because the underlying dictionary is accessible as an attribute.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)collections.rst, line 1299); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)collections.rst, line 1299); *backlink*

Unknown interpreted text role "class".

Class that simulates a dictionary. The instance's contents are kept in a regular dictionary, which is accessible via the :attr:`data` attribute of :class:`UserDict` instances. If *initialdata* is provided, :attr:`data` is initialized with its contents; note that a reference to *initialdata* will not be kept, allowing it to be used for other purposes.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)collections.rst, line 1307); *backlink*

Unknown interpreted text role "attr".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)collections.rst, line 1307); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)collections.rst, line 1307); *backlink*

Unknown interpreted text role "attr".

In addition to supporting the methods and operations of mappings, :class:`UserDict` instances provide the following attribute:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)collections.rst, line 1313); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)collections.rst, line 1316)

Unknown directive type "attribute".

```
.. attribute:: data

   A real dictionary used to store the contents of the :class:`UserDict`
   class.
```

:class:`UserList` objects

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)collections.rst, line 1323); *backlink*

Unknown interpreted text role "class".

This class acts as a wrapper around list objects. It is a useful base class for your own list-like classes which can inherit from them and override existing methods or add new ones. In this way, one can add new behaviors to lists.

The need for this class has been partially supplanted by the ability to subclass directly from :class:`list`; however, this class can be easier to work with because the underlying list is accessible as an attribute.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)collections.rst, line 1331); *backlink*

Unknown interpreted text role "class".

Class that simulates a list. The instance's contents are kept in a regular list, which is accessible via the :attr:`data` attribute of :class:`UserList` instances. The instance's contents are initially set to a copy of *list*, defaulting to the empty list `[]`. *list* can be any iterable, for example a real Python list or a :class:`UserList` object.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)collections.rst, line 1337); *backlink*

Unknown interpreted text role "attr".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)collections.rst, line 1337); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)collections.rst, line 1337); *backlink*

Unknown interpreted text role "class".

In addition to supporting the methods and operations of mutable sequences, :class:`UserList` instances provide the following attribute:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)collections.rst, line 1343); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)collections.rst, line 1346)

Unknown directive type "attribute".

```
.. attribute:: data

    A real :class:`list` object used to store the contents of the
    :class:`UserList` class.
```

Subclassing requirements: Subclasses of :class:`UserList` are expected to offer a constructor which can be called with either no arguments or one argument. List operations which return a new sequence attempt to create an instance of the actual implementation class. To do so, it assumes that the constructor can be called with a single parameter, which is a sequence object used as a data source.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)collections.rst, line 1351); *backlink*

Unknown interpreted text role "class".

If a derived class does not wish to comply with this requirement, all of the special methods supported by this class will need to be overridden; please consult the sources for information about the methods which need to be provided in that case.

:class:`UserString` objects

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)collections.rst, line 1363); *backlink*

Unknown interpreted text role "class".

The class, :class:`UserString` acts as a wrapper around string objects. The need for this class has been partially supplanted by the ability to subclass directly from :class:`str`; however, this class can be easier to work with because the underlying string is accessible as an attribute.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)collections.rst, line 1366); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)collections.rst, line 1366); *backlink*

Unknown interpreted text role "class".

Class that simulates a string object. The instance's content is kept in a regular string object, which is accessible via the :attr:`data` attribute of :class:`UserString` instances. The instance's contents are initially set to a copy of *seq*. The *seq* argument can be any object which can be converted into a string using the built-in :func:`str` function.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)collections.rst, line 1374); *backlink*

Unknown interpreted text role "attr".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)collections.rst, line 1374); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)collections.rst, line 1374); *backlink*

Unknown interpreted text role "func".

In addition to supporting the methods and operations of strings, :class:`UserString` instances provide the following attribute:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)collections.rst, line 1381); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)collections.rst, line 1384)

Unknown directive type "attribute".

```
.. attribute:: data

    A real :class:`str` object used to store the contents of the
    :class:`UserString` class.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)collections.rst, line 1389)

Unknown directive type "versionchanged".

```
.. versionchanged:: 3.5
   New methods ``__getnewargs__``, ``__rmod__``, ``casefold``,
   ``format_map``, ``isprintable``, and ``maketrans``.
```