8.3. [collections](https://docs.python.org/zh-cn/3.6/library/collections.html#module-collections) — 其他容器数据类型

**Source code:** [Lib/collections/\_\_init\_\_.py](https://github.com/python/cpython/tree/3.6/Lib/collections/__init__.py)

这个模块实现了特定目标的容器**，以提供Python标准内建容器**[**dict**](https://docs.python.org/zh-cn/3.6/library/stdtypes.html#dict)**,**[**list**](https://docs.python.org/zh-cn/3.6/library/stdtypes.html#list)**,**[**set**](https://docs.python.org/zh-cn/3.6/library/stdtypes.html#set)**, 和**[**tuple**](https://docs.python.org/zh-cn/3.6/library/stdtypes.html#tuple)**的替代选择。**

|  |  |
| --- | --- |
| [namedtuple()](https://docs.python.org/zh-cn/3.6/library/collections.html#collections.namedtuple) | 创建命名元组子类的工厂函数 |
| [deque](https://docs.python.org/zh-cn/3.6/library/collections.html#collections.deque) | 类似列表(list)的容器，实现了在两端快速添加(append)和弹出(pop) |
| [ChainMap](https://docs.python.org/zh-cn/3.6/library/collections.html#collections.ChainMap) | 类似字典(dict)的容器类，将多个映射集合到一个视图里面 |
| [Counter](https://docs.python.org/zh-cn/3.6/library/collections.html#collections.Counter) | 字典的子类，提供了可哈希对象的计数功能 |
| [OrderedDict](https://docs.python.org/zh-cn/3.6/library/collections.html#collections.OrderedDict) | 字典的子类，保存了他们被添加的顺序 |
| [defaultdict](https://docs.python.org/zh-cn/3.6/library/collections.html#collections.defaultdict) | 字典的子类，提供了一个工厂函数，为字典查询提供一个默认值 |
| [UserDict](https://docs.python.org/zh-cn/3.6/library/collections.html#collections.UserDict) | 封装了字典对象，简化了字典子类化 |
| [UserList](https://docs.python.org/zh-cn/3.6/library/collections.html#collections.UserList) | 封装了列表对象，简化了列表子类化 |
| [UserString](https://docs.python.org/zh-cn/3.6/library/collections.html#collections.UserString) | 封装了字符串对象，简化了字符串子类化 |

*在 3.3 版更改:*Moved [容器抽象基类](https://docs.python.org/zh-cn/3.6/library/collections.abc.html#collections-abstract-base-classes) to the [collections.abc](https://docs.python.org/zh-cn/3.6/library/collections.abc.html#module-collections.abc) module. For backwards compatibility, they continue to be visible in this module as well.

**其中比较重要的是Counter, deque, OrderedDict, namedtuple**

8.3.1. [ChainMap](https://docs.python.org/zh-cn/3.6/library/collections.html#collections.ChainMap) 对象

*3.3 新版功能.*

一个 [ChainMap](https://docs.python.org/zh-cn/3.6/library/collections.html#collections.ChainMap) 类是为了将多个映射快速的链接到一起，这样它们就可以作为一个单元处理。它通常比创建一个新字典和多次调用 [update()](https://docs.python.org/zh-cn/3.6/library/stdtypes.html#dict.update) 要快很多。

这个类可以用于模拟嵌套作用域，并且在模版化的时候比较有用。

*class*collections.**ChainMap**(*\*maps*)

一个 [ChainMap](https://docs.python.org/zh-cn/3.6/library/collections.html#collections.ChainMap) 将多个字典或者其他映射组合在一起，创建一个单独的可更新的视图。 如果没有 *maps* 被指定，就提供一个默认的空字典，这样一个新链至少有一个映射。

底层映射被存储在一个列表中。这个列表是公开的，可以通过 *maps* 属性存取和更新。没有其他的状态。

搜索查询底层映射，直到一个键被找到。不同的是，写，更新和删除只操作第一个映射。

一个 [ChainMap](https://docs.python.org/zh-cn/3.6/library/collections.html#collections.ChainMap) 通过引用合并底层映射。 所以，如果一个底层映射更新了，这些更改会反映到 [ChainMap](https://docs.python.org/zh-cn/3.6/library/collections.html#collections.ChainMap) 。

支持所有常用字典方法。另外还有一个 *maps* 属性(attribute)，一个创建子上下文的方法(method)， 一个存取它们首个映射的属性(property):

**maps**

一个可以更新的映射列表。这个列表是按照第一次搜索到最后一次搜索的顺序组织的。它是仅有的存储状态，可以被修改。列表最少包含一个映射。

**new\_child**(*m=None*)

返回一个新的 [ChainMap](https://docs.python.org/zh-cn/3.6/library/collections.html#collections.ChainMap) 类，包含了一个新映射(map)，后面跟随当前实例的全部映射(map)。如果 m 被指定，它就成为不同新的实例，就是在所有映射前加上 m，如果没有指定，就加上一个空字典，这样的话一个 d.new\_child() 调用等价于 ChainMap({}, \*d.maps) 。这个方法用于创建子上下文，不改变任何父映射的值。

*在 3.4 版更改:*添加了 m 可选参数。

**parents**

属性返回一个新的 [ChainMap](https://docs.python.org/zh-cn/3.6/library/collections.html#collections.ChainMap) 包含所有的当前实例的映射，除了第一个。这样可以在搜索的时候跳过第一个映射。 使用的场景类似在 [nested scopes](https://docs.python.org/zh-cn/3.6/glossary.html#term-nested-scope) 嵌套作用域中使用 [nonlocal](https://docs.python.org/zh-cn/3.6/reference/simple_stmts.html#nonlocal) 关键词。用例也可以类比内建函数 [super()](https://docs.python.org/zh-cn/3.6/library/functions.html#super) 。一个 d.parents 的引用等价于 ChainMap(\*d.maps[1:]) 。

**参见**

* [MultiContext class](https://github.com/enthought/codetools/blob/4.0.0/codetools/contexts/multi_context.py) 在 Enthought [CodeTools package](https://github.com/enthought/codetools) 有支持写映射的选项。
* Django’s [Context class](https://github.com/django/django/blob/master/django/template/context.py) for templating is a read-only chain of mappings. It also features pushing and popping of contexts similar to the [new\_child()](https://docs.python.org/zh-cn/3.6/library/collections.html#collections.ChainMap.new_child) method and the [parents()](https://docs.python.org/zh-cn/3.6/library/collections.html#collections.ChainMap.parents) property.
* [Nested Contexts recipe](https://code.activestate.com/recipes/577434/) 提供了是否对第一个映射或其他映射进行写和其他修改的选项。
* 一个 [极简的只读版 Chainmap](https://code.activestate.com/recipes/305268/).

8.3.1.1. [ChainMap](https://docs.python.org/zh-cn/3.6/library/collections.html#collections.ChainMap) 例子和方法

这一节提供了多个使用链映射的案例。

模拟Python内部lookup链的例子

**import** **builtins**

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

让用户指定的命令行参数优先于环境变量，优先于默认值的例子

**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}

combined = ChainMap(command\_line\_args, os.environ, defaults)

print(combined['color'])

print(combined['user'])

用 [ChainMap](https://docs.python.org/zh-cn/3.6/library/collections.html#collections.ChainMap) 类模拟嵌套上下文的例子

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'] *# Get first key in the chain of contexts*

d['x'] = 1 *# Set value in current context*

**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*

[ChainMap](https://docs.python.org/zh-cn/3.6/library/collections.html#collections.ChainMap) 类只更新链中的第一个映射，但lookup会搜索整个链。 然而，如果需要深度写和删除，也可以很容易的通过定义一个子类来实现它

**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'})

8.3.2. [Counter](https://docs.python.org/zh-cn/3.6/library/collections.html#collections.Counter) 对象

一个计数器工具提供快速和方便的计数。比如

>>>

**>>>** *# 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)]

*class*collections.**Counter**([*iterable-or-mapping*])

A [Counter](https://docs.python.org/zh-cn/3.6/library/collections.html#collections.Counter) is a [dict](https://docs.python.org/zh-cn/3.6/library/stdtypes.html#dict) subclass for counting hashable objects. It is an unordered 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 [Counter](https://docs.python.org/zh-cn/3.6/library/collections.html#collections.Counter) class is similar to bags or multisets in other languages.

元素从一个 *iterable* 被计数或从其他的 *mapping* (or counter)初始化：

**>>>** 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对象有一个字典接口，如果引用的键没有任何记录，就返回一个0，而不是弹出一个 [KeyError](https://docs.python.org/zh-cn/3.6/library/exceptions.html#KeyError) :

**>>>** c = Counter(['eggs', 'ham'])

**>>>** c['bacon'] *# count of a missing element is zero*

0

设置一个计数为0不会从计数器中移去一个元素。使用 del 来删除它:

**>>>** c['sausage'] = 0 *# counter entry with a zero count*

**>>> del** c['sausage'] *# del actually removes the entry*

*3.1 新版功能.*

计数器对象除了字典方法以外，还提供了三个其他的方法：

**elements**()

返回一个迭代器，每个元素重复计数的个数。元素顺序是任意的。如果一个元素的计数小于1， [elements()](https://docs.python.org/zh-cn/3.6/library/collections.html#collections.Counter.elements) 就会忽略它。

**>>>** c = Counter(a=4, b=2, c=0, d=-2)

**>>>** sorted(c.elements())

['a', 'a', 'a', 'a', 'b', 'b']

**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, [most\_common()](https://docs.python.org/zh-cn/3.6/library/collections.html#collections.Counter.most_common) returns *all* elements in the counter. Elements with equal counts are ordered arbitrarily:

**>>>** Counter('abracadabra').most\_common(3)

[('a', 5), ('r', 2), ('b', 2)]

**subtract**([*iterable-or-mapping*])

从 *迭代对象* 或 *映射对象* 减去元素。像 [dict.update()](https://docs.python.org/zh-cn/3.6/library/stdtypes.html#dict.update) 但是是减去，而不是替换。输入和输出都可以是0或者负数。

**>>>** 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})

*3.2 新版功能.*

通常字典方法都可用于 [Counter](https://docs.python.org/zh-cn/3.6/library/collections.html#collections.Counter) 对象，除了有两个方法工作方式与字典并不相同。

**fromkeys**(*iterable*)

这个类方法没有在 [Counter](https://docs.python.org/zh-cn/3.6/library/collections.html#collections.Counter) 中实现。

**update**([*iterable-or-mapping*])

从 *迭代对象* 计数元素或者 从另一个 *映射对象* (或计数器) 添加。 像 [dict.update()](https://docs.python.org/zh-cn/3.6/library/stdtypes.html#dict.update) 但是是加上，而不是替换。另外，*迭代对象* 应该是序列元素，而不是一个 (key, value) 对。

[Counter](https://docs.python.org/zh-cn/3.6/library/collections.html#collections.Counter) 对象的常用案例

sum(c.values()) *# 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*

提供了几个数学操作，可以结合 [Counter](https://docs.python.org/zh-cn/3.6/library/collections.html#collections.Counter) 对象，以生产 multisets (计数器中大于0的元素）。 加和减，结合计数器，通过加上或者减去元素的相应计数。交集和并集返回相应计数的最小或最大值。每种操作都可以接受带符号的计数，但是输出会忽略掉结果为零或者小于零的计数。

**>>>** 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})

单目加和减（一元操作符）意思是从空计数器加或者减去。

**>>>** c = Counter(a=2, b=-4)

**>>>** +c

Counter({'a': 2})

**>>>** -c

Counter({'b': 4})

*3.3 新版功能:*添加了对一元加，一元减和位置集合操作的支持。

**注解**

计数器主要是为了表达运行的正的计数而设计；但是，小心不要预先排除负数或者其他类型。为了帮助这些用例，这一节记录了最小范围和类型限制。

* [Counter](https://docs.python.org/zh-cn/3.6/library/collections.html#collections.Counter) 类是一个字典的子类，不限制键和值。值用于表示计数，但你实际上 *可以* 存储任何其他值。
* The most\_common() method requires only that the values be orderable.
* 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 update() and subtract() which allow negative and zero values for both inputs and outputs.
* Multiset多集合方法只为正值的使用情况设计。输入可以是负数或者0，但只输出计数为正的值。没有类型限制，但值类型需要支持加，减和比较操作。
* The elements() method requires integer counts. It ignores zero and negative counts.

**参见**

* [Bag class](https://www.gnu.org/software/smalltalk/manual-base/html_node/Bag.html) 在 Smalltalk。
* Wikipedia 链接 [Multisets](https://en.wikipedia.org/wiki/Multiset).
* [C++ multisets](http://www.java2s.com/Tutorial/Cpp/0380__set-multiset/Catalog0380__set-multiset.htm) 教程和例子。
* 数学操作和多集合用例，参考 *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 [itertools.combinations\_with\_replacement()](https://docs.python.org/zh-cn/3.6/library/itertools.html#itertools.combinations_with_replacement):

map(Counter, combinations\_with\_replacement(‘ABC’, 2)) –> AA AB AC BB BC CC

8.3.3. [deque](https://docs.python.org/zh-cn/3.6/library/collections.html#collections.deque) 对象

*class*collections.**deque**([*iterable*[, *maxlen*]])

返回一个新的双向队列对象，从左到右初始化(用方法 [append()](https://docs.python.org/zh-cn/3.6/library/collections.html#collections.deque.append)) ，从 *iterable* （迭代对象) 数据创建。如果 *iterable* 没有指定，新队列为空。

Deque队列是由栈或者queue队列生成的（发音是 “deck”，”double-ended queue”的简称）。Deque 支持线程安全，内存高效添加(append)和弹出(pop)，从两端都可以，两个方向的大概开销都是 O(1) 复杂度。

虽然 [list](https://docs.python.org/zh-cn/3.6/library/stdtypes.html#list) 对象也支持类似操作，不过这里优化了定长操作和 pop(0) 和 insert(0, v) 的开销。它们引起 O(n) 内存移动的操作，改变底层数据表达的大小和位置。

如果 *maxlen* 没有指定或者是 None ，deques 可以增长到任意长度。否则，deque就限定到指定最大长度。一旦限定长度的deque满了，当新项加入时，同样数量的项就从另一端弹出。限定长度deque提供类似Unix filter tail 的功能。它们同样可以用与追踪最近的交换和其他数据池活动。

双向队列(deque)对象支持以下方法：

**append**(*x*)

添加 *x* 到右端。

**appendleft**(*x*)

添加 *x* 到左端。

**clear**()

移除所有元素，使其长度为0.

**copy**()

创建一份浅拷贝。

*3.5 新版功能.*

**count**(*x*)

计算 deque 中元素等于 *x* 的个数。

*3.2 新版功能.*

**extend**(*iterable*)

扩展deque的右侧，通过添加iterable参数中的元素。

**extendleft**(*iterable*)

扩展deque的左侧，通过添加iterable参数中的元素。注意，左添加时，在结果中iterable参数中的顺序将被反过来添加。

**index**(*x*[, *start*[, *stop*]])

返回 *x* 在 deque 中的位置（在索引 *start* 之后，索引 *stop* 之前）。 返回第一个匹配项，如果未找到则引发 [ValueError](https://docs.python.org/zh-cn/3.6/library/exceptions.html#ValueError)。

*3.5 新版功能.*

**insert**(*i*, *x*)

在位置 *i* 插入 *x* 。

如果插入会导致一个限长 deque 超出长度 *maxlen* 的话，就引发一个 [IndexError](https://docs.python.org/zh-cn/3.6/library/exceptions.html#IndexError)。

*3.5 新版功能.*

**pop**()

移去并且返回一个元素，deque 最右侧的那一个。 如果没有元素的话，就引发一个 [IndexError](https://docs.python.org/zh-cn/3.6/library/exceptions.html#IndexError)。

**popleft**()

移去并且返回一个元素，deque 最左侧的那一个。 如果没有元素的话，就引发 [IndexError](https://docs.python.org/zh-cn/3.6/library/exceptions.html#IndexError)。

**remove**(*value*)

移除找到的第一个 *value*。 如果没有的话就引发 [ValueError](https://docs.python.org/zh-cn/3.6/library/exceptions.html#ValueError)。

**reverse**()

将deque逆序排列。返回 None 。

*3.2 新版功能.*

**rotate**(*n=1*)

向右循环移动 *n* 步。 如果 *n* 是负数，就向左循环。

如果deque不是空的，向右循环移动一步就等价于 d.appendleft(d.pop()) ， 向左循环一步就等价于 d.append(d.popleft()) 。

Deque对象同样提供了一个只读属性:

**maxlen**

Deque的最大尺寸，如果没有限定的话就是 None 。

*3.1 新版功能.*

除了以上，deque还支持迭代，清洗，len(d), reversed(d), copy.copy(d), copy.deepcopy(d), 成员测试 [in](https://docs.python.org/zh-cn/3.6/reference/expressions.html#in) 操作符，和下标引用 d[-1] 。索引存取在两端的复杂度是 O(1)， 在中间的复杂度比 O(n) 略低。要快速存取，使用list来替代。

Deque从版本3.5开始支持 \_\_add\_\_(), \_\_mul\_\_(), 和 \_\_imul\_\_() 。

示例:

**>>> 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'])

8.3.3.1. [deque](https://docs.python.org/zh-cn/3.6/library/collections.html#collections.deque) 用法

这一节展示了deque的多种用法。

限长deque提供了类似Unix tail 过滤功能

**def** tail(filename, n=10):

'Return the last n lines of a file'

**with** open(filename) **as** f:

**return** deque(f, n)

另一个用法是维护一个近期添加元素的序列，通过从右边添加和从左边弹出

**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

The rotate() method provides a way to implement [deque](https://docs.python.org/zh-cn/3.6/library/collections.html#collections.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:

**def** delete\_nth(d, n):

d.rotate(-n)

d.popleft()

d.rotate(n)

To implement [deque](https://docs.python.org/zh-cn/3.6/library/collections.html#collections.deque) slicing, use a similar approach applying rotate() to bring a target element to the left side of the deque. Remove old entries with popleft(), add new entries with 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.

8.3.4. [defaultdict](https://docs.python.org/zh-cn/3.6/library/collections.html#collections.defaultdict) 对象

*class*collections.**defaultdict**([*default\_factory*[, *...*]])

返回一个新的类似字典的对象。 [defaultdict](https://docs.python.org/zh-cn/3.6/library/collections.html#collections.defaultdict) 是内置 [dict](https://docs.python.org/zh-cn/3.6/library/stdtypes.html#dict) 类的子类。它重载了一个方法并添加了一个可写的实例变量。其余的功能与 [dict](https://docs.python.org/zh-cn/3.6/library/stdtypes.html#dict) 类相同，此处不再重复说明。

第一个参数 [default\_factory](https://docs.python.org/zh-cn/3.6/library/collections.html#collections.defaultdict.default_factory) 提供了一个初始值。它默认为 None 。所有的其他参数都等同与 [dict](https://docs.python.org/zh-cn/3.6/library/stdtypes.html#dict) 构建器中的参数对待，包括关键词参数。

[defaultdict](https://docs.python.org/zh-cn/3.6/library/collections.html#collections.defaultdict) 对象除了支持 [dict](https://docs.python.org/zh-cn/3.6/library/stdtypes.html#dict) 的操作，还支持下面的方法作为扩展:

**\_\_missing\_\_**(*key*)

如果 [default\_factory](https://docs.python.org/zh-cn/3.6/library/collections.html#collections.defaultdict.default_factory) 属性为 None，则调用本方法会抛出 [KeyError](https://docs.python.org/zh-cn/3.6/library/exceptions.html#KeyError) 异常，附带参数 *key*。

如果 [default\_factory](https://docs.python.org/zh-cn/3.6/library/collections.html#collections.defaultdict.default_factory) 不为 None ， 它就会会被调用，不带参数，为 *key* 提供一个默认值， 这个值和 *key* 作为一个对被插入到字典中，并返回。

如果调用 [default\_factory](https://docs.python.org/zh-cn/3.6/library/collections.html#collections.defaultdict.default_factory) 时抛出了异常，这个异常会原封不动地向外层传递。

在无法找到所需键值时，本方法会被 [dict](https://docs.python.org/zh-cn/3.6/library/stdtypes.html#dict) 中的 [\_\_getitem\_\_()](https://docs.python.org/zh-cn/3.6/reference/datamodel.html#object.__getitem__) 方法调用。无论本方法返回了值还是抛出了异常，都会被 [\_\_getitem\_\_()](https://docs.python.org/zh-cn/3.6/reference/datamodel.html#object.__getitem__) 传递。

注意 [\_\_missing\_\_()](https://docs.python.org/zh-cn/3.6/library/collections.html#collections.defaultdict.__missing__) *不会* 被 [\_\_getitem\_\_()](https://docs.python.org/zh-cn/3.6/reference/datamodel.html#object.__getitem__) 以外的其他方法调用。意思就是 get() 会向正常的dict那样返回 None ，而不是使用 [default\_factory](https://docs.python.org/zh-cn/3.6/library/collections.html#collections.defaultdict.default_factory) 。

[defaultdict](https://docs.python.org/zh-cn/3.6/library/collections.html#collections.defaultdict) 支持以下实例变量:

**default\_factory**

这个属性被 [\_\_missing\_\_()](https://docs.python.org/zh-cn/3.6/library/collections.html#collections.defaultdict.__missing__) 方法使用；它从构建器的第一个参数初始化，如果提供了的话，否则就是 None 。

8.3.4.1. [defaultdict](https://docs.python.org/zh-cn/3.6/library/collections.html#collections.defaultdict) 例子

Using [list](https://docs.python.org/zh-cn/3.6/library/stdtypes.html#list) as the default\_factory, it is easy to group a sequence of key-value pairs into a dictionary of lists:

**>>>** 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 default\_factory function which returns an empty [list](https://docs.python.org/zh-cn/3.6/library/stdtypes.html#list). The 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 list.append() operation adds another value to the list. This technique is simpler and faster than an equivalent technique using [dict.setdefault()](https://docs.python.org/zh-cn/3.6/library/stdtypes.html#dict.setdefault):

**>>>** d = {}

**>>> for** k, v **in** s:

**...**  d.setdefault(k, []).append(v)

**...**

**>>>** sorted(d.items())

[('blue', [2, 4]), ('red', [1]), ('yellow', [1, 3])]

Setting the default\_factory to [int](https://docs.python.org/zh-cn/3.6/library/functions.html#int) makes the [defaultdict](https://docs.python.org/zh-cn/3.6/library/collections.html#collections.defaultdict) useful for counting (like a bag or multiset in other languages):

**>>>** 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 default\_factory function calls [int()](https://docs.python.org/zh-cn/3.6/library/functions.html#int) to supply a default count of zero. The increment operation then builds up the count for each letter.

函数 [int()](https://docs.python.org/zh-cn/3.6/library/functions.html#int) 总是返回0，是常数函数的特殊情况。一个更快和灵活的方法是使用lambda函数，可以提供任何常量值(不只是0):

**>>> 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 default\_factory to [set](https://docs.python.org/zh-cn/3.6/library/stdtypes.html#set) makes the [defaultdict](https://docs.python.org/zh-cn/3.6/library/collections.html#collections.defaultdict) useful for building a dictionary of sets:

**>>>** 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})]

8.3.5. [namedtuple()](https://docs.python.org/zh-cn/3.6/library/collections.html#collections.namedtuple) 命名元组的工厂函数

命名元组赋予每个位置一个含义，提供可读性和自文档性。它们可以用于任何普通元组，并添加了通过名字获取值的能力，通过索引值也是可以的。

collections.**namedtuple**(*typename*, *field\_names*, *\**, *verbose=False*, *rename=False*, *module=None*)

返回一个新的元组子类，名为 *typename* 。这个新的子类用于创建类元组的对象，可以通过域名来获取属性值，同样也可以通过索引和迭代获取值。子类实例同样有文档字符串（类名和域名）另外一个有用的 [\_\_repr\_\_()](https://docs.python.org/zh-cn/3.6/reference/datamodel.html#object.__repr__) 方法，以 name=value 格式列明了元组内容。

*field\_names* 是一个像 [‘x’, ‘y’] 一样的字符串序列。另外 *field\_names* 可以是一个纯字符串，用空白或逗号分隔开元素名，比如 'x y' 或者 'x, y' 。

任何有效的Python 标识符都可以作为域名，除了下划线开头的那些。有效标识符由字母，数字，下划线组成，但首字母不能是数字或下划线，另外不能是关键词 [keyword](https://docs.python.org/zh-cn/3.6/library/keyword.html#module-keyword) 比如 *class*, *for*, *return*, *global*, *pass*, 或 *raise* 。

如果 *rename* 为真， 无效域名会自动转换成位置名。比如 ['abc', 'def', 'ghi', 'abc'] 转换成 ['abc', '\_1', 'ghi', '\_3'] ， 消除关键词 def 和重复域名 abc 。

If *verbose* is true, the class definition is printed after it is built. This option is outdated; instead, it is simpler to print the \_source attribute.

如果 *module* 值有定义，命名元组的 \_\_module\_\_ 属性值就被设置。

命名元组实例没有字典，所以它们要更轻量，并且占用更小内存。

*在 3.1 版更改:*添加了对 *rename* 的支持。

*在 3.6 版更改: verbose* 和 *rename* 参数成为 [仅限关键字参数](https://docs.python.org/zh-cn/3.6/glossary.html#keyword-only-parameter).

*在 3.6 版更改:*添加了 *module* 参数。

**>>>** *# 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)

命名元组尤其有用于赋值 [csv](https://docs.python.org/zh-cn/3.6/library/csv.html#module-csv) [sqlite3](https://docs.python.org/zh-cn/3.6/library/sqlite3.html#module-sqlite3) 模块返回的元组

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

**import** **csv**

**for** emp **in** map(EmployeeRecord.\_make, csv.reader(open("employees.csv", "r"))): #注意,这里写’rb’是会报错的

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)

除了继承元组的方法，命名元组还支持三个额外的方法和两个属性。为了防止域名冲突，方法和属性以下划线开始。

*classmethod*somenamedtuple.**\_make**(*iterable*)

类方法从存在的序列或迭代实例创建一个新实例。

**>>>** t = [11, 22]

**>>>** Point.\_make(t)

Point(x=11, y=22)

somenamedtuple.**\_asdict**()

Return a new [OrderedDict](https://docs.python.org/zh-cn/3.6/library/collections.html#collections.OrderedDict) which maps field names to their corresponding values:

**>>>** p = Point(x=11, y=22)

**>>>** p.\_asdict()

OrderedDict([('x', 11), ('y', 22)])

*在 3.1 版更改:*返回一个 [OrderedDict](https://docs.python.org/zh-cn/3.6/library/collections.html#collections.OrderedDict) 而不是 [dict](https://docs.python.org/zh-cn/3.6/library/stdtypes.html#dict) 。

somenamedtuple.**\_replace**(*\*\*kwargs*)

返回一个新的命名元组实例，并将指定域替换为新的值

>>>

**>>>** 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())

somenamedtuple.**\_source**

A string with the pure Python source code used to create the named tuple class. The source makes the named tuple self-documenting. It can be printed, executed using [exec()](https://docs.python.org/zh-cn/3.6/library/functions.html#exec), or saved to a file and imported.

*3.3 新版功能.*

somenamedtuple.**\_fields**

字符串元组列出了域名。用于提醒和从现有元组创建一个新的命名元组类型。

**>>>** 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)

要获取这个名字域的值，使用 [getattr()](https://docs.python.org/zh-cn/3.6/library/functions.html#getattr) 函数 :

**>>>** getattr(p, 'x')

11

To convert a dictionary to a named tuple, use the double-star-operator (as described in [解包参数列表](https://docs.python.org/zh-cn/3.6/tutorial/controlflow.html#tut-unpacking-arguments)):

**>>>** d = {'x': 11, 'y': 22}

**>>>** Point(\*\*d)

Point(x=11, y=22)

因为一个命名元组是一个正常的Python类，它可以很容易的通过子类更改功能。这里是如何添加一个计算域和定宽输出打印格式:

**>>> 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

上面的子类设置 \_\_slots\_\_ 为一个空元组。通过阻止创建实例字典保持了较低的内存开销。

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

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

文档字符串可以自定义，通过直接赋值给 \_\_doc\_\_ 属性:

**>>>** 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'

*在 3.5 版更改:*文档字符串属性变成可写。

Default values can be implemented by using \_replace() to customize a prototype instance:

**>>>** Account = namedtuple('Account', 'owner balance transaction\_count')

**>>>** default\_account = Account('<owner name>', 0.0, 0)

**>>>** johns\_account = default\_account.\_replace(owner='John')

**>>>** janes\_account = default\_account.\_replace(owner='Jane')

**参见**

* [Recipe for named tuple abstract base class with a metaclass mix-in](https://code.activestate.com/recipes/577629-namedtupleabc-abstract-base-class-mix-in-for-named/) by Jan Kaliszewski. Besides providing an [abstract base class](https://docs.python.org/zh-cn/3.6/glossary.html#term-abstract-base-class) for named tuples, it also supports an alternate [metaclass](https://docs.python.org/zh-cn/3.6/glossary.html#term-metaclass)-based constructor that is convenient for use cases where named tuples are being subclassed.
* 对于以字典为底层的可变域名， 参考 [types.SimpleNamespace()](https://docs.python.org/zh-cn/3.6/library/types.html#types.SimpleNamespace) 。
* See [typing.NamedTuple()](https://docs.python.org/zh-cn/3.6/library/typing.html#typing.NamedTuple) for a way to add type hints for named tuples.

8.3.6. [OrderedDict](https://docs.python.org/zh-cn/3.6/library/collections.html#collections.OrderedDict) 对象

Ordered dictionaries are just like regular dictionaries but they remember the order that items were inserted. When iterating over an ordered dictionary, the items are returned in the order their keys were first added.

*class*collections.**OrderedDict**([*items*])

Return an instance of a dict subclass, supporting the usual [dict](https://docs.python.org/zh-cn/3.6/library/stdtypes.html#dict) methods. An *OrderedDict* is a dict that remembers the order that keys were first inserted. If a new entry overwrites an existing entry, the original insertion position is left unchanged. Deleting an entry and reinserting it will move it to the end.

*3.1 新版功能.*

**popitem**(*last=True*)

有序字典的 [popitem()](https://docs.python.org/zh-cn/3.6/library/collections.html#collections.OrderedDict.popitem) 方法移除并返回一个 (key, value) 键值对。 如果 *last* 值为真，则按 LIFO 后进先出的顺序返回键值对，否则就按 FIFO 先进先出的顺序返回键值对。

**move\_to\_end**(*key*, *last=True*)

将现有 *key* 移动到有序字典的任一端。 如果 *last* 为真值（默认）则将元素移至末尾；如果 *last* 为假值则将元素移至开头。如果 *key* 不存在则会触发 [KeyError](https://docs.python.org/zh-cn/3.6/library/exceptions.html#KeyError):

>>>

**>>>** d = OrderedDict.fromkeys('abcde')

**>>>** d.move\_to\_end('b')

**>>>** ''.join(d.keys())

'acdeb'

**>>>** d.move\_to\_end('b', last=**False**)

**>>>** ''.join(d.keys())

'bacde'

*3.2 新版功能.*

相对于通常的映射方法，有序字典还另外提供了逆序迭代的支持，通过 [reversed()](https://docs.python.org/zh-cn/3.6/library/functions.html#reversed) 。

[OrderedDict](https://docs.python.org/zh-cn/3.6/library/collections.html#collections.OrderedDict) 之间的相等测试是顺序敏感的，实现为 list(od1.items())==list(od2.items()) 。 [OrderedDict](https://docs.python.org/zh-cn/3.6/library/collections.html#collections.OrderedDict) 对象和其他的 [Mapping](https://docs.python.org/zh-cn/3.6/library/collections.abc.html#collections.abc.Mapping) 的相等测试，是顺序敏感的字典测试。这允许 [OrderedDict](https://docs.python.org/zh-cn/3.6/library/collections.html#collections.OrderedDict) 替换为任何字典可以使用的场所。

*在 3.5 版更改:*[OrderedDict](https://docs.python.org/zh-cn/3.6/library/collections.html#collections.OrderedDict) 的项(item)，键(key)和值(value) [视图](https://docs.python.org/zh-cn/3.6/glossary.html#term-dictionary-view) 现在支持逆序迭代，通过 [reversed()](https://docs.python.org/zh-cn/3.6/library/functions.html#reversed) 。

*在 3.6 版更改:*[**PEP 468**](https://www.python.org/dev/peps/pep-0468) 赞成将关键词参数的顺序保留， 通过传递给 [OrderedDict](https://docs.python.org/zh-cn/3.6/library/collections.html#collections.OrderedDict) 构造器和它的 update() 方法。

8.3.6.1. [OrderedDict](https://docs.python.org/zh-cn/3.6/library/collections.html#collections.OrderedDict) 例子和用法

Since an ordered dictionary remembers its insertion order, it can be used in conjunction with sorting to make a sorted dictionary:

>>>

**>>>** *# regular unsorted dictionary*

**>>>** d = {'banana': 3, 'apple': 4, 'pear': 1, 'orange': 2}

**>>>** *# dictionary sorted by key*

**>>>** OrderedDict(sorted(d.items(), key=**lambda** t: t[0]))

OrderedDict([('apple', 4), ('banana', 3), ('orange', 2), ('pear', 1)])

**>>>** *# dictionary sorted by value*

**>>>** OrderedDict(sorted(d.items(), key=**lambda** t: t[1]))

OrderedDict([('pear', 1), ('orange', 2), ('banana', 3), ('apple', 4)])

**>>>** *# dictionary sorted by length of the key string*

**>>>** OrderedDict(sorted(d.items(), key=**lambda** t: len(t[0])))

OrderedDict([('pear', 1), ('apple', 4), ('orange', 2), ('banana', 3)])

The new sorted dictionaries maintain their sort order when entries are deleted. But when new keys are added, the keys are appended to the end and the sort is not maintained.

It is also straight-forward 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):

**if** key **in** self:

**del** self[key]

OrderedDict.\_\_setitem\_\_(self, key, value)

An ordered dictionary can be combined with the [Counter](https://docs.python.org/zh-cn/3.6/library/collections.html#collections.Counter) class so that the counter remembers the order elements are first encountered:

**class** **OrderedCounter**(Counter, OrderedDict):

'Counter that remembers the order elements are first encountered'

**def** \_\_repr\_\_(self):

**return** '*%s*(*%r*)' % (self.\_\_class\_\_.\_\_name\_\_, OrderedDict(self))

**def** \_\_reduce\_\_(self):

**return** self.\_\_class\_\_, (OrderedDict(self),)

8.3.7. [UserDict](https://docs.python.org/zh-cn/3.6/library/collections.html#collections.UserDict) 对象

[UserDict](https://docs.python.org/zh-cn/3.6/library/collections.html#collections.UserDict) 类是用作字典对象的外包装。对这个类的需求已部分由直接创建 [dict](https://docs.python.org/zh-cn/3.6/library/stdtypes.html#dict) 的子类的功能所替代；不过，这个类处理起来更容易，因为底层的字典可以作为属性来访问。

*class*collections.**UserDict**([*initialdata*])

模拟一个字典类。这个实例的内容保存为一个正常字典， 可以通过 [UserDict](https://docs.python.org/zh-cn/3.6/library/collections.html#collections.UserDict) 实例的 [data](https://docs.python.org/zh-cn/3.6/library/collections.html#collections.UserDict.data) 属性存取。如果提供了 *initialdata* 值， [data](https://docs.python.org/zh-cn/3.6/library/collections.html#collections.UserDict.data) 就被初始化为它的内容；注意一个 *initialdata* 的引用不会被保留作为其他用途。

[UserDict](https://docs.python.org/zh-cn/3.6/library/collections.html#collections.UserDict) 实例提供了以下属性作为扩展方法和操作的支持:

**data**

一个真实的字典，用于保存 [UserDict](https://docs.python.org/zh-cn/3.6/library/collections.html#collections.UserDict) 类的内容。

8.3.8. [UserList](https://docs.python.org/zh-cn/3.6/library/collections.html#collections.UserList) 对象

这个类封装了列表对象。它是一个有用的基础类，对于你想自定义的类似列表的类，可以继承和覆盖现有的方法，也可以添加新的方法。这样我们可以对列表添加新的行为。

对这个类的需求已部分由直接创建 [list](https://docs.python.org/zh-cn/3.6/library/stdtypes.html#list) 的子类的功能所替代；不过，这个类处理起来更容易，因为底层的列表可以作为属性来访问。

*class*collections.**UserList**([*list*])

模拟一个列表。这个实例的内容被保存为一个正常列表，通过 [UserList](https://docs.python.org/zh-cn/3.6/library/collections.html#collections.UserList) 的 [data](https://docs.python.org/zh-cn/3.6/library/collections.html#collections.UserList.data) 属性存取。实例内容被初始化为一个 *list* 的copy，默认为 [] 空列表。 *list* 可以是迭代对象，比如一个Python列表，或者一个 [UserList](https://docs.python.org/zh-cn/3.6/library/collections.html#collections.UserList) 对象。

[UserList](https://docs.python.org/zh-cn/3.6/library/collections.html#collections.UserList) 提供了以下属性作为可变序列的方法和操作的扩展:

**data**

一个 [list](https://docs.python.org/zh-cn/3.6/library/stdtypes.html#list) 对象用于存储 [UserList](https://docs.python.org/zh-cn/3.6/library/collections.html#collections.UserList) 的内容。

**子类化的要求:** [UserList](https://docs.python.org/zh-cn/3.6/library/collections.html#collections.UserList) 的子类需要提供一个构造器，可以无参数调用，或者一个参数调用。返回一个新序列的列表操作需要创建一个实现类的实例。它假定了构造器可以以一个参数进行调用，这个参数是一个序列对象，作为数据源。

如果一个分离的类不希望依照这个需求，所有的特殊方法就必须重写；请参照源代码进行修改。

8.3.9. [UserString](https://docs.python.org/zh-cn/3.6/library/collections.html#collections.UserString) 对象

[UserString](https://docs.python.org/zh-cn/3.6/library/collections.html#collections.UserString) 类是用作字符串对象的外包装。对这个类的需求已部分由直接创建 [str](https://docs.python.org/zh-cn/3.6/library/stdtypes.html#str) 的子类的功能所替代；不过，这个类处理起来更容易，因为底层的字符串可以作为属性来访问。

*class*collections.**UserString**([*sequence*])

Class that simulates a string or a Unicode string object. The instance’s content is kept in a regular string object, which is accessible via the data attribute of [UserString](https://docs.python.org/zh-cn/3.6/library/collections.html#collections.UserString) instances. The instance’s contents are initially set to a copy of *sequence*. The *sequence* can be an instance of [bytes](https://docs.python.org/zh-cn/3.6/library/stdtypes.html#bytes), [str](https://docs.python.org/zh-cn/3.6/library/stdtypes.html#str), [UserString](https://docs.python.org/zh-cn/3.6/library/collections.html#collections.UserString) (or a subclass) or an arbitrary sequence which can be converted into a string using the built-in [str()](https://docs.python.org/zh-cn/3.6/library/stdtypes.html#str) function.

*在 3.5 版更改:*新方法 \_\_getnewargs\_\_, \_\_rmod\_\_, casefold, format\_map, isprintable, 和 maketrans。

[目录](https://docs.python.org/zh-cn/3.6/contents.html)

* [8.3. **collections** — 容器数据类型](https://docs.python.org/zh-cn/3.6/library/collections.html)
  + [8.3.1. **ChainMap** 对象](https://docs.python.org/zh-cn/3.6/library/collections.html#chainmap-objects)
    - [8.3.1.1. **ChainMap** 例子和方法](https://docs.python.org/zh-cn/3.6/library/collections.html#chainmap-examples-and-recipes)
  + [8.3.2. **Counter** 对象](https://docs.python.org/zh-cn/3.6/library/collections.html#counter-objects)
  + [8.3.3. **deque** 对象](https://docs.python.org/zh-cn/3.6/library/collections.html#deque-objects)
    - [8.3.3.1. **deque** 用法](https://docs.python.org/zh-cn/3.6/library/collections.html#deque-recipes)
  + [8.3.4. **defaultdict** 对象](https://docs.python.org/zh-cn/3.6/library/collections.html#defaultdict-objects)
    - [8.3.4.1. **defaultdict** 例子](https://docs.python.org/zh-cn/3.6/library/collections.html#defaultdict-examples)
  + [8.3.5. **namedtuple()** 命名元组的工厂函数](https://docs.python.org/zh-cn/3.6/library/collections.html#namedtuple-factory-function-for-tuples-with-named-fields)
  + [8.3.6. **OrderedDict** 对象](https://docs.python.org/zh-cn/3.6/library/collections.html#ordereddict-objects)
    - [8.3.6.1. **OrderedDict** 例子和用法](https://docs.python.org/zh-cn/3.6/library/collections.html#ordereddict-examples-and-recipes)
  + [8.3.7. **UserDict** 对象](https://docs.python.org/zh-cn/3.6/library/collections.html#userdict-objects)
  + [8.3.8. **UserList** 对象](https://docs.python.org/zh-cn/3.6/library/collections.html#userlist-objects)
  + [8.3.9. **UserString** 对象](https://docs.python.org/zh-cn/3.6/library/collections.html#userstring-objects)