第二章 第三节 Python入门之collections模块

Python中 collections模块的详细用法介绍

1. 介绍

collections是Python内建的一个集合模块，提供了许多有用的集合类和方法。

可以把它理解为一个容器，里面提供Python标准内建容器 dict , list , set , 和 tuple 的替代选择。

```
import collections
print(dir(collections))
```

['ChainMap', 'Counter', 'OrderedDict', 'UserDict', 'UserList', 'UserString', '_Link', '_OrderedDictItemsView', '_OrderedDictKeysView', '_OrderedDictValuesView', '__all__', '__builtins__', '__cached__', '__doc__', '__file__', '__loader__', '__name__', '__package__', '__path__', '__spec__', '_chain', '_collections_abc', '_count_elements', '_deque_iterator', '_eq', '_iskeyword', '_itemgetter', '_proxy', '_recursive_repr', '_repeat', '_starmap', '_sys', '_tuplegetter', 'abc', 'defaultdict', 'deque', 'namedtuple']

第二章 第三节 Python入门之collections模块

Python中 collections模块的详细用法介绍

2.常用方法

namedtuple()： 创建一个命名元组子类的工厂函数

deque： 高效增删改双向列表，类似列表(list)的容器，实现了在两端快速添加(append)和弹出(pop)

defaultdict： 当字典查询时，为key不存在提供一个默认值。

OrderedDict： 有序词典，就是记住了插入顺序

Counter： 计数功能

1. namedtuple() 命名元组

参数

第二章 第三节 Python入门之collections模块

Python中 collections模块的详细用法介绍

collections.namedtuple(typename, field_names, *, rename=False, defaults=None, module=None)

typename： 命名的名字，返回一个新的元组子类，名为 typename

field_names： 可以是一个['x', 'y']这样的序列，也可以是'x, y'或者'x y'

rename : python3.1添加，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 。

defaults： python3.7添加， defaults 可以为 None 或者是一个默认值的 iterable（可迭代对象）。如果一个默认值域必须跟其他没有默认值的域在一起出现， defaults 就应用到最右边的参数。比如如果域名 ['x', 'y', 'z'] 和默认值 (1, 2) ， 那么 x 就必须指定一个参数值 ， y 默认值 1 ， z 默认值 2 。

module： python3.6添加，如果 module 值有定义，命名元组的 __module__属性值就被设置。

使用

例如我想定义一个点(x, y)，可以给它起个名字为Points

```
import collections point = collections.namedtuple('Points', ['x', 'y'])
p1 = point(2, 3)
p2 = point(4, 2)
print(p1) # Points(x=2, y=3)
print(p2) # Points(x=4, y=2)
```

用 isinstance 判断其类型

```
print(isinstance(p1, point)) # True
print(isinstance(p1, tuple)) # True
```

第二章 第三节 Python入门之collections模块

Python中 collections模块的详细用法介绍

可以发现它即属于 point 类型，也属于 tuple 类型。

使用 _make 赋值

```
a= [11, 3]
p1._make(a)
print(p1) # Points(x=11, y=3)
```

使用 _replace 更改值

```
p1._replace(x=5)
print(p1) # Points(x=5, y=3)
```

2. deque 双端队列

参数

```
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())，从 iterable （迭代对象) 数据创建。如果 iterable 没有指定，新队列为空。

•iterable： 迭代对象，可以是字符串，列表等可迭代对象。

•maxlen： maxlen 没有指定或者是 None， deque 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。

使用

第二章 第三节 Python入门之collections模块

Python中 collections模块的详细用法介绍

```
from collections import deque
q = deque(['a', 'b', 'c'], maxlen=10)
# 从右边添加一个元素
q.append('d’)
print(q) # deque(['a', 'b', 'c', 'd'], maxlen=10)
# 从左边删除一个元素
print(q.popleft()) # a
print(q) # deque(['b', 'c', 'd'], maxlen=10)
# 扩展队列
q.extend(['i', 'j’])
print(q) # deque(['b', 'c', 'd', 'i', 'j'], maxlen=10)
# 查找下标
print(q.index('c')) # 1
# 移除第一个'd’
q.remove('d’)
print(q) # deque(['b', 'c', 'i', 'j'], maxlen=10)
# 逆序
q.reverse()
print(q) # deque(['j', 'i', 'c', 'b'], maxlen=10)
# 最大长度
print(q.maxlen) # 10
```

第二章 第三节 Python入门之collections模块

Python中 collections模块的详细用法介绍

全部方法

里面有许多方法，我们只介绍常用的方法。

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中个数等于 x 的元素。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]])：返回第 x 个元素（从 start 开始计算，在 stop 之前）。返回第一个匹配，如果没找到的话，升起 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，就升起一个 IndexError 。

第二章 第三节 Python入门之collections模块

Python中 collections模块的详细用法介绍

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，就升起 IndexError 索引错误。

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，就升起 IndexError 索引错误。

remove(value)：移去找到的第一个 value。 如果没有的话就升起 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 。

第二章 第三节 Python入门之collections模块

Python中 collections模块的详细用法介绍

3. defaultdict 默认值字典

使用

当key不存在时返回默认值

```
from collections import defaultdict
dd = defaultdict(lambda: 'not exist’)
dd['key1'] = 'abc’
print(dd['key1']) # key1存在
# 'abc’
print(dd['key2']) # key2不存在，返回默认值
# 'not exist’
```

使用 list 作为 default_factory，很容易将序列作为键值对加入字典:

```
from collections import defaultdict
d = defaultdict(list)
s = [('yellow', 1), ('blue', 2), ('yellow', 3), ('blue', 4), ('red', 1)]
for k, v in s:
    d[k].append(v)
    print(d) # defaultdict(<class 'list'>, {'yellow': [1, 3], 'blue': [2, 4], 'red': [1]})
```

第二章 第三节 Python入门之collections模块

Python中 collections模块的详细用法介绍

4. OrderedDict 有序字典

有序词典就像常规词典一样，但有一些与排序操作相关的额外功能。

但是内置的 dict 类已经有了记住插入顺序的能力（在 Python 3.7 中保证了这种新行为），所以它变得不那么重要了。

使用

popitem(last=True)： 有序字典的 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。

```
from collections import OrderedDict
d = OrderedDict(a=1, b=2, c=3, d=4,e=5)
print(d) # OrderedDict([('a', 1), ('b', 2), ('c', 3), ('d', 4), ('e', 5)])
print(d.popitem(last=True)) # ('e', 5)
print(d.popitem(last=False)) # ('a', 1)
print(d) # OrderedDict([('b', 2), ('c', 3), ('d', 4)]
```

第二章 第三节 Python入门之collections模块

Python中 collections模块的详细用法介绍

4. OrderedDict 有序字典

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。

```
from collections import OrderedDict
d = OrderedDict(a=1, b=2, c=3, d=4,e=5)
print(d) # OrderedDict([('a', 1), ('b', 2), ('c', 3), ('d', 4), ('e', 5)])
d.move_to_end(key='c', last=True)
print(d) # OrderedDict([('a', 1), ('b', 2), ('d', 4), ('e', 5), ('c', 3)])
d.move_to_end(key='b', last=False)
print(d) # OrderedDict([('b', 2), ('a', 1), ('d', 4), ('e', 5), ('c', 3)]
```

第二章 第三节 Python入门之collections模块

Python中 collections模块的详细用法介绍

5. Counter 计数

Counter 是一个 dict 的子类，用于计数可哈希对象。特别方便!

使用

字符串

```
from collections import Counter
c = Counter() for i in 'sfsadfsdjklgsdla’:
c[i] += 1
print(isinstance(c,Counter)) # True
print(isinstance(c,dict)) # True
print(c) # Counter({'s': 4, 'd': 3, 'f': 2, 'a': 2, 'l': 2, 'j': 1, 'k': 1, 'g': 1})
c2 = Counter('asfjslfjsdlfjgkls’)
print(c2) # Counter({'s': 4, 'd': 3, 'f': 2, 'a': 2, 'l': 2, 'j': 1, 'k': 1, 'g': 1})
```

第二章 第三节 Python入门之collections模块

Python中 collections模块的详细用法介绍

列表

```
from collections import Counter
c = Counter(['red', 'blue', 'red', 'green', 'blue', 'blue’])
print(c) # Counter({'blue': 3, 'red': 2, 'green': 1})
```

elements()：返回一个迭代器，其中每个元素将重复出现计数值所指定次。 元素会按首次出现的顺序返回。 如果一个元素的计数值小于一， elements() 将会忽略它。

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

most_common([n])：返回一个列表，其中包含 n 个最常见的元素及出现次数，按常见程度由高到低排序。 如果 n 被省略或为 None， most_common() 将返回计数器中的 所有 元素。 计数值相等的元素按首次出现的顺序排序:

```
c = Counter('abracadabra’)
print(c.most_common(3)) # [('a', 5), ('b', 2), ('r', 2)]
```

subtract([iterable-or-mapping])：从 迭代对象 或 映射对象 减去元素。像 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)
print(c) # Counter({'a': 3, 'b': 0, 'c': -3, 'd': -6})
```