

# Collections 模块

---

## 1.1 实验介绍

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

## 1.2 知识点

- Counter 类
- defaultdict 类
- namedtuple 类

## 1.3 实验环境

- python3.5
- Xfce终端
- Vim

## 1.4 适合人群

本课程属于初级级别课程，不仅适用于那些有其它语言基础的同学，对没有编程经验的同学也非常友好

# 二、实验步骤

---

## 2.1 Counter

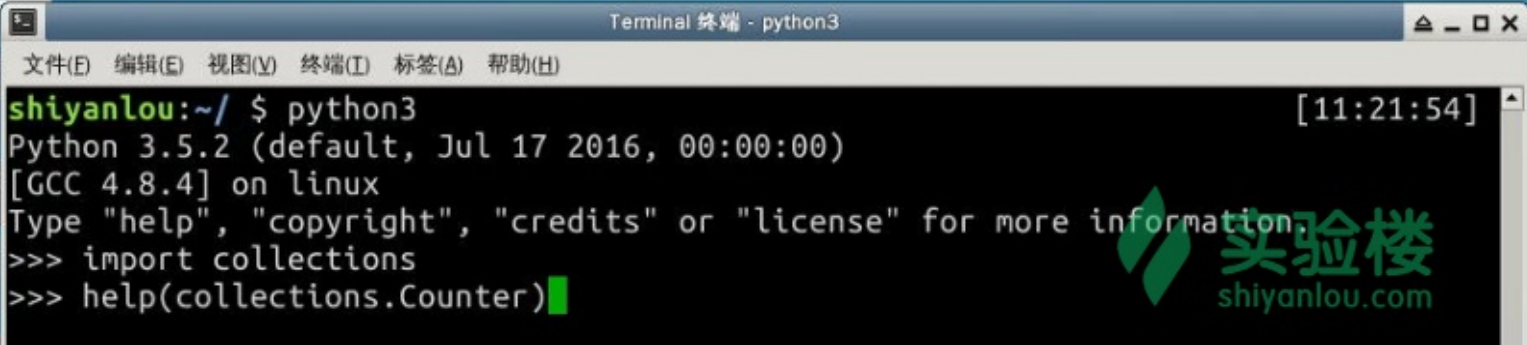
在这个实验我们会学习 `Collections` 模块。这个模块实现了一些很好的数据结构，它们能帮助你解决各种实际问题。

```
>>> import collections
```

这是如何导入这个模块，现在我们来看看其中的一些类。

`Counter` 是一个有助于 *hashable* 对象计数的 `dict` 子类。它是一个无序的集合，其中 *hashable* 对象的元素存储为字典的键，它们的计数存储为字典的值，计数可以为任意整数，包括零和负数。

我们可以这样查看 `Counter` 的帮助信息，事实上这些信息来源于 `Counter` 的文档字符串（`collections.Counter.__doc__`）。

A screenshot of a terminal window titled "Terminal 终端 - python3". The terminal shows the following commands and output:

```
shiyancelou:~/ $ python3
Python 3.5.2 (default, Jul 17 2016, 00:00:00)
[GCC 4.8.4] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import collections
>>> help(collections.Counter)
```

On the right side of the terminal window, there is a green logo and the text "实验楼 shiyanlou.com". The time "[11:21:54]" is displayed in the top right corner of the terminal area.

Help on class Counter in module collections:

```
class Counter(builtins.dict)
| Dict subclass for counting hashable items. Sometimes called a bag
| or multiset. Elements are stored as dictionary keys and their counts
| are stored as dictionary values.
|
| >>> c = Counter('abcdeabcdabcaba') # count elements from a string
|
| >>> c.most_common(3)                 # three most common elements
| [('a', 5), ('b', 4), ('c', 3)]
| >>> sorted(c)                         # list all unique elements
| ['a', 'b', 'c', 'd', 'e']
| >>> ''.join(sorted(c.elements()))    # list elements with repetitions
| 'aaaaabbbbcccdde'
| >>> sum(c.values())                  # total of all counts
| 15
|
| >>> c['a']                           # count of letter 'a'
| 5
| >>> for elem in 'shazam':            # update counts from an iterable
| ...     c[elem] += 1                # by adding 1 to each element's count
| >>> c['a']                           # now there are seven 'a'
```

下面我们来看一个例子，例子中我们查看 Python 的 LICENSE 文件中某些单词出现的次数。

### 2.1.1 Counter 示例

```
>>> from collections import Counter
>>> import re
>>> path = '/usr/lib/python3.4/LICENSE.txt'
>>> words = re.findall('\w+', open(path).read().lower())
>>> Counter(words).most_common(10)
[('the', 80), ('or', 78), ('1', 66), ('of', 61), ('to', 50), ('and', 48), ('python', 46), ('in', 38), ('license', 37), ('any', 37)]
```

Counter 对象有一个叫做 `elements()` 的方法，其返回的序列中，依照计数重复元素相同次数，元素顺序是无序的。

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

`most_common()` 方法返回最常见的元素及其计数，顺序为最常见到最少。

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

## 2.2 defaultdict

`defaultdict` 是内建 `dict` 类的子类，它覆写了一个方法并添加了一个可写的实例变量。其余功能与字典相同。

`defaultdict()` 第一个参数提供了 `default_factory` 属性的初始值，默认值为 `None`，`default_factory` 属性值将作为字典的默认数据类型。所有剩余的参数与字典的构造方法相同，包括关键字参数。

同样的功能使用 `defaultdict` 比使用 `dict.setdefault` 方法快。

### defaultdict 用例

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

在例子中你可以看到，即使 `defaultdict` 对象不存在某个键，它会自动创建一个空列表。

## 2.3 namedtuple

命名元组有助于对元组每个位置赋予意义，并且让我们的代码有更好的可读性和自文档性。你可以在任何使用元组地方使用命名元组。在例子中我们会创建一个命名元组以展示为元组每个位置保存信息。

```
>>> from collections import namedtuple
>>> Point = namedtuple('Point', ['x', 'y']) # 定义命名元组
>>> p = Point(10, y=20) # 创建一个对象
>>> p
Point(x=10, y=20)
>>> p.x + p.y
30
>>> p[0] + p[1] # 像普通元组那样访问元素
30
>>> x, y = p # 元组拆封
>>> x
10
>>> y
20
```

## 三、总结

---

这个实验我们使用了 Collections 中的一些数据结构，可能你目前并用不上他，但希望你以后需要的时候会想起它们：-)

*\*本课程内容，由作者授权实验楼发布，未经允许，禁止转载、下载及非法传播。*

上一节：[模块 \(/courses/596/labs/2047/document\)](/courses/596/labs/2047/document)

下一节：[挑战：类和Collection模块 \(/courses/596/labs/2775/document\)](/courses/596/labs/2775/document)