1.7 字典排序¶

问题¶

你想创建一个字典，并且在迭代或序列化这个字典的时候能够控制元素的顺序。

解决方案¶

为了能控制一个字典中元素的顺序，你可以使用 `collections` 模块中的 `OrderedDict` 类。 在迭代操作的时候它会保持元素被插入时的顺序，示例如下：

```
from collections import OrderedDict

d = OrderedDict()
d['foo'] = 1
d['bar'] = 2
d['spam'] = 3
d['grok'] = 4
# Outputs "foo 1", "bar 2", "spam 3", "grok 4"
for key in d:
    print(key, d[key])
```

当你想要构建一个将来需要序列化或编码成其他格式的映射的时候， `OrderedDict` 是非常有用的。 比如，你想精确控制以 JSON 编码后字段的顺序，你可以先使用 `OrderedDict` 来构建这样的数据：

```
>>> import json
>>> json.dumps(d)
'{"foo": 1, "bar": 2, "spam": 3, "grok": 4}'
>>>
```

讨论¶

`OrderedDict` 内部维护着一个根据键插入顺序排序的双向链表。每次当一个新的元素插入进来的时候， 它会被放到链表的尾部。对于一个已经存在的键的重复赋值不会改变键的顺序。

需要注意的是，一个 `OrderedDict` 的大小是一个普通字典的两倍，因为它内部维护着另外一个链表。 所以如果你要构建一个需要大量 `OrderedDict` 实例的数据结构的时候（比如读取 100,000 行 CSV 数据到一个 `OrderedDict` 列表中去）， 那么你就得仔细权衡一下是否使用 `OrderedDict` 带来的好处要大过额外内存消耗的影响。