1.10 删除序列相同元素并保持顺序¶

问题¶

怎样在一个序列上面保持元素顺序的同时消除重复的值？

解决方案¶

如果序列上的值都是 hashable 类型，那么可以很简单的利用集合或者生成器来解决这个问题。比如：

```
def dedupe(items):
    seen = set()
    for item in items:
        if item not in seen:
            yield item
            seen.add(item)
```

下面是使用上述函数的例子：

```
>>> a = [1, 5, 2, 1, 9, 1, 5, 10]
>>> list(dedupe(a))
[1, 5, 2, 9, 10]
>>>
```

这个方法仅仅在序列中元素为 hashable 的时候才管用。 如果你想消除元素不可哈希（比如 dict 类型）的序列中重复元素的话，你需要将上述代码稍微改变一下，就像这样：

```
def dedupe(items, key=None):
    seen = set()
    for item in items:
        val = item if key is None else key(item)
        if val not in seen:
            yield item
            seen.add(val)
```

这里的key参数指定了一个函数，将序列元素转换成 hashable 类型。下面是它的用法示例：

```
>>> a = [ {'x':1, 'y':2}, {'x':1, 'y':3}, {'x':1, 'y':2}, {'x':2, 'y':4}]
>>> list(dedupe(a, key=lambda d: (d['x'],d['y'])))
[{'x': 1, 'y': 2}, {'x': 1, 'y': 3}, {'x': 2, 'y': 4}]
>>> list(dedupe(a, key=lambda d: d['x']))
[{'x': 1, 'y': 2}, {'x': 2, 'y': 4}]
>>>
```

如果你想基于单个字段、属性或者某个更大的数据结构来消除重复元素，第二种方案同样可以胜任。

讨论¶

如果你仅仅就是想消除重复元素，通常可以简单的构造一个集合。比如：

```
>>> a
[1, 5, 2, 1, 9, 1, 5, 10]
>>> set(a)
{1, 2, 10, 5, 9}
>>>
```

然而，这种方法不能维护元素的顺序，生成的结果中的元素位置被打乱。而上面的方法可以避免这种情况。

在本节中我们使用了生成器函数让我们的函数更加通用，不仅仅是局限于列表处理。 比如，如果如果你想读取一个文件，消除重复行，你可以很容易像这样做：

```
with open(somefile,'r') as f:
for line in dedupe(f):
```

...

上述key函数参数模仿了 `sorted()`, `min()` 和 `max()` 等内置函数的相似功能。 可以参考 1.8 和 1.13 小节了解更多。