1.13 通过某个关键字排序一个字典列表¶

问题¶

你有一个字典列表，你想根据某个或某几个字典字段来排序这个列表。

解决方案¶

通过使用 `operator` 模块的 `itemgetter` 函数，可以非常容易的排序这样的数据结构。 假设你从数据库中检索出来网站会员信息列表，并且以下列的数据结构返回：

```
rows = [
    {'fname': 'Brian', 'lname': 'Jones', 'uid': 1003},
    {'fname': 'David', 'lname': 'Beazley', 'uid': 1002},
    {'fname': 'John', 'lname': 'Cleese', 'uid': 1001},
    {'fname': 'Big', 'lname': 'Jones', 'uid': 1004}
]
```

根据任意的字典字段来排序输入结果行是很容易实现的，代码示例：

```
from operator import itemgetter
rows_by_fname = sorted(rows, key=itemgetter('fname'))
rows_by_uid = sorted(rows, key=itemgetter('uid'))
print(rows_by_fname)
print(rows_by_uid)
```

代码的输出如下：

```
[{'fname': 'Big', 'uid': 1004, 'lname': 'Jones'},
{'fname': 'Brian', 'uid': 1003, 'lname': 'Jones'},
{'fname': 'David', 'uid': 1002, 'lname': 'Beazley'},
{'fname': 'John', 'uid': 1001, 'lname': 'Cleese'}]
[{'fname': 'John', 'uid': 1001, 'lname': 'Cleese'},
{'fname': 'David', 'uid': 1002, 'lname': 'Beazley'},
{'fname': 'Brian', 'uid': 1003, 'lname': 'Jones'},
{'fname': 'Big', 'uid': 1004, 'lname': 'Jones'}]
```

`itemgetter()` 函数也支持多个 keys，比如下面的代码

```
rows_by_lfname = sorted(rows, key=itemgetter('lname','fname'))
print(rows_by_lfname)
```

会产生如下的输出：

```
[{'fname': 'David', 'uid': 1002, 'lname': 'Beazley'},
{'fname': 'John', 'uid': 1001, 'lname': 'Cleese'},
{'fname': 'Big', 'uid': 1004, 'lname': 'Jones'},
{'fname': 'Brian', 'uid': 1003, 'lname': 'Jones'}]
```

讨论¶

在上面例子中， `rows` 被传递给接受一个关键字参数的 `sorted()` 内置函数。 这个参数是 `callable` 类型，并且从 `rows` 中接受一个单一元素，然后返回被用来排序的值。 `itemgetter()` 函数就是负责创建这个 `callable` 对象的。

`operator.itemgetter()` 函数有一个被 `rows` 中的记录用来查找值的索引参数。可以是一个字典键名称， 一个整形值或者任何能够传入一个对象的 `__getitem__()` 方法的值。 如果你传入多个索引参数给 `itemgetter()` ，它生成的 `callable` 对象会返回一个包含所有元素值的元组， 并且 `sorted()` 函数会根据这个元组中元素顺序去排序。 但你想要同时在几个字段上面进行排序（比如通过姓和名来排序，也就是例子中的那样）的时候这种方法是很有用的。

`itemgetter()` 有时候也可以用 `lambda` 表达式代替，比如：

```
rows_by_fname = sorted(rows, key=lambda r: r['fname'])
rows_by_lfname = sorted(rows, key=lambda r: (r['lname'],r['fname']))
```

这种方案也不错。但是，使用 `itemgetter()` 方式会运行的稍微快点。因此，如果你对性能要求比较高的话就使用 `itemgetter()` 方式。

最后，不要忘了这节中展示的技术也同样适用于 `min()` 和 `max()` 等函数。比如：

```
>>> min(rows, key=itemgetter('uid'))
{'fname': 'John', 'lname': 'Cleese', 'uid': 1001}
>>> max(rows, key=itemgetter('uid'))
{'fname': 'Big', 'lname': 'Jones', 'uid': 1004}
>>>
```