1.8 字典的运算¶

问题¶

怎样在数据字典中执行一些计算操作（比如求最小值、最大值、排序等等）？

解决方案¶

考虑下面的股票名和价格映射字典：

```
prices = {
    'ACME': 45.23,
    'AAPL': 612.78,
    'IBM': 205.55,
    'HPQ': 37.20,
    'FB': 10.75
}
```

为了对字典值执行计算操作，通常需要使用 `zip()` 函数先将键和值反转过来。 比如，下面是查找最小和最大股票价格和股票值的代码：

```
min_price = min(zip(prices.values(), prices.keys()))
# min_price is (10.75, 'FB')
max_price = max(zip(prices.values(), prices.keys()))
# max_price is (612.78, 'AAPL')
```

类似的，可以使用 `zip()` 和 `sorted()` 函数来排列字典数据：

```
prices_sorted = sorted(zip(prices.values(), prices.keys()))
# prices_sorted is [(10.75, 'FB'), (37.2, 'HPQ'),
#                   (45.23, 'ACME'), (205.55, 'IBM'),
#                   (612.78, 'AAPL')]
```

执行这些计算的时候，需要注意的是 `zip()` 函数创建的是一个只能访问一次的迭代器。 比如，下面的代码就会产生错误：

```
prices_and_names = zip(prices.values(), prices.keys())
print(min(prices_and_names)) # OK
print(max(prices_and_names)) # ValueError: max() arg is an empty sequence
```

讨论¶

如果你在一个字典上执行普通的数学运算，你会发现它们仅仅作用于键，而不是值。比如：

```
min(prices) # Returns 'AAPL'
max(prices) # Returns 'IBM'
```

这个结果并不是你想要的，因为你想要在字典的值集合上执行这些计算。 或许你会尝试着使用字典的 `values()` 方法来解决这个问题：

```
min(prices.values()) # Returns 10.75
max(prices.values()) # Returns 612.78
```

不幸的是，通常这个结果同样也不是你想要的。 你可能还想要知道对应的键的信息（比如那种股票价格是最低的？）。

你可以在 `min()` 和 `max()` 函数中提供 `key` 函数参数来获取最小值或最大值对应的键的信息。比如：

```
min(prices, key=lambda k: prices[k]) # Returns 'FB'
max(prices, key=lambda k: prices[k]) # Returns 'AAPL'
```

但是，如果还想要得到最小值，你又得执行一次查找操作。比如：

```
min_value = prices[min(prices, key=lambda k: prices[k])]
```

前面的 `zip()` 函数方案通过将字典"反转"为 (值，键) 元组序列来解决了上述问题。 当比较两个元组的时候，值会先进行比较，然后才是键。 这样的话你就能通过一条简单的语句就能很轻松的实现在字典上的求最值和排序操作了。

需要注意的是在计算操作中使用到了 (值，键) 对。当多个实体拥有相同的值的时候，键会决定返回结果。 比如，在执行 `min()` 和 `max()` 操作的时候，如果恰巧最小或最大值有重复的，那么拥有最小或最大键的实体会返回：

```
>>> prices = { 'AAA' : 45.23, 'ZZZ': 45.23 }
>>> min(zip(prices.values(), prices.keys()))
(45.23, 'AAA')
>>> max(zip(prices.values(), prices.keys()))
(45.23, 'ZZZ')
>>>
```