# Chapter 1

# 数据结构和算法

## 1.1 将序列分解为单独的变量

任何序列（或可迭代的对象）都可以通过一个简单的赋值操作来分解为单独的变量。唯一的要求是变量的总数和结构要与序列相吻合。只要对象恰好是可迭代的，那么就可以执行分解操作。这包括字符串、文件、迭代器以及生成器。当做分解操作时，有时候可能想丢弃某些特定的值。Python 并没有提供特殊的语法来实现这一点，但是通常可以选一个用不到的变量名，以此来作为要丢弃的值的名称。

## 1.2 从任意长度的可迭代对象中分解元素

Python 的“*表达式”可以用来解决这个问题。由*修饰的变量也可以位于列表的第一个位置。这类可迭代对象中会有一些已知的组件或模式（例如，元素 1 之后的所有内容都是电话号码），利用*表达式分解可迭代对象使得开发者能够轻松利用这些模式，而不必在可迭代对象中做复杂花哨的操作才能得到相关的元素。

当和某些特定的字符串处理操作相结合，比如做拆分（splitting）操作时，这种*式的语法所支持的分解操作也非常有用。

## 1.3 保存最后 $N$ 个元素

保存有限的历史记录可算是 collections.deque 的完美应用场景了。

更普遍的是，当需要一个简单的队列结构时，deque 可祝你一臂之力。如果不指定队列的大小，也就得到了一个无界限的队列，可以在两端执行添加和弹出操作。

从队列两端添加或弹出元素的复杂度都是 $O(1)$。这和列表不同，当从列表的头部插入或移除元素时，列表的复杂度为 $O(N)$。

## 1.4 找到最大或最小的 $N$ 个元素

heapq 模块中有两个函数—nlargest()和 nsmallest()。堆最重要的特性就是 heap[0]总是最小那个的元素。此外，接下来的元素可依次通过 heapq.heappop()方法轻松找到。该方法会将第一个元素（最小的）弹出，然后以第二小的元素取而代之（这个操作的复杂度是 $O(logN)$，$N$ 代表堆的大小）。

当所要找的元素数量相对较小时，函数 nlargest()和 nsmallest()才是最适用的。如果只是简单地想找到最小或最大的元素（$N = 1$ 时），那么用 min()和 max()会更加快。同样，如果 $N$ 和集合本身的大小差不多大，通常更快的方法是先对集合排序，然后做切片操作（例如，使用 sorted(items)[:N]或者 sorted(items)[-N:]）。应该要注意的是，nlargest()和 nsmallest()的实际实现会根据使用它们的方式而有所不同，可能会相应作出一些优化措施（比如，当 N 的大小同输入大小很接近时，就会采用排序的方法）。

## 1.5 实现优先级队列

## 1.6 在字典中将键映射到多个值上

为了能方便地创建这样的字典，可以利用 collections 模块中的 defaultdict 类。defaultdict的一个特点就是它会自动初始化第一个值，这样只需关注添加元素即可。

关于 defaultdict，需要注意的一个地方是，它会自动创建字典表项以待稍后的访问（即使这些表项当前在字典中还没有找到）。如果不想要这个功能，可以在普通的字典上调用 setdefault()方法来取代。

## 1.7 让字典保持有序

要控制字典中元素的顺序，可以使用 collections 模块中的 OrderedDict 类。当对字典做迭代时，它会严格按照元素初始添加的顺序进行。当想构建一个映射结构以便稍后对其做序列化或编码成另一种格式时，OrderedDict 就显得特别有用。请注意 OrderedDict 的大小是普通字典的 2 倍多，这是由于它额外创建的链表所致。(**读者注：Python 后续的版本字典默认是有序的，这个保留下来是为了兼容性问题**)

## 1.8 与字典有关的计算问题

如果尝试在字典上执行常见的数据操作，将会发现它们只会处理键，而不是值。

## 1.9 在两个字典中寻找相同点

要找出两个字典中的相同之处，只需通过 keys()或者 items()方法执行常见的集合操作即可。这些类型的操作也可用来修改或过滤掉字典中的内容。例如，假设想创建一个新的字典，其中会去掉某些键。

字典就是一系列键和值之间的映射集合。字典的 keys()方法会返回 keys-view 对象，其中暴露了所有的键。关于字典的键有一个很少有人知道的特性，那就是它们也支持常见的集合操作，比如求并集、交集和差集。因此，如果需要对字典的键做常见的集合操作，那么就能直接使用 keys-view 对象而不必先将它们转化为集合。

字典的 items()方法返回由(key,value)对组成的 items-view 对象。这个对象支持类似的集合操作，可用来完成找出两个字典间有哪些键值对有相同之处的操作。

尽管类似，但字典的 values()方法并不支持集合操作。部分原因是因为在字典中键和值是不同的，从值的角度来看并不能保证所有的值都是唯一的。单这一条原因就使得某些特定的集合操作是有问题的。但是，如果必须执行这样的操作，还是可以先将值转化为集合来实现。