8.4 创建大量对象时节省内存方法¶

问题¶

你的程序要创建大量(可能上百万)的对象，导致占用很大的内存。

解决方案¶

对于主要是用来当成简单的数据结构的类而言，你可以通过给类添加 `__slots__` 属性来极大的减少实例所占的内存。比如：

```
class Date:
    __slots__ = ['year', 'month', 'day']
    def __init__(self, year, month, day):
        self.year = year
        self.month = month
        self.day = day
```

当你定义 `__slots__` 后，Python就会为实例使用一种更加紧凑的内部表示。 实例通过一个很小的固定大小的数组来构建，而不是为每个实例定义一个字典，这跟元组或列表很类似。 在 `__slots__` 中列出的属性名在内部被映射到这个数组的指定小标上。 使用slots一个不好的地方就是我们不能再给实例添加新的属性了，只能使用在 `__slots__` 中定义的那些属性名。

讨论¶

使用slots后节省的内存会跟存储属性的数量和类型有关。 不过，一般来讲，使用到的内存总量和将数据存储在一个元组中差不多。 为了给你一个直观认识，假设你不使用slots直接存储一个Date实例， 在64位的Python上面要占用428字节，而如果使用了slots，内存占用下降到156字节。 如果程序中需要同时创建大量的日期实例，那么这个就能极大的减小内存使用量了。

尽管slots看上去是一个很有用的特性，很多时候你还是得减少对它的使用冲动。 Python的很多特性都依赖于普通的基于字典的实现。 另外，定义了slots后的类不再支持一些普通类特性了，比如多继承。 大多数情况下，你应该只在那些经常被使用到的用作数据结构的类上定义slots (比如在程序中需要创建某个类的几百万个实例对象)。

关于 `__slots__` 的一个常见误区是它可以作为一个封装工具来防止用户给实例增加新的属性。 尽管使用slots可以达到这样的目的，但是这个并不是它的初衷。 `__slots__` 更多的是用来作为一个内存优化工具。