```
        city=city,
    )

addr = latlon_to_address(lat, lon)
# 通过属性名来使用 addr
# addr.country / addr.province / addr.city
```

假如我们在 Address 里增加了新的返回值 district，已有的函数调用代码也不用进行任何适配性修改，因为函数结果只是多了一个新属性，没有任何破坏性影响。

3.4 总结

在本章中，我们简单介绍了四种内置容器类型，它们是 Python 语言最为重要的组成之一。在介绍这些容器类型的过程中，我们引申出了对象的可变性、可哈希性等诸多基础概念。

内置容器功能丰富，基于它构建的自定义容器更为强大，能帮助我们完成许多有趣的事情。在案例故事里，我们就通过一个自定义字典类型，优化了整个日志分析脚本。

虽然无须了解列表的底层实现原理就可以使用列表，但如果你深入理解了列表是基于数组实现的，就能避开一些性能陷阱，知道在什么情况下应该选择其他数据结构实现某些需求。所以，不论使用多么高级的编程语言，掌握基础的算法与数据结构知识永远不会过时。

以下是本章要点知识总结。

(1) 基础知识

- 在进行函数调用时，传递的不是变量的值或者引用，而是变量所指对象的引用
- Python 内置类型分为可变与不可变两种，可变性会影响一些操作的行为，比如 +=
- 对于可变类型，必要时对其进行拷贝操作，能避免产生意料之外的影响
- 常见的浅拷贝方式：copy.copy、推导式、切片操作
- 使用 copy.deepcopy 可以进行深拷贝操作

(2) 列表与元组

- 使用 enumerate 可以在遍历列表的同时获取下标
- 函数的多返回值其实是一个元组
- 不存在元组推导式，但可以使用 tuple 来将生成器表达式转换为元组
- 元组经常用来表示一些结构化的数据