第九章：元编程¶

软件开发领域中最经典的口头禅就是“don’t repeat yourself”。 也就是说，任何时候当你的程序中存在高度重复(或者是通过剪切复制)的代码时，都应该想想是否有更好的解决方案。 在Python当中，通常都可以通过元编程来解决这类问题。 简而言之，元编程就是关于创建操作源代码(比如修改、生成或包装原来的代码)的函数和类。 主要技术是使用装饰器、类装饰器和元类。不过还有一些其他技术， 包括签名对象、使用 `exec()` 执行代码以及对内部函数和类的反射技术等。 本章的主要目的是向大家介绍这些元编程技术，并且给出实例来演示它们是怎样定制化你的源代码行为的。

Contents:

9.1 在函数上添加包装器
9.2 创建装饰器时保留函数元信息
9.3 解除一个装饰器
9.4 定义一个带参数的装饰器
9.5 可自定义属性的装饰器
9.6 带可选参数的装饰器
9.7 利用装饰器强制函数上的类型检查
9.8 将装饰器定义为类的一部分
9.9 将装饰器定义为类
9.10 为类和静态方法提供装饰器
9.11 装饰器为被包装函数增加参数
9.12 使用装饰器扩充类的功能
9.13 使用元类控制实例的创建
9.14 捕获类的属性定义顺序
9.15 定义有可选参数的元类
9.16 *args和**kwargs的强制参数签名
9.17 在类上强制使用编程规约
9.18 以编程方式定义类
9.19 在定义的时候初始化类的成员
9.20 利用函数注解实现方法重载
9.21 避免重复的属性方法
9.22 定义上下文管理器的简单方法
9.23 在局部变量域中执行代码
9.24 解析与分析Python源码
9.25 拆解Python字节码