修饰符

杨资璋（翻译）

2021 年 10 月 12 日

1 警告警告警告

这篇文档意在描述修饰符的语法以及导致我们做出决定的过程。它并不准备涵盖广大的潜在可替代语法，也不准备详细地列出每一种形式的优缺点。

2 摘要

目前用于转化函数和方法的方法（例如，将他们声明为类或静态方法）有些尴尬，并可能导致代码难以理解。理想化情况下，这些转化应该与转化的声明同时产生。这个 PEP 介绍了转化函数或方法声明的新语法。

3 动机

目前为函数或方法进行转化的途径是在函数体后放置一个实际的转化。对于较大的函数，这将函数行为的一个关键组件与函数剩余的外部接口定义分离了。例如：

```
def foo(self):
    perform method operation
foo = classmethod(foo)
```

对于更长的方法，这边的更加难以阅读。在概念上实际上仅是单个声明但是却三次命名这个函数看起来更加不”python”。这个问题的一个解决方法是将方法的转化移到离方法本身的定义更近的地方。新语法的意在替换

```
def foo(self):
    pass
foo = synchronized(lock)(foo)
foo = classmethod(foo)
```

为一种在函数声明放置装饰的替代。

```
@classmethod
@synchronized(lock)
def foo(self):
    pass
```

通过这种形式修改类也是可能的，虽然好处并不是显而易见的。几乎可以肯定的是，任何可以通过类修饰符完成的事情也可以通过元类完成，然而使用元类是十分隐晦的，这使得使用一种更简单的方法修改类有一些吸引力。对于 Python2.4，只有函数和方法修饰符被加入了。

3.1 为什么它如此困难

4 背景

5 关于修饰符这个名字

6 设计目标

新的语法应该

- 使用任意封装，包括用户定义的可调用和已存在的自建`classmethod()`和`staticmethod()`。这个要求也意味着修饰符语法必须支持向封装构造者传入参数
- 对于每个定义支持多个封装
- 使得发生了什么显而易见；至少它应该显然到新用户可以在写自己的代码时忽视它
- 作为一种“一旦被解释很容易记住”的语法
- 不会使得将来的扩展更加困难
- 键入简单；使用它的程序可能非常频繁地使用它

- 不会使得快速浏览代码更加困难。对于搜索所有定义、一个特定的定义或者一个函数接收的参数也应该依旧简单。
- 不会将例如语言敏感的编辑器和其他“玩具分析工具”等次级支持工具变得不必要的复杂
- 允许后来的编译器为了修饰符优化

7 当前语法

8 语法的可替代者

9 当前的实现以及历史

10 例子