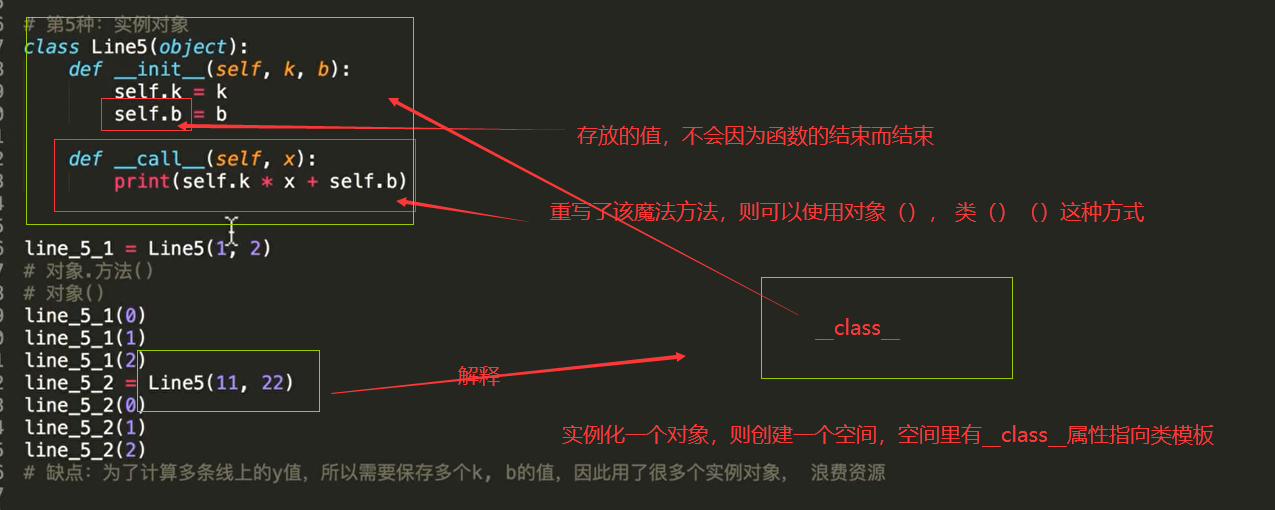
闭包和装饰器

详细介绍了python作用域的搜索规则：

<https://segmentfault.com/a/1190000000640834> 思否

面向对象的思想：



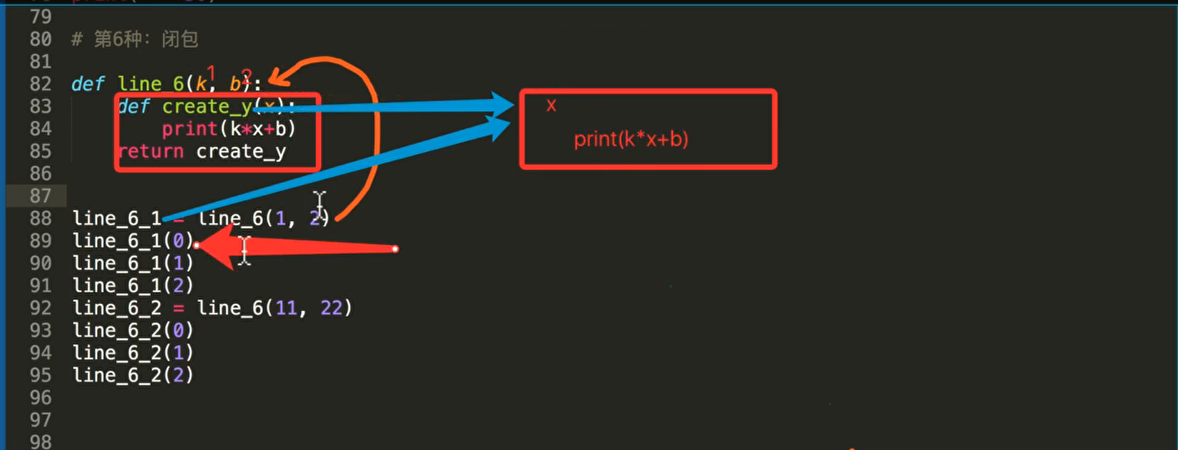
闭包：在内部方法中用到外部方法的变量，进行数据处理得到结果。

应用一：可以当做实参一起传递，并且可以一起传递过去变量和函数。

应用二：在装饰器中使用

面向对象实例化对象的空间，包括魔法方法和继承来的属性和方法，占用空间较大

闭包产生的空间，只包括外部方法的变量，和内部方法的代码，占用空间小，数据较干净



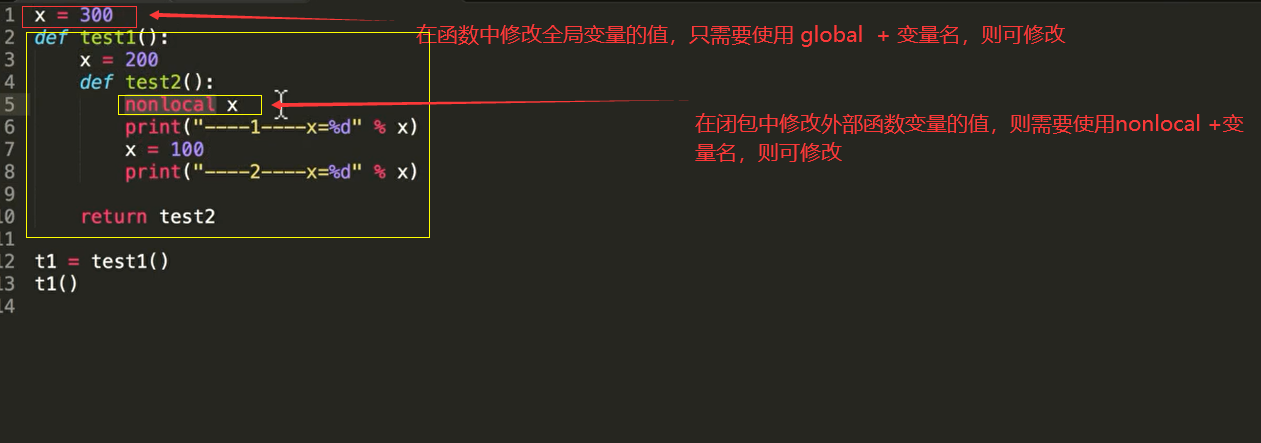
函数，匿名函数，对象，闭包的区别：

函数：封装一小部分代码，完成一个功能

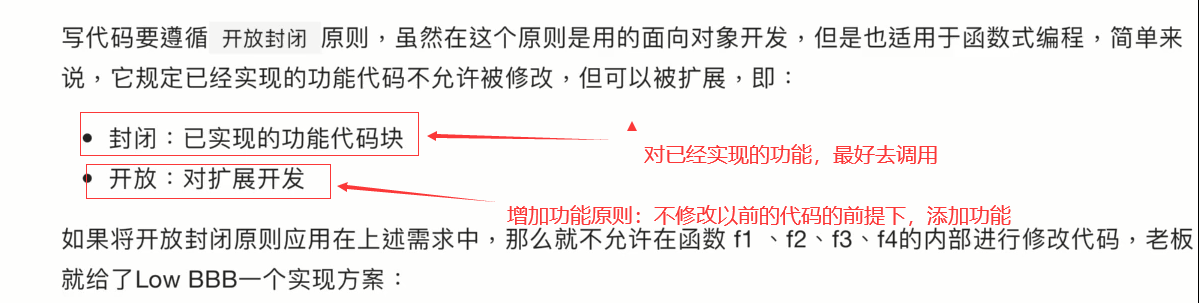
闭包：不仅有函数的功能，还能提供函数实现功能所需要的变量

对象：得到对象的引用，对象空间中数据较多，可能不需要

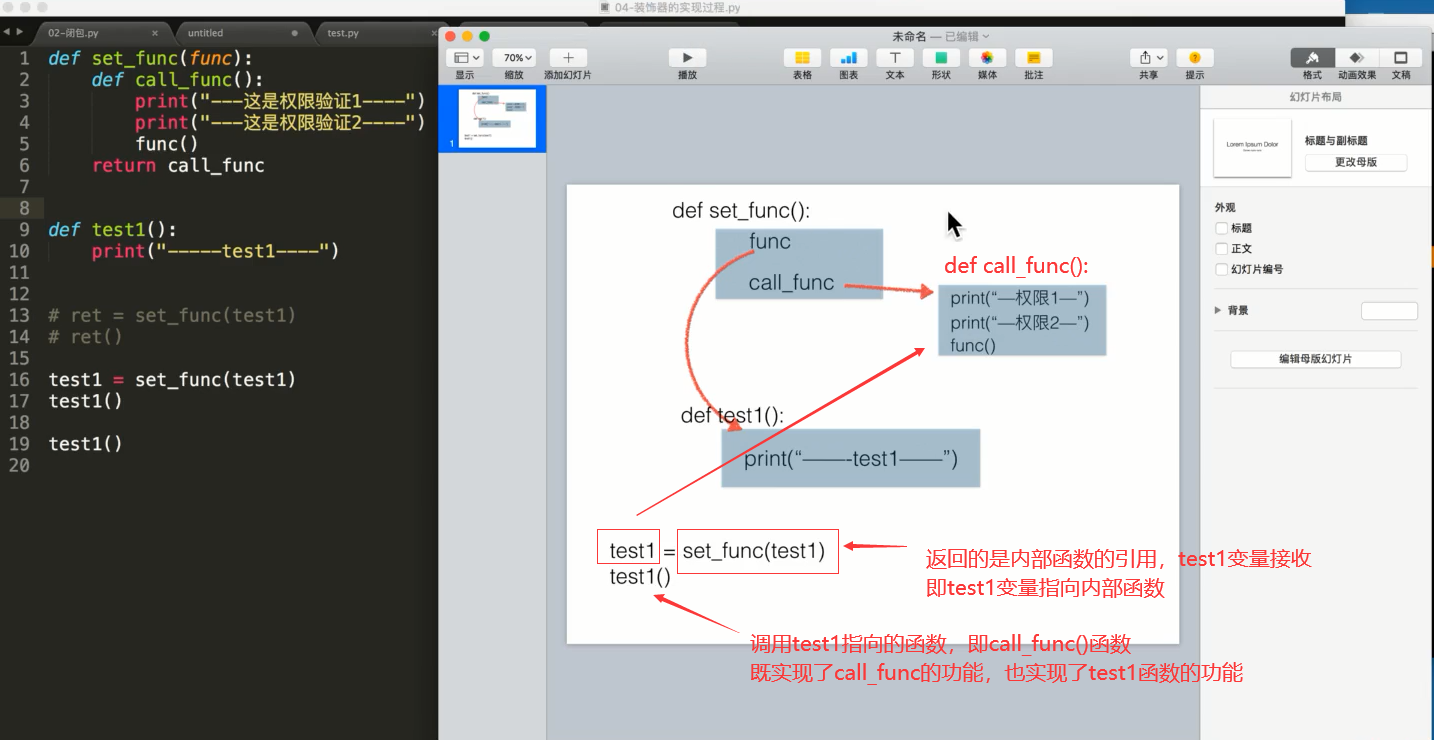
匿名函数：比函数写法更简便，不用考虑函数命名，更方便地完成函数的功能



装饰器引言：

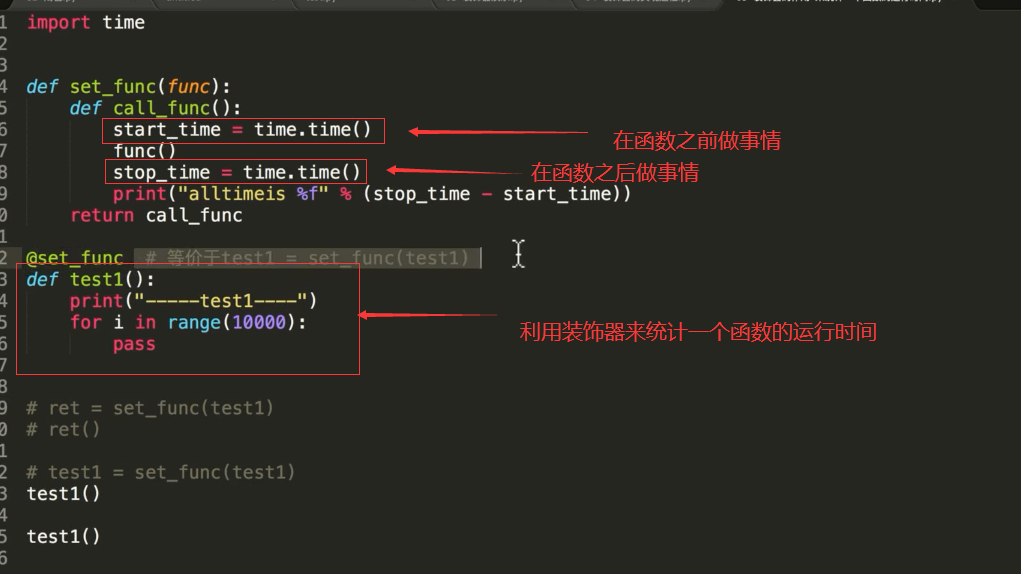


装饰器的好处：原有功能正常实现；装饰器在原有功能之上添加功能



可以利用装饰器来统计函数的运行时间

结论：可以在函数之前做事情，也可以在函数之后做事情，却不用改变原函数



装饰器的研究：

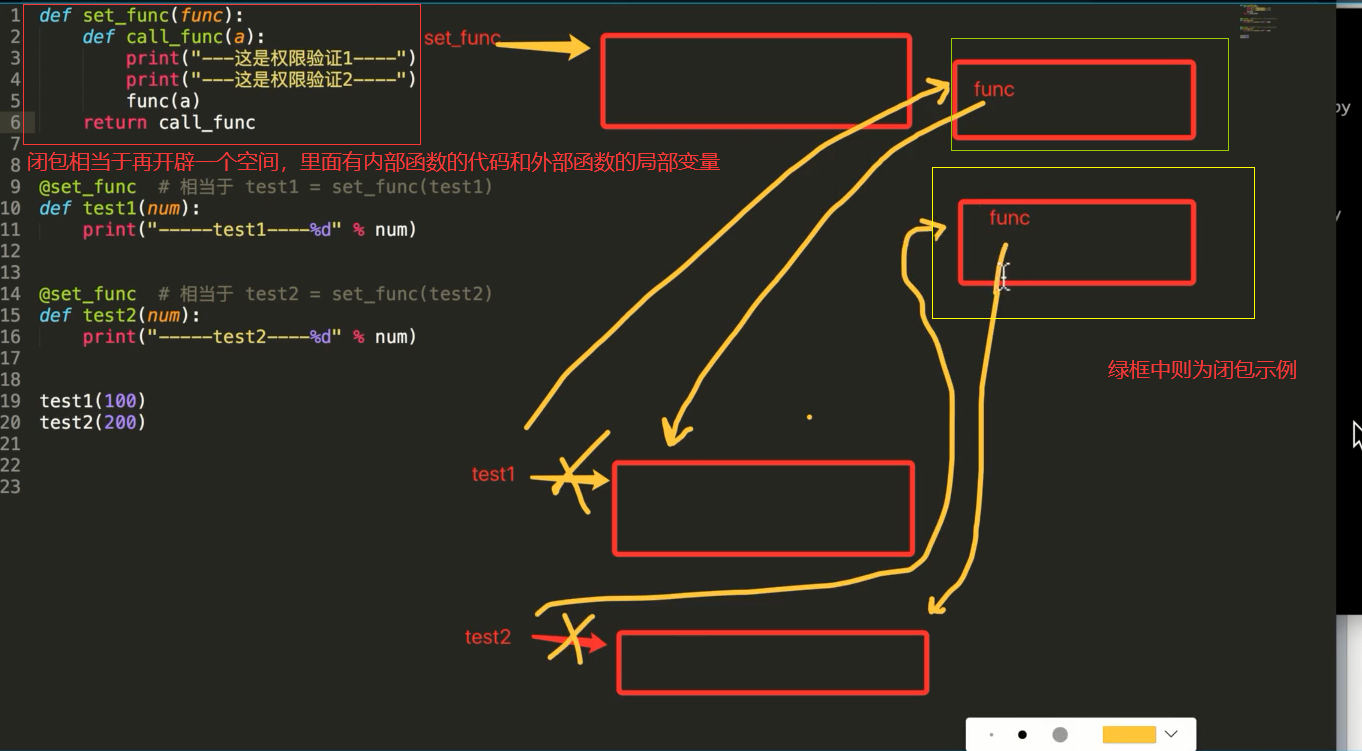
1、对无参数、无返回值的函数进行装饰，上例中test1函数就是例子

2、对有参数、无返回值的函数进行装饰

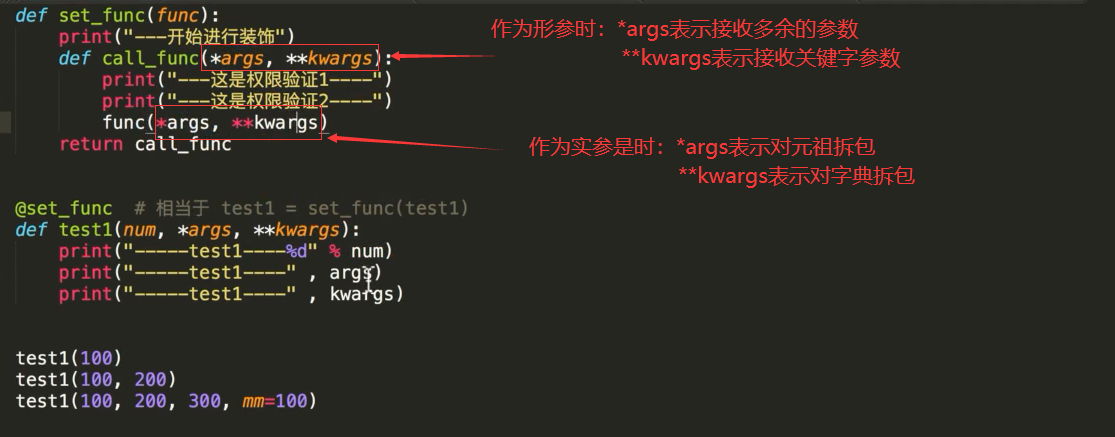


1. 同一个装饰器对多个函数进行装饰，相当于创建多个闭包空间。

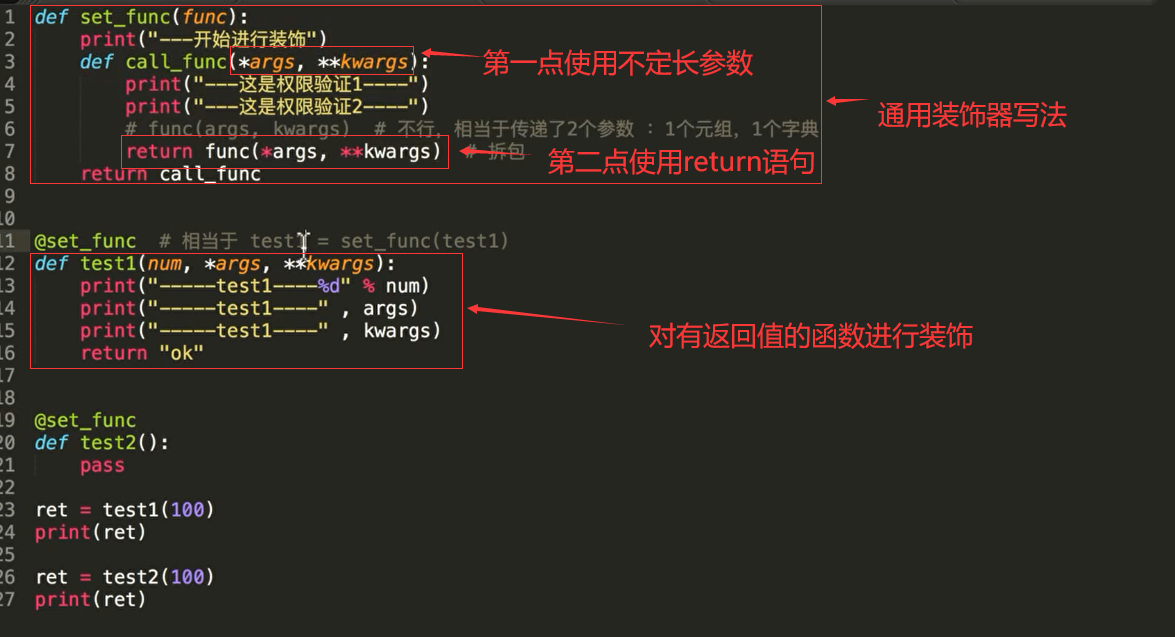
深刻理解点：遇到装饰器，python解释器就会装饰，而非是因为调用才装饰



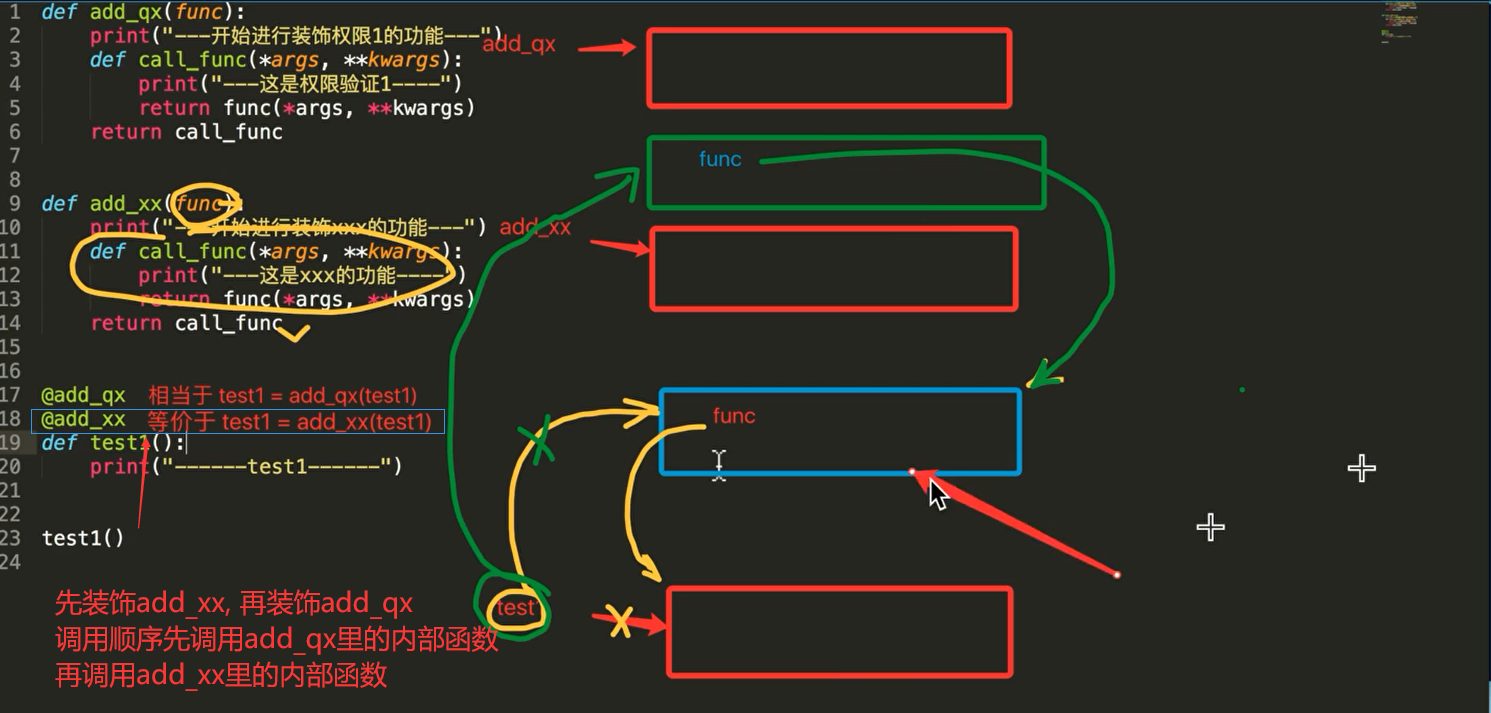
1. 对不定长参数进行装饰，最好就是直接使用不定长参数



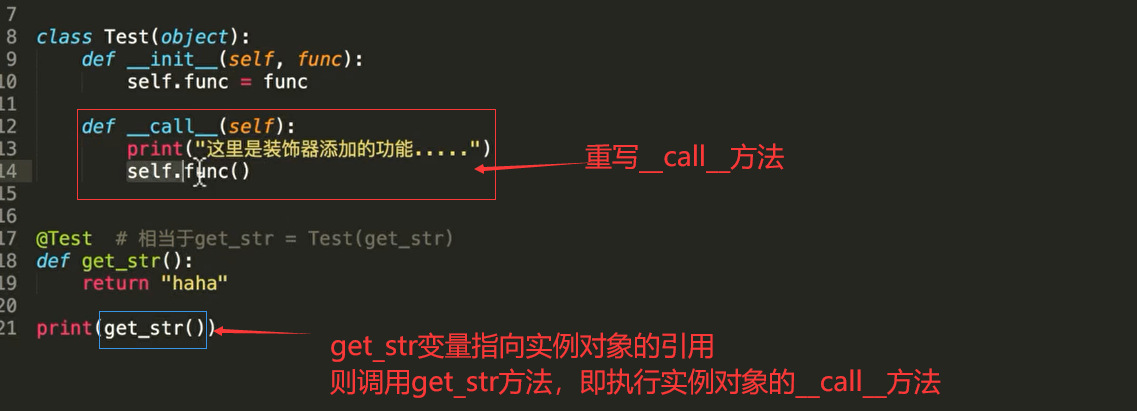
1. 对有返回值的函数进行装饰



1. 多个装饰器装饰同一个函数



1. 用类对函数进行装饰



[更复杂的装饰器@Test.staticmethod](mailto:更复杂的装饰器@Test.staticmethod) ,用类里面的静态方法进行装饰

或装饰器中带参数