**Python函数高阶之必会知识闭包装饰器**

# 本章所讲内容：

**1、装饰器使基础之闭包**

**2、装饰器之基于开发实现登录验证**

**1、闭包（面试题）**

**1.2.1 闭包定义**

**1. 闭包函数必须有内嵌函数；**

**2. 内嵌函数需要引用该嵌套函数上一级namespace中的变量；**

**3. 闭包函数必须返回内嵌函数；**

**通过这三点，就可以创建一个闭包；Python装饰器就是使用了闭包。**

**闭包的好处：**

**使代码变得简洁；**

**提高代码的拓展性；**

**案例：**

**def outer():  
 def inner():  
 print('我是inner')  
 return inner  
outer()()#进行连续调用**

### 1.2.2 闭包分析

**def outer(num):  
 def inner(num\_in):  
 print('inner,num\_in is %d'%num\_in)  
 return num+num\_in  
 return inner  
*# a 就相当于 inner*a = outer(20) *# inner  
# print(a)  
#a(200)==inner(200)*print(a(200))**

**施主，你会了吗？**



**2、装饰器(面试题)**

### 2.1 装饰器的定义

**装饰器是程序开发中经常会用到的一个功能，用好了装饰器，开发效率如虎添翼。**



**应用场景：**

**1、当我们已经写好一个函数时，项目也已经上线了，突然客户想要增添一个需求，让这个函数处理的更加细致，这个就可以用到装饰器了；**

**2、淘宝登录验证；**

**3、log日志；**

**执行顺序：**

**先返回内嵌函数🡪有参数直接传参给内嵌函数🡪执行内嵌函数**

**案例：**

**#装饰器快速入门假如现在我们定义了两个函数，分别来打印他们的名字。**

**def func1():  
 print('this is xurgod1')  
def func2():  
 print('this is xuegod2')  
func1()  
func2()**

**运行结果如下：**

**this is xurgod1**

**this is xuegod2**

**现在，我们要对这两个做个修改，要分别打印他们的国家，那我们第一时间想到的自然是给每个函数都去加上这个功能**

**def func1():  
 print('this is xuegod1')  
 print('I come from china')  
def func2():  
 print('this is xuegod2')  
 print('I come from china')**

**运行结果如下：**

**this is xuegod1**

**I come from china**

**this is xuegod2**

**I come from china**

**那如何有n个函数都要这么打印呢？还要一个个的去修改，显然很费劲，那我们想想如何用装饰器来实现这个新增的功能。**

**def func1():  
 print('this is xuegod1')  
def func2():  
 print('this is xuegod2')  
def outer(func):  
 def inner():  
 func()  
 print('come from china')  
 return inner**

**#inner 赋值给f1  
f1 = outer(func1)**

**#f1() 相当于inner的调用  
f1()**

**运行结果如下：**

**this is xuegod1**

**come from china**

**但是上面的操作我们在给一个函数进行装饰时候会调用f1这个函数！**

**python认为上面的写法不简洁，所以提出了@语法糖**

**def outer(func):  
 def inner():  
 func()  
 print('come from china')  
 return inner  
@outer # 等同于func1 = outer(func1)**

**#第一个func1 是为了接收返回的inner**

**#第二个func1是def func1()函数，作为一个实参传递给形参func。  
def func1():  
 print('this is xuegod1')  
@outer  
def func2():  
 print('this is xuegod2')  
func1()  
func2()**

**运行结果如下：**

**this is xuegod1**

**come from china**

**this is xuegod2**

**come from china**

**问题：**

**现在想调用的函数的时候传递name参数，并的func1和func2中调用name，来获取我们传进去的名字，怎么办？**

**def outer(func):  
 def inner(name):  
 *# 直接将name = for 传递给 p（name），调用p函数后，直接打印出name* func(name)  
 print('inner中的%s'%name)  
 return inner  
@outer # func1 = outer(func1)  
def func1(name): #name = ‘for’  
 print('this is xuegod1')  
 print('I am %s'%name)  
*# func1(name) = Dec(func1(name))  
#此时第一个func1中的name=‘for*func1('for') #inner(‘for’)   
#第一个func1 相当于inner 所以inner需要参数来接受for值**

**更加通用的装饰器：**

**def outer(func):  
 def inner(\*args,\*\*kwargs):  
 *# func(\*args,\*\*kwargs)相当于func1(5,5)* c = func(\*args,\*\*kwargs)  
 print(c)  
 return inner  
@outer  
*#func1 = dec(func1)*def func1(a,b=1):  
 return a\*b  
func1(5,5)**

**运行结果如下：**

**25**

**实战：简单实现log日志  
import datetime  
def outer(func):  
 def inner(\*args,\*\*kwargs):  
 print('时间:%s,%s 开始处理 (%s,%s)...'%(datetime.datetime.now(),func.\_\_name\_\_,args,kwargs))  
 return func(\*args,\*\*kwargs)  
 return inner  
@outer  
def first():  
 pass  
@outer  
def second(a,b,c):  
 pass  
if \_\_name\_\_ == '\_\_main\_\_':  
 first()  
 second(1,2,c=3)**

**实战：利用结合本节课知识点，让我们进入装饰器的分析，模拟淘宝加入购物车时候我们进行客户登录验证！**

 **import time  
a = ['while','for','django']  
def outer(func):  
 def inner(name):  
 *# name = 'for'* func(name)  
 print('开始判断你有没有登录')   
 time.sleep(2)  
 if name in a:  
 print('已经登录成功，请尽情访问')  
 time.sleep(1)  
 else:  
 print('你没有登录，没有访问权限')  
 time.sleep(1)  
 return inner  
@outer  
def login(name):  
 print('我要浏览')  
login('fo')**

**带有参数的装饰器：**

**def Out(info):  
 def Dec(func):  
 def inner(name):  
 print('info is %s'%info)  
 func(name)  
 print('come from china')  
 return inner  
 return Dec**

**#直接调用 返回 Dec  
@Out('装饰器传递参数')  
def func1(name):  
 print('this is xuegod1')  
 print('I am %s'%name)  
@Out('装饰器传递参数')  
def func2(name):  
 print('this is xuegod2')  
 print('I am %s' % name)  
#func1 = Dec(func1)  
func1('for')  
#func2 = Dec(func2)  
func2('while')**

**两个装饰器：**

**def set\_fun1(func1):  
 print("set\_fun1")  
 def call\_fun1():  
 print("call\_fun1")  
 func1()  
 return call\_fun1  
def set\_fun2(func2):  
 print("set\_fun2")  
 def call\_fun2():  
 print("call\_fun2")  
 func2()  
 return call\_fun2  
@set\_fun2  
@set\_fun1  
def test():  
 print("test")  
test()**

**总结：**

**1、装饰器使基础之闭包**

**2、装饰器之基于开发实现登录验证**