闭包：

笔记本：	总复习		
创建时间：	2018/5/13 19:30	**更新时间：**	2018/5/13 19:33
作者：	王伟		

闭包：

1.引出闭包的概念：

```
class Line5(object):
    def __init__(self, k, b):
        self.k = k
        self.b = b

    def __call__(self, x):
        print(self.k * x + self.b)


line_5_1 = Line5(1, 2)
# 对象.方法()
# 对象()  #会调用这个对象的__call__方法
line_5_1(0)
line_5_1(1)
line_5_1(2)

line_5_2 = Line5(11, 22)
line_5_2(0)
line_5_2(1)
line_5_2(2)

line_5_1(3)
# 缺点：为了计算多条线上的y值，所以需要保存多个k, b的值，因此用了很多个实例对象，又因为一个对象占用的资源相对来说比较大，所以浪费资源，因为对象继承了object类，里面还有很多的默认属性。

print("-"*50)

# 第6种：闭包
print("第6种：闭包")

def line_6(k, b):
    def create_y(x):
        print(k*x+b)
    return create_y

line_6_1 = line_6(1, 2)  # 创建一个新的闭包，让line_6_1指向
line_6_1(0)
line_6_1(1)
```

```
line_6_1(2)
line_6_2 = line_6(11, 22)  # 创建另外一个新的闭包，让line_6_1指向
line_6_2(0)
line_6_2(1)
line_6_2(2)

# 这个空间里面有k , b的值（函数外面的变量），还有一个被定义的函数，外面有一个变量指向里面的函数，导致空间里面的变量是不会被释放的。
```

2. 思考：函数、匿名函数、闭包、对象 当做实参时 有什么区别?

```
def xxxx(temp):
    passs

xxxx(函数名)
xxxx(匿名函数名)
xxxx(闭包)
xxxx(实例对象)

# 1. 匿名函数能够完成基本的简单功能，，，传递是这个函数的引用 只有功能
# 2. 普通函数能够完成较为复杂的功能，，，传递是这个函数的引用 只有功能
# 3. 闭包能够完成较为复杂的功能，，，传递是这个闭包中的函数以及数据，因此传递是功能+数据
# 4. 对象能够完成最为复杂的功能，，，传递是很多数据+很多功能，因此传递是功能+数据
# 注意空间内存的那个示例图，结合着看。
```

3.内部函数修改外部函数的变量nonlocal

```
# coding=utf-8
def line_6(k, b):
    # all_num = 0  # 来记录执行的次数
    all_num = [0]
    def create_y(x):
        # nonlocal all_num  # python3的方式；使用这个不是改变全局变量，而是函数里面的变量，所以就不能使用global，而使用nonlocal。
        all_num[0] += 1  # python2的方式    # 这个不是可变不可变的问题，而是指向的问题。
        print("这是第%d次执行,当x=%d时，y=%d" % (all_num[0] , x, k*x+b))
    return create_y

line_6_1 = line_6(1, 2)  # 创建一个新的闭包，让line_6_1指向
# print(id(line_6_1))
line_6_1(0)
line_6_1(1)
line_6_1(2)
# line_6_2 = line_6(11, 22)  # 创建另外一个新的闭包，让line_6_1指向
# print(id(line_6_2))
# line_6_2(0)
# line_6_2(1)
# line_6_2(2)
```