Python内嵌作用域与函数闭包

1. python中独特的嵌套函数

2. 嵌套作用域与闭包现象

3. nonlocal关键字与内嵌作用域变量修改

变量的LEGB索引机制：对一个变量，首先在本地（函数内）查找；之后查找嵌套函数的本地作用域，然后再是查找当前的全局作用域。

接下来介绍嵌套作用域，即E，它是嵌套函数的本地作用域。

什么是嵌套函数?

python有一个很有意思的地方，就是def函数可以嵌套在另一个def函数之中。调用外层函数时，运行到的内层def语句仅仅是完成对内层函数的定义，而不会去调用内层函数，除非在嵌套函数之后又显式的对其进行调用。

```
x = 99

def f1():
    x = 88
    def f2():
        print(x)
    f2()

f1()

88
```

可以看出，f1中的嵌套变量x覆盖了全局变量x=99，然后f2中的本地变量按照引用规则，就引用了x=88。

下面我们来说说嵌套作用域的一个特殊之处:

本地作用域在函数结束后就立即失效，而嵌套作用域在嵌套的函数返回后却仍然有效。

```
def f1():
    x = 88
    def f2():
        print(x)
```

```
    return f2

action = f1()
action()

88
```

这个例子非常重要，也很有意思，函数f1中定义了函数f2，f2引用了f1嵌套作用域内的变量x，并且f1将函数f2作为返回对象进行返回。最值得注意的是我们通过变量action获取了返回的f2，虽然此时f1函数已经退出结束了，但是f2仍然记住了f1嵌套作用域内的变量名x。

上面这种语言现象称之为闭包：一个能记住嵌套作用域变量值的函数，尽管作用域已经不存在。

这里有一个应用就是工厂函数，工厂函数定义了一个外部的函数，这个函数简单的生成并返回一个内嵌的函数，仅仅是返回却不调用，因此通过调用这个工厂函数，可以得到内嵌函数的一个引用，内嵌函数就是通过调用工厂函数时，运行内部的def语句而创建的。

```
def maker(n):
    k = 8
    def action(x):
        return x ** n + k
    return action

f = maker(2)
print(f)

<function maker.<locals>.action at 0x00000000021C51E0>
```

再看一个例子：

```
def maker(n):
    k = 8
    def action(x):
        return x ** n + k
    return action

f = maker(2)
print(f(4))

24
```

这里我们可以看出，内嵌的函数action记住了嵌套作用域内的两个嵌套变量，一个是变量k，一个是参数n，即使后面maker返回并退出。我们通过调用外部的函数maker，得到内嵌的函

数action的引用。这种函数嵌套的方法在后面要介绍的装饰器中会经常用到。这种嵌套作用域引用，就是python的函数能够保留状态信息的主要方法了。

这里接着说说另一个关键字nonlocal

本地函数通过global声明对全局变量进行引用修改，那么对应的，内嵌函数内部想对嵌套作用域中的变量进行修改，就要使用nonlocal进行声明。

```
def test(num):
    in_num = num
    def nested(label):
        nonlocal in_num
        in_num += 1
        print(label, in_num)
    return nested

F = test(0)
F('a')
F('b')
F('c')

a 1
b 2
c 3
```

这里我们可以看到几个点，我们在nested函数中通过nonlocal关键字引用了内嵌作用域中的变量in_num，那么我们就可以在nested函数中修改他，即使test函数已经退出调用，这个“记忆”依然有效。

再最后一个例子：

```
def test(num):
    in_num = num
    def nested(label):
        nonlocal in_num
        in_num += 1
        print(label, in_num)
    return nested

F = test(0)
F('a')
F('b')
F('c')
```

```
G = test(100)
G('mm')

a 1
b 2
c 3
mm 101
```

多次调用工厂函数返回的不同内嵌函数副本F和G，彼此间的内嵌变量in_num是彼此独立隔离的。

✳

作者：*酱油哥，清华程序猿、IT非主流*

专栏地址：https://zhuanlan.zhihu.com/c_147297848