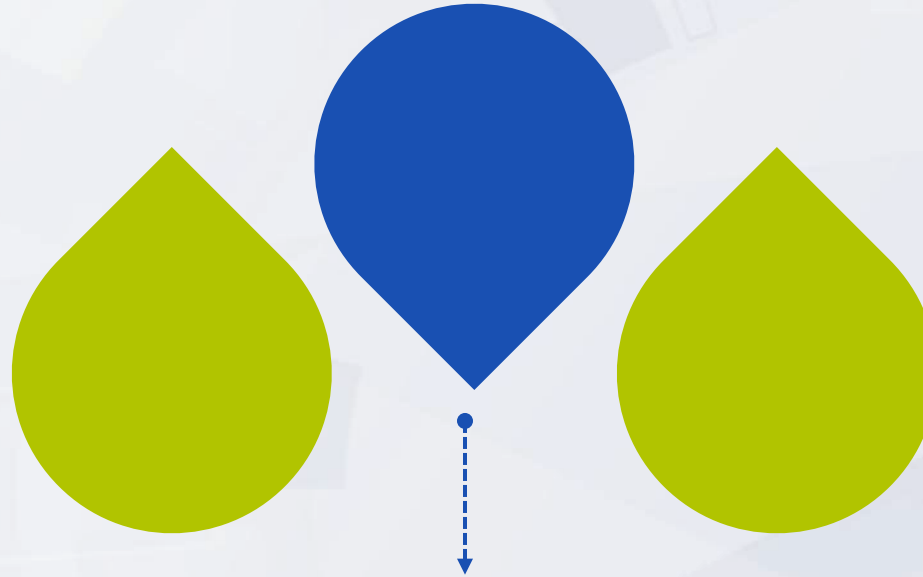


# PYTHON

## 编程基础

# 闭包

# 概述



如果内层函数使用了外层函数中定义的内部变量，并且外层函数的返回值是内层函数的引用，就构成了闭包。

定义在外层函数中但由内层函数使用的变量被称为自由变量。

一般情况下，如果一个函数结束，那么该函数中定义的内部变量就都会释放。

## 概述

然而



闭包是一种特殊情况，外层函数在结束时会发现其定义的内部变量将来会在内层函数中使用，此时外层函数就会把这些自由变量绑定到内层函数。

因此



所谓闭包，实际上就是将内层函数的代码以及自由变量（外层函数定义、但会由内层函数使用）打包在一起。

## 例：闭包示例

```
1. def outer(x): #定义函数outer
2.     y=10 #定义局部变量y并赋为10
3.     def inner(z): #在outer函数中定义
4.         #嵌套函数inner
5.         nonlocal x,y #nonlocal声明
6.         return x+y+z #返回x+y+z的结果
7.     return inner #返回嵌套函数inner的引用
8. f=outer(5) #将返回的inner函数赋给f
9. g=outer(50) #将返回的inner函数赋给g
```

```
10. print('f(20)的值为: ', f(20))
11. print('g(20)的值为: ', g(20))
12. print('f(30)的值为: ', f(30))
13. print('g(30)的值为: ', g(30))
```

```
f(20)的值为: 35
g(20)的值为: 80
f(30)的值为: 45
g(30)的值为: 90
```

## 例：闭包示例



### 提示

闭包的主要作用在于可以封存函数执行的上下文环境。

例如，通过两次调用outer函数形成了两个闭包，这两个闭包具有相互独立的上下文环境（一个闭包中 $x=5$ 、 $y=10$ ，另一个闭包中 $x=50$ 、 $y=10$ ），且每个闭包可多次调用。