

Python闭包入门

=====

关于闭包

闭包实际上是一个`嵌套函数`,且外部函数的返回值是一个`函数`,内部函数需要使用到外部函数定义的`变量`,一个简单的闭包函数如下:

```
...
def band():
    """
    存钱罐
    """
    money = 0 # 初始余额为0
    def add(a:float):
        nonlocal money
        money += a
        return money
    return add
if __name__ == '__main__':
    add = band() # 获取add函数
    m = add(10) # 存入10元
    print(f"当前余额为:{m}")
    m = add(-5.21)
    print(f"当前余额为:{m}")
...
```

这里通过一个简单的存钱罐实例来创建了一个闭包函数,其中闭包函数相较于普通函数而言,优势在于,可以`持久化保存变量`。

关于持久化保存变量

首先要明白`作用域`这个概念,作用域分为`全局作用域`和`局部作用域`两种,全局作用域指的是全局变量的作用域,一般在外部创建,在程序的整个生命周期内都可以使用,且可以在任意地方使用;局部作用域是指局部变量的作用域,在内部创建,运行时创建,结束时销毁。

```

...
MAX_NUM = 10 # 这里的MAX_NUM就是全局变量,全局可用
def count(a:int):
    num = 0 # 这里的num就是局部变量,只能在count函数内部使用,且每次运行count函数时创建,count函数结束时销毁
    if a > MAX_NUM:
        num = a # 修改局部变量,仅在当前有效,第二次运行函数时,变量的值仍然是0
    c = a - MAX_NUM
    c = a + num
    print(f"当前num值为:{num}")
    return c
c = count(100)
print(c)
d = count(5)
print(d)
...

```

上面函数的打印结果如下

```

...
当前num值为:100
200
当前num值为:0
5
...

```

从这里我们就可以看出来,函数创建的局部变量,仅在函数当前运行时有效,函数返回后,局部变量将会被销毁,如果有类似存钱罐这样的需求,也就是持久的保存一个变量,将这个变量写在函数中明显是不行的。
那么使用全局变量呢?

使用全局变量实现持久化变量

```

...
Money = 0
def add(a:float):
    global Money

```

```

    Money += a
    return Money

add(50)
print(f"当前余额为:{Money}")
add(-30)
print(f"当前余额为:{Money}")
Money = 15
add(10)
print(f"当前余额为:{Money}")

...

```

上面的代码打印结果如下

```

> 当前余额为:50
>
> 当前余额为:20
>
> 当前余额为:25

```

你会发现前面两次的结果是在我们预期中的,但是第三次明显就不是预期中的内容了,这是因为全局变量的`全局可用`导致的异常,因为全局可用,所以我们可以任意地方修改这个变量,这也就容易引发结果的混乱。而闭包函数中的外部变量也是一个局部变量,仅在闭包函数内部可用,外部是无法修改它的,这样就保证了变量的安全性。那么当闭包函数中的变量和全局变量的变量名一致时,是否会引发异常呢?

当闭包函数中的变量和全局变量一致时

```

...

num = 100
def ADD():
    num = 10

    def add(a:float):
        return a + num
    return add

A = ADD()
c = A(5)

```

```
print(c)
d = A(10)
print(d)
```

...

上面代码的执行结果为

> 15

>

> 20

可以看到,当闭包函数中的变量名称和全局变量的名称一致时,内部函数使用的是离它自己最近的变量,而不是全局变量。

这也就涉及到了函数的变量的取值过程,如果我们使用一个变量时,变量没有被定义,那么Python将自动`从内到外`的去寻找这个变量,也就是全局变量是最后的取值,如果查询到全局变量,且全局变量中也没有这个变量的话,Python将直接产生异常。

闭包总结

通过上面的内容,可以发现闭包的好处主要有两个方面

> 变量可以持久化保存,并不会随着函数的返回而销毁,因为外部函数的返回值是内部函数,所以只要外部函数没有被del掉,变量就可以实现持久化存储(外部函数的变量)

>

> 相较于全局变量的持久化存储,闭包函数中的变量更有安全性,因为不能被随意更改

原文链接: <https://juejin.cn/post/7375030479496331274>