Python程序设计

第六讲 函数与模块化 函数嵌套定义的应用



张华 WHU

可调用对象

■ 有哪些可调用对象

- * 函数属于Python可调用对象之一,由于构造方法的存在, 类也是可调用的。
- *像list()、tuple()、dict()、set()这样的工厂函数实际上都是调用了类的构造方法。
- * 另外,任何包含__call__()方法的类的对象也是可调用的。



可调用对象

■ 定义可调用对象的方法1

*下面的代码使用函数的嵌套定义实现了可调用对象的定义:

```
def linear(a, b):
    def result(x):
        return a * x + b
    return result #返回可被调用的函数
```

```
>>> f=linear(3, 5) #f是一个计算 3x+5 的可调用对象
>>> f(10)
35
```

可调用对象

■ 定义可调用对象的方法2

*下面的代码演示了可调用对象类的定义:

```
class linear:
    def __init__(self, a, b):
        self.a, self.b = a, b

def __call__(self, x): #这里是关键
    return self.a * x + self.b
```

```
>>> f=linear(3, 5) #f是一个计算 3x+5 的可调用对象 >>> f(10) 35
```

■带装饰器的函数

- * 带装饰器的函数定义形式如下
 - @装饰器1
 - @装饰器2
 - **@...**

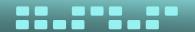
def 函数名(参数表): 函数体

*例如

```
@decor
def func():
    pass
```

》即

func=decor(func)



■ 装饰器 (decorator)

- **装饰器是函数嵌套定义的另一个重要应用。
- ** 装饰器本质上也是一个函数,只不过这个函数接收其他函数作为参数,并对其进行一定的改造之后使用新函数替换原来的函数。
- * Python面向对象程序设计中的静态方法、类方法、属性等也都是通过内置装饰器实现的。
- * 也可以需要自定义装饰器。



■ 自定义装饰器举例1

```
#定义装饰器
def before(func):
   def wrapper(*args, **kwargs):
       print('Before function called.')
       return func(*args, **kwargs)
   return wrapper
                                       #定义装饰器
def after(func):
   def wrapper(*args, **kwargs):
       result = func(*args, **kwargs)
       print('After function called.')
       return result
   return wrapper
@before
@after
                  #同时使用两个装饰器改造函数,相当于 before(after(test))
def test():
   print(3)
#调用被装饰的函数
                     Before function called
test()
                     After function called.
```

■ 自定义装饰器举例2

```
def makebold(fn):
    def wrapper(*s):
        return '<b>'+fn(*s)+'</b>'
    return wrapper
def makeitalic(fn):
    def wrapper(*s):
        return '<i>'+fn(*s)+'</i>'
    return wrapper
@makebold
@makeitalic
def htmltag(text):
    return text
print(htmltag('Decorator'))
```

<i>Decorator</i>

