
图解
Python
可能是迄今为止
最易懂的Python视频教程

```
"""
    对于某个函数，如果我们希望在不改变该函数代码的前提下，为该函数增加额外的功能，那么就可以使用装饰器来装饰该函数。
    装饰器是一个函数，装饰器接收一个函数作为参数（传入的实参是被装饰的函数），装饰器的内部嵌套定义另一个函数，内函数中会引用装饰器的参数，并且装饰器的返回值是内函数。这样，就构成了一个闭包。
为了让内函数接收任意类型的参数，将内函数的形参定义为(*args, **kwargs)。在函数中，首先完成为被装饰函数添加的新功能，然后调用被装饰的函数。
    把装饰器应用到被装饰函数的语法为：在被装饰函数的前面添加"@装饰器的函数名"。在被装饰函数add的前面添加@log后，相当于执行了语句：add = log(add)，首先，被装饰的函数add会作为实参传递给装饰器log，然后，返回装饰器的内函数wrapper，最后，将内函数wrapper赋值给名为add（被装饰函数的函数名）的变量，这样，再调用被装饰的函数add时，其实调用的是装饰器的内函数wrapper。
"""
def log(func):
    def wrapper(*args, **kwargs):
        print("函数%s被调用了" % func.__name__)
        return func(*args, **kwargs)
    return wrapper

@log
def add(sum1, sum2):
    print(sum1, sum2)
    return sum1 + sum2

print(add(1, 2))

print(add.__name__) # wrapper
```