
图解
Python
可能是迄今为止
最易懂的Python视频教程

```
"""
    如果希望被装饰函数的特殊属性__name__的值为其函数名，而不是装饰器的内函数的函数名，
可以在装饰器的内函数前面添加另外一个装饰器：@functools.wraps(装饰器的参数名)，
其中functools.wraps指的是标准库模块functools中的函数wraps。
"""
import functools

def log2(func):
    @functools.wraps(func)
    def wrapper(*args, **kwargs):
        print("函数%s被调用了" % func.__name__)
        return func(*args, **kwargs)
    return wrapper

@log2
def add2(sum1, sum2):
    print(sum1, sum2)
    return sum1 + sum2

print(add2(1, 2))

print(add2.__name__) # add2
```

```
"""
    把装饰器应用到被装饰函数时，还可以传递额外的参数。此时，需要编写一个3层嵌套的装饰器。
    对于@log3('6月', '18日')，相当于执行语句：add3 = log3('6月', '18日')(add3)。
"""
def log3(month, day):
    def decorator(func):
        @functools.wraps(func)
        def wrapper(*args, **kwargs):
            print("%s%s, 函数%s被调用了" % (month, day, func.__name__))
            return func(*args, **kwargs)
        return wrapper
    return decorator

@log3('6月', '18日')
def add3(sum1, sum2):
    print(sum1, sum2)
    return sum1 + sum2

print(add3(1, 2))
```