# 类视图中的装饰器：

decorator

装饰器本身就是一个函数, 但其参数和返回值都是函数

相当于对传入的函数做一些装饰后,传出一个结果函数

装饰器使用是使用@符号, 在被装饰的函数声明前

例如@app.route('/')就是一个装饰器

装饰器传参: 传未知参数和关键字参数 *args, **kwargs

这样装饰器函数所装饰的函数就可以穿任意参数

经过装饰器装饰的函数,函数名会被修改(func.__name__),可能会带来一些问题

解决的办法:

```
from functools import wraps
# 自己定义的检查登陆的装饰器
#########################################################
def login_required(func):
    @wraps(func)
    def wrapper(*args, **kwargs):
        if(session.get('userId')):
            return func(*args, **kwargs)
        else:
            return redirect(url_for('login'))
    return wrapper
#########################################################
```

引入的wraps本身也是一个装饰器, 用在自己定义的装饰器函数中, 装饰要返回的函数, 可以保证被装饰的函数的函数名不变

1. 如果使用的是函数视图，那么自己定义的装饰器必须放在app.route下面。否则这个装饰器就起不到任何作用。

2. 类视图的装饰器，需要重写类视图的一个类属性decorators，这个类属性是一个列表或者元组都可以，里面装的就是所有的装饰器。