python装饰器

简介

- 本质上是一个python函数
- 让其他函数在不需要任何代码变动的前提下增加功能
- （在短裤后面加一个长裤）
- 装饰器的返回值是一个函数对象

应用场景

- 插入日志
- 性能检测
- 事务处理
- 缓存
- 权限校验

作用

- 抽离出大量与函数功能无关的雷同代码
- （为已经存在的对象添加额外的功能）

例子

- 日志功能

```
def use_logging(func):

    def wrapper(*args,**kwargs):
        logging.warn("%s is running"%func.__name__)
        return func(*args)
    return wrapper

@use_logging
def foo():
    print("foo")

@use_logging
def bar():
    print("bar")
```

- 带参数装饰器

```
def use_logging(level):
    def decorator(func):
        def wrapper(*arg,**kwargs):
            if level == "warn":
                logging.warn("%s is runing"%func.__name__)
             return wrapper
        return decorator

@use_logging(level="warn")
def foo(name='foo')
    print("%s"%name)

foo()
```

- 类装饰器

```
class Foo(object):
    def __init__(swlf,func):
        self.__func=func
    def __call__(self):
        print("decorator run")
        self._func()
        print("decorator end")

@Foo
def bar():
    print("bar")

bar()
```

- 内置装饰器
  - @staticmathod、@classmethod、@property

```
@a
@b
@c
def f ():

#顺序
f = a(b(c(f)))
```