Python装饰器

Python装饰器的本质

- 装饰器(decorator)的本质:

函数闭包(function closure)的语法糖(Syntactic sugar)

1. 什么是函数闭包(function closure)
2. 什么是语法糖(Syntactic sugar)
3. 什么是装饰器(decorator)

1. 什么是函数闭包(function closure)

- 函数式语言(函数是一等公民,可作为变量使用)中的术语.

- 函数闭包: 一个函数,其参数和返回值都是函数.
  - 用于增强函数功能
  - 面向切面编程(AOP)

2. 什么是语法糖(Syntactic sugar)

- 指计算机语言中添加的某种语法，这种语法对语言的功能没有影响，但是更方便程序员使用。
  - 语法糖没有增加新功能,只是一种更方便的写法.
  - 语法糖可以完全等价地转换为原本非语法糖的代码.

- 装饰器在第一次调用被装饰函数时进行增强

```
@count_time_wrapper
def print_odds():...


print_odds()
```

完全等价

```
def print_odds():...


print_odds = count_time_wrapper(print_odds)
print_odds()
```

装饰器`@闭包函数名`

- 装饰器在第一次调用被装饰函数时进行增强
  - 增强时机?在第一次调用之前
  - 增强次数?只增强一次

保留函数参数和返回值的函数闭包

- 之前所写的函数闭包,在增强主要功能函数时,没能保留原主要功能函数的参数列表和返回值.

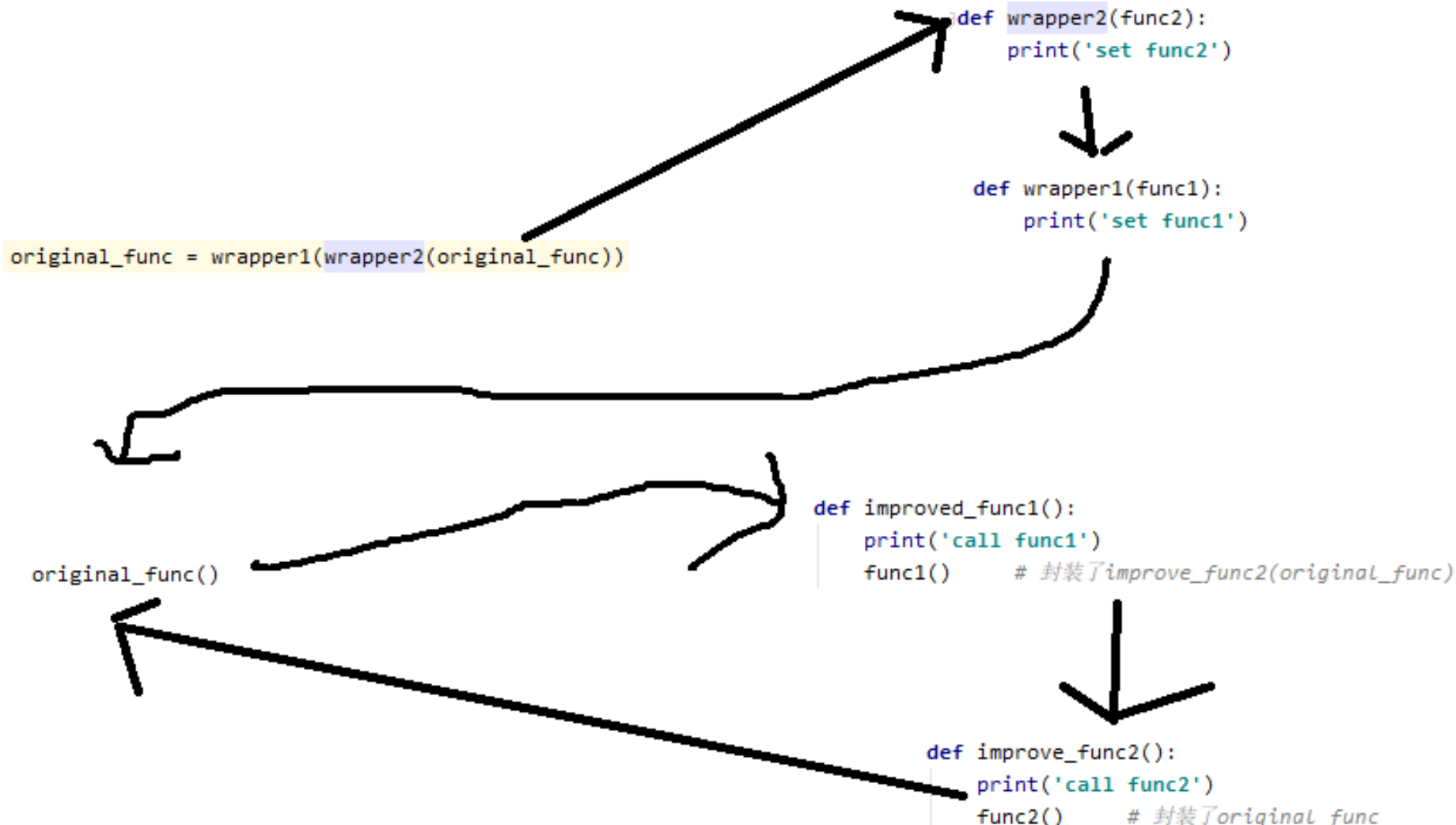
- 一个保留了参数列表和返回值的函数闭包写法:

```
def general_wrapper(func):

    def improved_func(*args, **kwargs):     # 接收函数参数
        # 增强功能
        ret = func(*args, **kwargs)         # 传入参数并记录返回值
        # 增强功能
        return ret                          # 返回未增强函数的返回值

    return improved_func
```

多个装饰器的执行顺序


def wrapper2(func2):
print('set func2')
def wrapper1(func1):
print('set func1')
original_func = wrapper1(wrapper2(original_func))
def improved_func1():
print('call func1')
func1() # 封装了improve_func2(original_func)
original_func()
def improve_func2():
print('call func2')
func2() # 封装了original_func

思考题: 如何构造一个带参数的装饰器

```
@log_wrapper(info='a try')
def count_odds():...

if __name__ == '__main__':
    count_odds()
```

```
Logging: func:count_odds runs from 2020-05-12 11:40:34 to 2020-05-12 11:40:34, info[a try]
```

提示: 多层闭包(外层闭包返回装饰器),
见https://python3-cookbook.readthedocs.io/zh_CN/latest/c09/p04_define_decorator_that_takes_arguments.html