一 课程回顾

```
"""
1. 对于序列 [-1,3,-5,-4] 的每个元素求绝对值
2. 用map来处理字符串列表,把列表中所有的元素后面都加上_xc,比方lulu_xc
3. 将一串数字‘1 3 5 7 8’转化为整型，并以列表形式输出
"""

list1 = [-1,3,-5,-4]
print(list(map(abs,list1)))

list1 = ['lili','tutu','zuozuo']

def fn(x):
    return x+'_xc'

print(list(map(fn,list1)))

print(list(map(lambda x:x+'_xc',list1)))

f = '1 3 5 7 8'.split(' ')
print(f)
print(list(map(int,f)))
```

二 闭包与装饰器

def 函数名 定义函数

函数名() 调用函数(使用函数)

```
x = 10
def f():
    a = 5
    def g():
        y = 7
        print(b)
```

```
def f():
    x = 10  # 变量  在内存里面 看到这个变量 认识他
    def g():   # 经过函数g
        print(x)
    return g   # fn()  最终能接收到的  return g
f()()  # g()   f() () = g ()
```

```
def f():
    x = 10  # 变量  在内存里面 看到这个变量 认识他
    def g():   # 经过函数g
        print(x)
    return g   # fn()  最终能接收到的  return g
h = f() # g()   f() () = g ()
h()
```

2.1 闭包

定义：如果在一个内部函数里面，对在外作用域(但不是全局作用域)的变量进行引用，那么内部函数就被认为是闭包

闭包 = 内部函数 + 定义函数时的环境 （定义的变量）

```
def f(x):
    def g():
        print(x)
    return g
h = f(1)
h()
```

不一定需要return才能有闭包 只要满足闭包的条件 只要内层函数引用了外层函数变量 就会产生闭包

```
def fn(x):
    def fn1(y):
        return x + y
    return fn1
```

这也是一个闭包

```
def fn():
    a = 10
    def fn1():
        print(a)
    fn1()
fn()
```

这个就不是闭包

2.2 装饰器

```
def foo():
    print('foo...')
foo()


def foo():
    start = time.time()  # 家里 开始时间 从0秒开始计算
    print('foo...')   #  从家里到学校的过程
    time.sleep(1)  # sellp()  可以休眠程序  秒数
```

```
    end = time.time()  # 从家里到学校的时间
    print(f'运行了{end - start}秒')
foo()

# print(time.time())  # 当前时间 从1970年1月1日 00时 00分 00秒 到现在的总秒数

开放封闭原则
```

```
def foo():
    print('foo...')   #  从家里到学校的过程
    time.sleep(1)  # sellp()  可以休眠程序  秒数



def show_time(func):  # 功能是展示时间
    start = time.time()  # 家里 开始时间 从0秒开始计算
    func()  # 调用  具体调用谁 由实参决定   func() = foo()
    end = time.time()  # 从家里到学校的时间
    print(f'运行了{end - start}秒')

show_time(foo)
```

装饰器，生成器，迭代器这些都是函数器，本质上就是一个函数

装饰字面意思就是添加一个新功能，为你之前的某个函数添加新的功能，也就是在不修改源代码的基础上给函数添加新的功能

```
import time


def foo():
    print('foo...')   #  从家里到学校的过程
    time.sleep(1)  # sellp()  可以休眠程序  秒数


def show_time(func):  # 功能是展示时间
    def inner():
        start = time.time()  # 家里 开始时间 从0秒开始计算
        func()  # 调用  具体调用谁 由实参决定   func() = foo()
        end = time.time()  # 从家里到学校的时间
        print(f'运行了{end - start}秒')
    return inner

foo = show_time(foo)  # show_time(foo) = inner
foo()
```

@ ==> 语法糖

```
import time

def show_time(func):  # 功能是展示时间  show_time 是一个装饰器函数
    def inner():  # inner 是一个闭包函数
        start = time.time()  # 家里 开始时间 从0秒开始计算
        func()  # 调用  具体调用谁 由实参决定   func() = foo()
```

```
        end = time.time()  # 从家里到学校的时间
        print(f'运行了{end - start}秒')
    return inner

@ show_time  # foo = show_time(foo)    a += c  a = a + c
def foo():
    print('foo...')   #  从家里到学校的过程
    time.sleep(1)  # sellp()  可以休眠程序  秒数
foo()

@ show_time
def bay():
    print('bay...')
    time.sleep(1)
foo()
```

2.2.1 原函数参数

求两位数的和

```
def show_time(func):  # 功能是展示时间  show_time 是一个装饰器函数
    def inner(a,b):  # inner 是一个闭包函数
        start = time.time()  # 家里 开始时间 从0秒开始计算
        func(a,b)  # 调用  具体调用谁 由实参决定   func() = foo()
        end = time.time()  # 从家里到学校的时间
        print(f'运行了{end - start}秒')
    return inner


@ show_time  # add = show_time(add)
def add(a,b):
    print(a+b)
    time.sleep(1)
add(1,2)   # add(1,2)  = inner()
```

```
def show_time(func):  # 功能是展示时间  show_time 是一个装饰器函数
    def inner(*args,**kwargs):  # inner 是一个闭包函数
        start = time.time()  # 家里 开始时间 从0秒开始计算
        func(*args,**kwargs)  # 调用  具体调用谁 由实参决定   func() = foo()
        end = time.time()  # 从家里到学校的时间
        print(f'运行了{end - start}秒')
    return inner


@ show_time  # add = show_time(add)
def add(*args,**kwargs):
    sum = 0
    for i in args:
        sum += i
    print(sum)
    time.sleep(1)
add(1,2,3,4,5,6,7,8,9)   # add(1,2)  = inner()
```

总结：
闭包：就是能够读取其他函数内部变量的函数
作用；保存外部函数的变量，不会随着外部函数调用完而销毁
形成条件：

1.函数嵌套
2.内部函数必须使用了外部函数的变量或者参数
3.外部函数返回内部函数，这个使用了外部函数变量的内部函数称为闭包

装饰器：实质上也是一个闭包函数，也就是说他也是一个函数嵌套
作用：在不改变原函数的情况下，对已有函数进行额外的功能扩展
形成条件：

1.不修改已有的函数的源代码
2.不修改已有函数的调用方式
3.给已有函数增加额外的功能

注意：装饰器与闭包的区别，装饰器是一个闭包函数，但是装饰器这个闭包函数，他的参数有且只有一个并且是函数类型的话，他才是装饰器，否则他就是闭包函数