图灵大海老师

函数高级

核心重点理解

行！我理解了，求你别说了

匿名函数

核心

一行代码图省事

除了没有名字其他的都有

表达式

```
# 表达式
print((lambda x,y:x+y)(1,2))
```

```
# 函数
(lambda x,y:print(x+y))(1,2)
```

匿名函数与内置函数结合使用

```
# max,min,sorted
salaries= {
    'xialuo':3000000,
    'xishi':30000,
    'dahai':3000
}
# 求最小值
print(min(salaries,key=lambda name:salaries[name]))
```

递归函数

两个明确的阶段

1. 回溯：一次次递归调用下去，说白了就一个重复的过程，
   但需要注意的是每一次重复问题的规模都应该有所减少，
   直到逼近一个最终的结果，即回溯阶段一定要有一个明确的结束条件

2. 递推：往回一层一层推算出结果

例子

```
def search(L):
    for n in L:
        # print(n)
        if type(n) is not list:
            print(n)# 1
        else:
            # [2,[3,[4,[5,[6,[7,[8,[9,]]]]]]]]
            search(n)
search(L)
```

```
# 递归是函数的定义里面嵌套函数的调用
# 递归与循环的区别，循环每一次都要判断，需要考虑多少次  ****
# 而递归只需要确定结束条件就行，按照规律进行重复调用，不需要考虑次数
```

闭包函数

```
# 闭包
# 闭指的是:该函数是一个内部函数
# 包指的是:指的是该内部的函数名字在外部被引用
def outer():# 没有调用outer(),但是创造了outer这个函数
    # 1 只检测函数体的语法（ 工厂合不合格），不执行函数体代码 （不使用工厂）
    print('外面的函数正在运行')
    def inner():
        print('里面的函数正在运行')
        # 3返回inner函数的内存地址  想象成inner工厂的一把钥匙，由outer的物流return
    return inner
#2 执行了outer这个函数体代码，inner定义了
inner=outer()# 4,得到了里面的inner钥匙，取名字inner  包子肉拿出来
# print(inner)
# # # inner工厂的钥匙     加开关()
# # 5 里面的inner钥匙加括号运行    inner这个函数
inner()
# 一起 不规范
# outer()()
```

五部曲

软件的维护应该遵循开放封闭原则

开放封闭原则指的是：

软件一旦上线运行后对修改源代码是封闭的，对扩展功能的是开放的

这就用到了装饰器

装饰器的实现必须遵循两大原则：

1、不修改被装饰对象的源代码(人的原来的性格，生活方式)

2、不修改被装饰对象的调用方式(人的原来的外貌，名字)

^

装饰器

形象理解

```
# 装饰器的实现必须遵循两大原则：
#         1、不修改被装饰对象的源代码(人的原来的性格，生活方式)
#         2、不修改被装饰对象的调用方式(人的原来的外貌，名字)
#     装饰器其实就在遵循1和2原则的前提下为被装饰对象添加新功能
# 比如男孩1给女朋友买衣服，项链或化妆品  ，  变的更加自信，原来的外貌，原来的性格没有发生改变 # 装饰器

# 比如男孩2带女朋友去整容   ，  变的更加自信(不一定)，  原来的外貌，原来的性格发生改变 # 不是装饰器
# 女朋友
```

代码分步骤理解

```
# 把一个整体综合的知识点拆解开来理解     能力的提升
name = '大海'
def run(name):
    print('==========')
    print('我是%s'%name)
    print('==========')
# print(run)
# # run(name)
#
# # 装饰器就是一个特殊的闭包函数
# # 1.定义了decorate，检测decorate语法，    new_func没有定义
def decorate(func):# func等下我们要传入的run
    # print(func) # run
    def new_func(name): #  run(name)  的name
        print('我是装饰函数前面的代码')
        func(name)# run(name)
        print('我是装饰函数后面面的代码')
    return new_func
#2.1 定义了new_func(name)函数 2.2 返回了new_func的内存地址 2.3 传入了一个run函数名作为decorate函数参数
run=decorate(run)
#

#
# # 魔术 并不是  魔法 真相是调用了new_func函数
#
#
# print(run)
run(name)
```