第八周问答作业

朱士杭 231300027

2024年04月19日

（一）什么是高阶函数（Higher-order function）

高阶函数从数学上的理解本质就是复合函数

但是在python里面高阶函数实际上指的是函数作为另一个函数的参数进行传递

由于python闭包功能的存在，比如有函数h(g(f(x1,x2)))执行的时候，先执行h()然后返回一个函数对象赋值给另一个函数变量名， 然后再调用g、f，最后传递参数x1与x2，这个时候在h的frame里面，传进去的g和f作为形参是被保留下来的了可以直接调用，进行函数复合运算

下面是使用高阶函数实现y=Ax+b运算的例子：

```
#一个通用的高阶函数框架如下
def higher_func(func_input):
    def wrapped(list_input):
        res=func_input(list_input)
        return res
    return wrapped
#接下来写y=Ax+b的功能
def linear_transformation(A,x):
    '''实现A*x操作 前提是A与x一定得是同维的'''
    y=[]
    for i in A:
        sum=0
        for j in range(len(i)):
            sum+=i[j]*x[j]
        y.append(sum)
    return y
def add_vector(x,b):
    '''实现向量相加的运算 前提是x和b一定是同维的'''
    return [x[i]+b[i] for i in range(len(x))]
def x_to_y(func1):
    def wrapped_1(func2):
        def wrapped_2(list_input,A,b):
            res1=func1(A,list_input)
            res2=func2(res1,b)
            return res2
        return wrapped_2
    return wrapped_1
A=[[1,0,0],
   [0,2,0],
   [0,0,1]]
b=[1,2,3]
```

```
x=[2,3,4]
y=x_to_y(linear_transformation)(add_vector)(x,A,b)
print(y)
```

（二）医生工作状态

使用面向对象创建一个“医生类”，从而实现切换某位医生的忙碌状态功能

```
class GetSwitcher:
    def __init__(self,name):
        self.name=name
        self.state="空闲"
    def switch(self):
        if self.state=="空闲":
            self.state="忙碌"
        else:
            self.state="空闲"
        print(f"{self.name}医生当前状态为：{self.state}")
sw1 = GetSwitcher("Sun")  #为孙医生建立一个sw，初始为空闲
sw1.switch() #切换状态，原先为空闲则改为忙碌，反之亦然；输出当前状态
sw1.switch() #再次切换状态，原先为空闲则改为忙碌，反之亦然；输出当前状态
sw2 = GetSwitcher("Zhang") #为张医生建立一个sw ...
sw2.switch()
```

应该如何处理提供设置和查询某位医生的状态的功能—— 只需要在医生类中添加两个成员函数分别实现设置与查询功能，并在主函数里面调用这两个成员函数即可

在上述代码中的class GetSwitcher中续上如下代码即可：

```
    def set_state(self,state_input):
        self.state=state_input
    def query_state(self):
        print(f"{self.name}医生当前状态为：{self.state}")
```

（三）医生工作状态修改版

利用函数闭包来管理医生的工作状态

```
def createDoctorStatusTracker(name):
    '''''''
    state="空闲"
    def wrapped(str_input):
        '''''''
        nonlocal state
        if str_input=="state":
            print(f"{name}医生当前的状态为：{state}")
        elif str_input=="switch":
            if state=="空闲":
                state="忙碌"
```

```
            else:
                state="空闲"
            print(f"{name}医生当前的状态为：{state}")
    return wrapped
dr_sun = createDoctorStatusTracker("Sun")  # 为孙医生创建一个状态跟踪器，初始状态为“空闲”
dr_sun("state") # 查询当前医生工作状态
dr_sun("switch") # 切换状态，如果原先为空闲则改为忙碌，反之亦然，并输出新的状态print(dr_sun()) # 再次查询当前状态，输出
dr_zhang = createDoctorStatusTracker("Zhang") # 为张医生创建一个状态跟踪器 ...
dr_zhang("switch") # 切换张医生的状态并输出...
```

（四）比较二、三中两种方式的异同

异

（二）采用了面向对象编程的方式，通过对数据进行抽象封装整理，一名医生对应一个具体的对象创建一个单独的frame， 每个医生之间的数据不会产生冲突

（三）采用了函数式编程的思想，通过python中的闭包功能，为每一个医生创建一个大的函数frame，然后通过传参的方式进行函数复合， 比如dr_sun = createDoctorStatusTracker("Sun")与dr_sun("state")这两条语句本质上就是在执行createDoctorStatusTracker("Sun")("state") 这一条代码。为每个医生创建的函数之间也是相互独立的，这样子数据间也不会产生冲突

同

无论是面向对象编程还是函数式编程，其最终目的都是为了实现抽象与封装，只不过一个是对数据一个是对过程，各有其优点。 而且它们都巧妙地运用到了python中的frame，都为每一个医生创建一个单独的frame帧，这样方便对数据的操作与管理，更加结构化。