一，闭包：（python函数是支持嵌套的）（实际引用的是内部函数占用的内存空间）

满足下列三个条件的是闭包：1）存在于嵌套关系的函数中

2）嵌套的内部函数引用了外部函数的变量

3）嵌套的外部函数会将内部函数名作为返回值返回

从生命周期的角度来说，在外部函数执行结束后，变量就被释放了，当外部函数执行外毕后，会在执行内部函数，由于内部函数中使用了外部函数中的变量，所有程序会出现异常，然而，在此函数中，程序依然正确的执行，是因为函数的闭包会记得外层函数的作用域，在内部函数（闭包）中引用了外部函数的变量，所有程序不会释放这个变量

二，装饰器：

1，装饰器实质上是一个python函数，它可以在不改动其他函数的前提下，对函数的功能进行扩充。

装饰器主要用于如下场景：1）引入日志 ；2）函数执行的时间统计； 3）执行函数前预备处理；4）执行函数后的清理功能；5）权限校验；6）缓存

在python中，装饰器是以@开头的

如：

```
def wrap(fun):
    print("正在校验")
    def inner():
        print("正在校验权限")
        fun()
    return inner
@wrap
def test():
    print("test")
test()
```

程序的执行过程：

1）当程序执行test（）时，发现函数test（）上面有装饰器@wrap，所以会先执行@wrap，@wrap等价于test=wrap（test），它可以拆分为俩步，①执行wrap（test）将函数名test作为参数传递给wrap，②将wrap（test）的返回值赋给test，此时，test指向inner函数，2）调用test（）指向的函数

2，注意：多个修饰器可以应用在一个函数上，他们的调用顺序是自下而上的

3，装饰器对有参数函数进行修饰：如果无法确定函数的参数个数以及参数类型，我们可以使用不定长参数来传递，当调用不同参数的函数，发现装饰器适用于不同参数的函数

```
def wrap1(func):
    def inner(*args,**kwargs):
        print("开始验证权限")
        func(*args,**kwargs)
    return inner
@wrap1
def test1(*args,**kwargs0):
    print("----test-----")
test1()
test1(1,2,3)
test1(a=1,b=2,c=3)
```

4，装饰器对带有返回值的函数进行装饰。需要将return语句调用后的结果返回

```
def func2(func):
    def fun():
        return func()
    return fun()
def test4():
    return "anqili"
result = test4()
print(result)
```

5，带有参数的装饰器：如果给装饰器添加参数，那么需要增加一层封装，先传递参数，然后在传递函数名

```
def funca(args):
    def func(funcname):
        def funcin():
            print("记录日志%s"%args)
            funcname()
```

```
        return funcin()
    return func
@funca("hello")
def test11():
    print("----test---")
test11()
```

三，常见的python内置函数：内置函数，就是在python中被自动加载的函数，任何时候都可以使用

1，

map函数：map函数会根据提供的函数对指定的序列做映射

map()是 Python 内置的高阶函数，它接收一个函数 f 和一个 list，并通过把函数 f 依次作用在 list 的每个元素上，得到一个新的 list 并返回。

map函数的定义如下：map（function，iterable，....）function表示一个函数名，iterable可以是序列、支持迭代的容器或迭代器，当调用map函数时，iterable中的每一个元素都会调用function函数，所有元素调用function函数返回的结果会保存到一个迭代器对象中，在python2.x中，map函数的返回值是列表list类型。

1）当传入的函数需要俩个参数的时候，则需要传递俩个列表

2）在python3.x之前，如果调用map函数时传入的function参数为none，那么相当于合并参数为元

组。

3）如果俩个序列的元素个数不一致，那么元素少的序列会以None补齐，

4）在python3.x以后，当map传入的函数是None时，就等同于zip函数（用于将可迭代的对象作为参数，将对象中对应的元素打包成一个个元组，然后返回由这些元组组成的列表）的功能，并且已经被zp函数取代了，另外，map函数无法处理俩个序列长度不一致，对应位置操作数类型不一致的情况，他们都会报类型错误

2，

filter函数：会对指定的序列进行过滤

1）filter函数的定义

filter（function，iterable）

其中function参数可以是函数名或者是None，iterable可以是序列，支持迭代的容器、迭代器，返回值为迭代器对象（在python2.x中，map函数的返回值是列表），且function只能接受一个参数，而且该函数的返回值为布尔值

2）filter函数的作用是以参数迭代器中的每一个元素分别调用function函数，最后返回的迭代器包含调用结果为true的元素

3，

reduce函数

1）reduce函数会对参数迭代器中元素进行累积

reduce函数的定义：reduce（function，iterable[,initializer])

在上述的定义中，function是一个带有俩个参数的函数，iterable是一个迭代器对象，initializer是固定的初始值，reduce函数会依次从迭代器对象中取出每个元素，和上一次function的结果作为参数再次调用function函数，

2）在python3.x中，reduce函数被放置在functools模块中，使用的时候需要先引入

3）reduce传入的是带有俩个参数的函数，该函数不能为none

注意：

1：装饰器的作用是为了切面编程(AOP)

2，装饰器是一个包裹函数

3，装饰器可以有多个参数，，相比无参数的装饰器，带参数的装饰器只是用来"加强修饰"的，如果希望装饰器可以根

据参数的不同，对不同的函数进行修饰，那么带参数的装饰器是一个很不错的选择

4，通过在函数定义的前面加上@符号和装饰器的名字，使得装饰器函数生效

5，如果装饰器支持参数，其必须在嵌套一层函数

6，容器：列表，元组，字典，集合

python中的基本顺序存储结构的是列表和元组，在操作的复杂度上和数组完全相同，其中列表是可变数据类型，元组是不可变数据类型

字符串是迭代器