1、递归函数

函数直接或者间接调用本身

注意事项：

a递归一定要控制层数，当符合某一个条件的时候要终止递归

b 集合所有的递归问题都可以用while来代替

优点：复杂的问题简单化，思路更清晰，代码更简洁

缺点：受环境影响比较大，当递归深度比较大时，可能会得到不可预测的结果

练习：用递归解决

练习1、1+2+3+…+100的和值

练习2、求n的阶乘

2、闭包

将内嵌函数(函数中定义的函数)的语句和环境打包在一起，得到一个对象称为闭包。

闭包必须满足三个条件

a 必须有一个内嵌函数

b 内嵌函数必须引用外部函数的局部变量

c 外部函数返回的内容必须是内嵌的引用

闭包详解：一般情况下，在我们的认知中，函数调用结束后，函数内部的所有的东西都会被释放，但是闭包是一种特殊情况，如果函数在调用结束后，发现自己有局部变量将来可能在内部函数中使用到，就会把这个局部变量绑定给内部函数，然后自己再结束。

闭包中修改外部函数的局部变量：

内嵌函数可以随意访问外部函数的局部变量，但是不能修改(重新赋值也仅仅是定义了一个名字相同的变量而已)，在一般的函数中如果修改全局变量，可以使用global，告诉编译器此处不是重新定义的变量，而是全局变量，或者这个变量是可变类型(list,set,dict),在闭包的内嵌函数中，可以使用nonlocal表示这个变量不是内嵌函数的变量，而是需要往上一层环境去查找。

还有一点需要注意，在闭包的使用过程中，一旦外部函数被调用且返回了内部函数的引用，虽然每次调用内部函数，是开启函数执行，然后消亡，但是闭包只有一份，每次开启的都是同一份闭包。

好处：可以访问函数的局部变量

让这些变量始终保存在内存中，不随外部函数结束而结束

进一步解释：

全局变量：优点：可重用， 缺点：易被污染

局部变量：优点：局部访问，不会被污染

缺点：不可重用

闭包：既在全局使用局部变量，又保证不被污染

练习：type(x)返回变量/对象的类型

例如 type([1,2,3]) is list 返回True

装饰器：函数装饰器是指装饰一个函数，传入一个函数，返回的一个函数，函数装饰器最终作用到一个函数上

装饰器语法：def 装饰器名(参数)

语句块

return 函数对象

函数使用装饰器的语法：@装饰器名

函数的文档字符串

在函数中第一个没有被赋值给任何变量的字符串称为文档字符串，可以理解为帮助文档

函数的\_\_doc\_\_属性

绑定了函数的文档字符串

函数的\_\_name\_\_属性

返回一个字符串，绑定了函数的名称。

一个标准的函数

[函数装饰器1]

[函数装饰器2]

… …

def 函数名(位置形参，\*元组形参，命名关键字形参，\*\*字典形参)

“文档字符串“

语句块