函数：用于步骤封装(黑盒)或对重复代码提取(整洁)

函数

位置参数：

def NAME(key1):

STATEMENT

默认参数：

def NAME(key1=value):

STATEMENT

可变参数：

def NAME(\*args):

STATEMENT

\*args代表将参数作为tuple传递给函数

关键字参数：

def NAME(\*\*keywords):

STATEMENT

\*\*keywords代表将参数作为dictionary传递给函数

命名关键字参数：

def NAME(key1,\*,key2)

STATEMENT

key2为命名关键字参数，与位置参数需要\*分隔(或者之间有可变参数分隔)

嵌套函数：

def outer\_func():

print(“this is out func”)

def inner\_func():

print(“this is inner func”)

print(“we are back to out func”)

return inner\_func

装饰器(decolator)：

from functools import wraps  
#定义装饰器  
def define\_decolator(f):  
 #还原被装饰函数的\_\_name\_\_属性  
 @wraps(f)  
 def inner\_dec():  
 print("we are prepare decolation")  
 f()  
 print("we are over decolation")  
 return inner\_dec  
  
#使用装饰器  
@define\_decolator  
def out\_myname():  
 print("my name is lintom")

装饰器原理：使用装饰器时，相当于函数被装饰器用作形参

须知：

1.当需要在函数内部修改全局变量值时，需要使用global关键字

2.在嵌套函数中，内部函数需要修改外部函数变量值时，需要使用nonlocal关键字

3.函数形参是对实参的二次赋值，而非直接修改原来的值

4.函数名称可通过赋值的方式(不带参数限定)，来通过其他名称调用函数

5.函数接收另一个函数作为参数，则称之为高阶函数

6.在嵌套函数中，外部函数可以返回内部函数

7.函数默认返回None

8.函数形参默认值只评估一次(当默认值为列表时，多次改变列表，效果会持续叠加)，因为形参指向值(默认值)在函数定义时初始。修正方法为形参默认值为immutable

变量引用顺序：局部变量--嵌套变量--全局变量--外部built-in变量

常用高阶函数：

map(func,iterable)：以iterable内的item为参数，进行逐次调用func函数。返回iterator对象

functools.reduce(func,iterable)：将iterable的前两项进行func运算，并将再次作为第一项和第三项累积运算，如此循环。返回最后累积结果

filter(func,iterable)：以iterable内的item为参数，逐次调用func函数，如果返回的结果为False则过滤。返回iterator对象

递归函数：多次反复调用自身的函数。汉诺塔是递归最好的体现

递归会导致栈的叠加

生成器函数：使用yield关键字返回一个生成器

def gene(n)：

a,b = 0,1

for i in range(n):

a,b = b,a+b

yield a

\*\*next()由函数的yield处返回，下次调用接着返回处执行，并且生成器函数的return包含在StopIteration异常的value值中

\*\*可使用next()这个built-in函数返回下一个值的(惰性序列)，称为iterator，只有generator为迭代器，但可使用iter()这个built-in函数获取迭代器

\*\*可使用for循环的，称为iterable，list、tuple、dict、set、str和generator都是可迭代对象

lambda表达式

lambda a,b:a+b类似于如下函数：

def multi(a,b):

return a+b

Document String

def NAME():

‘‘‘

this is a document

it will be ignore

’’’

pass

用于生成在线注释文档

方法注释：

**>>> def** f(ham: str, eggs: str = 'eggs') -> str:

**...**  print("Annotations:", f.\_\_annotations\_\_)