# Python简介

Python是一种面向对象的编程语言，跨平台，可以在不同操作系统上运行。Python提供了非常全面的代码基础库，覆盖了网络、文件、GUI、数据库、文本等大量内容，被形象地称作“内置电池”。用Python开发，许多功能不必从零编写，直接使用现成的即可。除了内置的库外，Python还有大量的第三方库可供使用。所以Python程序看上去总是简单易懂，可读性较好，上手比较容易。Python的哲学就是简单优雅，尽量写容易看明白的代码，尽量写少的代码。

当然优点颇多的Python，也有其缺点。第一个缺点是运行速度，与C语言等编译型语言相比，Python是解释型语言，代码在执行时是一边编译一边执行所以运行时比较慢。第二个缺点是保密性，发布Python程序，实际上就等于发布Python的源代码，不像C语言、Java语言发布的时候是发布编译之后的机器码。

# Python安装

首先根据Windows版本（64位还是32位）从Python的官方网站下载Python对应的64位安装程序或32位安装程序。



然后，运行下载的exe安装包，记得需要勾选将Python添加到环境变量变量中，如果不勾选则需要自己手动在环境变量中添加Python。



安装成功后，打开命令提示符窗口，输入Python，如出现以下信息则表示Python安装成功且环境变量配置正确。



# Python解释器

在编写Python代码时，会得到一个以.py扩展名的文本文件，要运行代码需要Python解释器。Python解释器将源代码转化为字节码，如果Python进程在运行的环境中具有写入的权限，那么将会生成一个.pyc的文件，字节码保存在.pyc文件中（类似java代码文件编译成的class文件）。每次执行Python代码时，如果Python文件改动过那么会重新编译生成（通过检查Python源文件与字节码文件的时间戳）。如果不具备写入的权限，那么字节码将在内存中生成，执行完毕后会被丢弃。Python代码转化成字节码后会被发送到PVM（Python虚拟机）执行，PVM是Python能够跨平台的核心。

1. CPython

CPython是标准Python，也是其他Python编译器的参考实现。CPython由C编写，将Python源码编译成CPython字节码，由虚拟机解释执行。

（2）JPython

Jython在JVM上实现的Python，由Java编写。Jython将Python源码编译成JVM字节码，由JVM执行对应的字节码，因此能很好的与JVM集成。

# Python输出与输入

Python中用print()来打印输出，用print()可以打印字符串，字符串需要用引号括起来，引号可以是单引号或者双引号，例如print(‘hello world!’)，结果输出hello world!。print()也可以打印数值，或者计算结果，例如print(33.33)：结果输出33.33，print(3\*5)：结果输出15。print()函数可以打印多个输出，每个输出用逗号隔开，逗号输出为空格。例如print(‘3+5’,’=’,3+5)，结果输出3+5 = 8。

Python中用print()输出结果，同时提供了一个input()处理用户输入。例如：

length=float(input(‘请输入正方形的边长：’))

print(‘正方形的面积为：’,length\*length)

input()函数返回值是str型，如果输入的值作为数值型的参数，那么需要进行强制转换。执行这段代码首先会提示让输入一个正方形的边长，然后在输入正方形的边长之后，会输出正方形的面积。

# Python基础

## Python编写规范

Python的语法比较简单，写出来的代码就像下面的样子：

# Python编写规范

a = 100

if a >= 0:

print(a)

else:

print(-a)

# Python类（class）

calss TEST():

def test1(self):

print(‘方法1’)

def test2(self):

print(‘方法2’)

以#开头的语句是注释，可以是任意内容，解释器会忽略掉注释。Python中，对于类定义、函数定义、控制语句、异常处理语句等，行尾的冒号和下一行的缩进，表示下一个代码块的开始，而缩进的结束则表示此代码块的结束。Python中实现对代码的缩进，可以使用空格或者Tab键实现。但无论是手动敲空格，还是使用Tab键，通常情况下都是采用4个空格长度作为一个缩进量（默认情况下，一个Tab键就表示4个空格）。缩进让代码格式化，增加代码的可读性。Python对代码的缩进要求非常严格，同一个级别代码块的缩进量必须一样，否则解释器会报 SyntaxError 异常错误。此外Python程序是大小写敏感的，如果写错了大小写，程序也会报错。

## 数据类型和变量

Python能够直接处理的数据类型有以下几种：

1. 整数

Python可以处理任意大小的整数，当然包括负整数，在程序中的表示方法和数学上的写法一模一样，例如：1，100，-8080，0，等等。

1. 浮点数

浮点数也就是小数，浮点数可以用数学写法，如1.23，3.14，-9.01，等等。但是对于很大或很小的浮点数，就必须用科学计数法表示，把10用e替代，1.23x109就是1.23e9，或者12.3e8，0.000012可以写成1.2e-5，等等。

（3）字符串

字符串是以单引号或双引号括起来的任意文本，比如'abc'，"中国"等等。引号本身只是一种表示方式，不是字符串的一部分，因此字符串'abc'只有a，b，c这3个字符。如果引号本身也是一个字符，那必须用引号括起来，比如"I'm OK"包含的字符是I，'，m，空格，O，K这6个字符。

如果字符串内部既包含单引号又包含双引号，就需要用转义字符\来标识，比如'I\'m \"OK\"!' 表示的字符串内容是I'm "OK"!。转义字符\可以转义很多字符，\t表示制表符，字符\本身也要转义，\\表示的字符就是\。\n表示换行，为了可读性比较好，python一般用’’’...’’’表示多行内容。但如果本身字符串就带有\，例如文件的路径信息，这时候就不需要进行转义，Python中在字符串前加r表示字符串不转义，例如r”E:\n\t\test”表示的字符串内容为E:\n\t\test。

（4）布尔值

布尔值和布尔代数的表示完全一致，一个布尔值只有True、False两种值，要么是True，要么是False。在Python中，可以直接用True、False表示布尔值（请注意大小写），也可以通过布尔运算计算出来：print(True)：True，print(False)：False，print(5>3)：True，print(5>10)：False，print(5+10==15)：True。

布尔值可以用and、or和not运算。and运算是与运算，只有所有都为True，and运算结果才是True，例如print(3>5 and 10>8)：False。or运算是或运算，只要其中有一个为True，or运算结果就是True，例如print(3>5 or 10>8)：True。not运算是非运算，它是一个单目运算符，把True变成False，False变成True，例如print(not 3>5)：True。

1. 空值

空值是Python里一个特殊的值，用None表示。None不能理解为0，因为0是有意义的。

（6）变量

变量的概念基本上和初中代数的方程变量是一致的，只是在计算机程序中，变量不仅可以是数字，还可以是任意数据类型。变量在程序中就是用一个变量名表示了，变量名必须是大小写英文、数字和\_的组合，且不能用数字开头，比如：a = 1，变量a是一个整数；t\_007 = 'T007'，变量t\_007是一个字符串；Answer = True，变量Answer是一个布尔值True。在Python中，等号是赋值语句，可以把任意数据类型赋值给变量。

（7）格式化

%运算符就是用来格式化字符串的，在字符串内部，有几个%?占位符，后面就跟几个变量或者值，顺序要对应好。如果只有一个%?，括号可以省略。常见的占位符有：%d——整数、%s——字符串，%f——浮点数(四舍五入保留6位小数)。例如’%d11’%100结果为10011，’%scde’%’ab’结果为abcde，’%f456’%1.2345678结果为1.234568456，’%d456%sdef%f’%(123,’abc’,1.234567)结果为123456abcdef1.234567。如果字符串里面的%是一个普通字符怎么办，就需要进行转义，用%%来表示一个%。另一种格式化字符串的方法是使用字符串的format()方法，它会用传入的参数依次替换字符串内的占位符{0}、{1}......，例如’{0}456{1}def{2}’.format(123,’abc’,1.2)结果为123456abcdef1.2。

## 列表与元组

（1）list列表

list是Python的一种有序列表，列出班里所有同学的名字，就可以用一个list表示：classmates=[‘小明’,’小红’,’小刚’]，变量classmates就是一个list，使用len函数可以获取列表的元素个数：len(classmates)=3。可以使用索引来访问列表的元素，注意索引从0开始classmates[0]表示第一个元素：小明，最后一个元素的索引为len(classmates)-1，如果索引超出范围Python会报一个IndexError错误。此外，可以利用-1，-2，-3来获取列表倒数几个的元素。

list是一个可变的有序表，可以往list中追加元素到末尾：classmates.append(‘小黄’)。也可以把元素插入到指定的位置，比如索引号为1的位置：classmates.insert(1,’小蓝’)。要删除list末尾的元素，用pop()方法：classmates.pop()，如果想要删除指定位置的元素，则使用pop(i)，i为索引。要把某个元素替换成别的元素，可以直接赋值给对应的索引位置：classmates[1]=’小白’。列表中的元素可以也是列表，例如classmates=['小白', ['小红', '小蓝'], '小明', '小刚']，classmates[1][1]结果等于小蓝。此外，可以定义一个空的列表，classmates=[]

（2）tuple元组

Python的另一种有序列表叫元组：tuple。tuple和list非常类似，但是tuple一旦初始化就不能修改，比如同样是列出同学的名字：classmates=(‘小明’,’小红’,’小刚’)。classmates这个tuple不能变，它没有append()，insert()这样的方法。其他获取元素的方法和list是一样的，可以正常地使用classmates[0]，classmates[-1]，但不能赋值成另外的元素。在定义tuple时需要注意的是当tuple只有一个元素时，需要在元素后面加上逗号，以此来区分小括号，例如classmates=(‘测试’,)。但有一些特殊的情况tuple是可以变的，比如tuple中的元素包含列表时，列表中的元素时可以改变的，例如classmates=(‘小明’,[’小红’,’小蓝’],’小刚’)，classmates[1]是一个列表，可以进行append、pop、insert、替换元素邓操作。所谓的“不变”是说，tuple的每个元素，指向永远不变。即指向'a'，就不能改成指向'b'，指向一个list，就不能改成指向其他对象，但指向的这个list本身是可变的。

## 字典与集合

1. dict字典

Python内置了字典：dict的支持，dict全称dictionary，在其他语言中也称为map，使用键-值（key-value）存储，具有极快的查找速度，例如{'小明':99,'小红':92,'小刚':95}就是一个典型的字典。dict内部存放的顺序和key放入的顺序是没有关系的，所以不能通过索引的方式对字典进行操作

创建字典有以下常用方法：

1.用等号创建

dict = {'name': 'Tom', 'age': 18}

2.使用dict()和zip()利用已有数据创建

keys = ['a', 'b', 'c']

values = [1, 2, 3]

dict = dict(zip(keys, values))

3.创建空字典

dict = dict()

4.以给定内容为键，创建值为空的字典

dict = dict.fromkeys(['name', 'age'])

可以通过字典的key快速找到其value值，例如chengji[‘小明’]的结果为99。将数据存入字典也可以通过key放入，例如chengji[‘小蓝’]=90。由于一个key只能对应一个value，所以，多次对一个key放入value，后面的值会把前面的值冲掉。也可以通过字典的setdefault(key,value)方法往字典里面添加条目，如果键已经存在则添加失败，例如chengji.setdefault(‘小蓝’,95)。当查询的key值在字典中不存在时，dict就会报错。可以通过in来判断key值是否存在在字典中，例如’小蓝’ in chengji，返回的值是一个布尔型。也可以通过dict的get()方法，如果key不存在，返回的结果为None或者自己定义的值，如果存在则返回对应的value值，例如chengji.get(‘小明’,-1)。

要删除一个key，可以使用del、pop(key)方法，对应的value也会从dict中删除，区别在于pop(key)方法会返回对应key的value值，而del方法没有返回值，用法为del(chengji[‘小明’])，chengji.pop(‘小明’)。通过字典的items()方法可以返回键值对的列表，通过字典的keys()方法可以返回键的列表，通过字典的values()方法可以返回值的列表，通过字典的clear()方法删除字典的所有键值。update()方法将另一个字典中的键值对添加到当前字典中，若键已存在，则把其对应的值修改为新的字典中相应的值。例如：dict1={‘a’:1,’b’:2,’c’:3}，dict2={‘d’:4,’e’:5}，dict1.update(dict2)结果为{‘a’:1,’b’:2,’c’:3,‘d’:4,’e’:5}。

（2）set集合

set和dict类似，也是一组key的集合，但不存储value。key不能重复，所以在set中，没有重复的key。set可以使用{}或者set()函数进行创建，创建空集合必须set()方法。

使用set(str)方法的方法创建集合会把每字符串每一个字符拆分成元素，例如set(‘abcd’)创建的集合为{‘a’,’b’,’c’,’d’}。也可以通过set(list\tuple)来创建集合，例如set([‘a’,’b’,’c’,’d’])创建的集合为{‘a’,’b’,’c’,’d’}，而使用{}是直接定义一个集合。

通过集合的add(key)方法可以往集合中添加元素，例如set.add(‘e’)，如果集合已经存在元素e，那么添加将无效。使用update方法也可以往集合中添加元素，语法为set.update(x)，x可以为多个，用逗号隔开，参数可以是列表、元组等。通过集合的remove(key)方法可以将集合中的元素删除，例如set.remove(‘a’)，如果元素不存在则会报错。另外一种删除元素方法discard(key)，元素不存在的话也不会报错。此外，还有清空集合的函数clear()。

集合与集合之间也可以进行操作，例如set1 - set2(集合1中含有的元素且集合2中不含有)，set1|set2(集合1与集合2所有元素)，set1&set2(集合1与集合2共同包含的元素)，set1^set2(集合1与集合2不同时包含的元素)。

## 条件判断

计算机之所以能做很多自动化的任务，因为它可以自己做条件判断。比如，输入用户年龄，根据年龄打印不同的内容，在Python程序中，用if语句实现：

age = 20

if age >= 18:

print('your age is', age)

print('adult')

根据Python的缩进规则，如果if语句判断是True，就把缩进的两行print语句执行了，否则，什么也不做。也可以给if添加一个else语句，意思是，如果if判断是False，不要执行if的内容，去把else执行了：

age = 3

if age >= 18:

print('your age is', age)

print('adult')

else:

print('your age is', age)

print('teenager')

注意不要少写了冒号，当然上面的判断是很粗略的，完全可以用elif做更细致的判断：

age = 3

if age >= 18:

print('adult')

elif age >= 6:

print('teenager')

else:

print('kid')

if语句执行有个特点，它是从上往下判断，如果在某个判断上是True，把该判断对应的语句执行后，就忽略掉剩下的elif和else。

## 循环

（1）for循环

Python的循环有两种，一种是for...in循环，依次把list或tuple中的每个元素迭代出来，例如：

names = ['Michael', 'Bob', 'Tracy']

for name in names:

print(name)

（2）while循环

第二种循环是while循环，只要条件满足，就不断循环，条件不满足时退出循环，例如计算1—10整数之和：

sum=0

i=1

while i<=10:

sum=sum+i

i+=1

print(sum)

1. break结束循环

在循环中，break语句可以提前退出循环，例如：

sum=0

i=1

while i<=10:

sum=sum+i

i+=1

if i==5:

break

print(sum)

结果为1+2+3+4。

（4）continue进入下一个循环

在循环过程中，也可以通过continue语句，跳过当前的这次循环，直接开始下一次循环，例如：

n=0

while n<10:

n+=1

if n%2==0:

continue

print(n)

结果输出1，3，5，7，9。

# 函数

## Python常用内置函数

（1）数学相关

abs(a) : 求取绝对值，abs(-1)=1。

max(list) : 求取list最大值，max([1,2,3])=3。

min(list) : 求取list最小值，min([1,2,3])=1

sum(list) : 求取list元素的和，sum([1,2,3])=6。

sorted(list) : 排序，返回排序后的list，sorted([5,2,7,9,1])=[1,2,5,7,9]。

len(list/str) : 获取list/str长度，len([1,2,3])=3，len(‘中国qwe’)=5。

divmod(a,b): 获取商和余数，divmod(10,3)=(2,1)。

pow(a,b) : 获取乘方数，pow(2,4)=16。

round(a,b) : 获取指定位数的小数（四舍五入），a代表浮点数，b代表要保留的位数，round(3.145678,2)=3.15。

range(a,b,i) : 生成一个a到b，差值为i左闭右开的数组，range(2,10,2)=[2,4,6,8]。

（2）类型转换

int(str) : 转换为int型，int('1')=1，int(1.9)=1。

float(int/str) : 将int型或字符型转换为浮点型，float(1)=1.0，float(‘5.1’)=5.1。

str(int) : 转换为字符，str(1)='1'。

list(iterable) : 转换为list，list((1,2,3))=[1,2,3]，list({1,2,3})=[1,2,3]。

dict(iterable) : 转换为dict，dict([('a', 1), ('b', 2), ('c', 3)])={'a':1, 'b':2, 'c':3}。

tuple(iterable) : 转换为tuple，tuple([1,2,3])=(1,2,3)。

set(iterable) : 转换为set，set([1,4,2,4,3,5])={1,2,3,4,5}。

（3）功能相关

eval(expression, globals=None, locals=None)：执行一个表达式，或字符串作为运算，也可以把字符串转换为列表、元组、字典、集合等，只能执行单个运算表达式，不能是复杂的代码逻辑。expression：必选参数，可以是字符串。globals：可选参数，表示全局命名空间（存放全局变量），如果被提供，则必须是一个字典对象。locals：可选参数，表示当前局部命名空间（存放局部变量），如果被提供，可以是任何映射对象。如果该参数被忽略，那么它将会取与globals相同的值。用法例如：eval(“’a’+’b’”)=ab，eval(‘1+2’)=3，eval(“[‘a’,’b’]”)=[‘a’,’b’]。

exec(expression, globals=None, locals=None)：动态执行Python代码，也就是说exec可以执行复杂的Python代码，而不像eval函数那样只能计算一个表达式的值，而且与eval相比，exec()函数返回值永远都是None。

lambda argument\_list:expression：argument\_list是参数列表，它的结构与Python中函数(function)的参数列表是一样的（输入）。expression是一个关于参数的表达式，表达式中出现的参数需要在argument\_list中有定义，并且表达式只能是单行的（输出）。

zip(iterable,...)：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。如果各个迭代器的元素个数不一致，则返回列表长度与最短的对象相同。

## 函数定义

在Python中，定义一个函数要使用def语句，依次写出函数名、括号、括号中的参数和冒号:，然后，在缩进块中编写函数体，函数的返回值用return语句返回。例如自定义一个输出数值绝对值的函数：

def test(x):

if x>=0:

return x

else:

return -x

函数体内部的语句在执行时，一旦执行到return时，函数就执行完毕，并将结果返回。如果没有return语句，函数执行完毕后也会返回结果，只是结果为None，return None可以简写为return。

**（1）空函数**

如果想定义一个什么事也不做的空函数，可以用pass语句。pass语句什么都不做，那有什么用？实际上pass可以用来作为占位符，比如现在还没想好怎么写函数的代码，就可以先放一个pass。

**（2）函数返回值**

函数可以有多个返回值，返回值其实是一个tuple元组。例如：

def test(x,y):

return x+y,x\*y

test(4,5)的结果为：(9,20)。

## 函数的参数

**（1）必选参数**

必选参数是函数中最常用的参数类型，在调用含有必选参数函数时，参数必须传入，且一个参数有且对应一个值，例如：

def test(x,y):

return x\*y

x与y就是必选参数。

**（2）默认参数**

定义函数参数值可以给默认值，在调用这种类型的函数，有默认值的参数可以不用给，但要注意默认值参数的位置都是在没默认值参数的后面。例如：

def test(x,y,z=10):

return x+y+z

test(1,2)=13，test(1,2,3)=6。

**（3）可变参数**

在Python函数中，还可以定义可变参数。顾名思义，可变参数就是传入的参数个数是可变的，可以是1个、2个到任意个，还可以是0个。可变参数定义时是在参数前面加\*，当调用函数时，函数接受到的参数是一个tuple。例如：

def test(\*number):

sum=0

for i in number:

sum=sum+pow(i,2)

print(sum)

test(1,2,3,4)=30，test(1)=1。可变参数函数可以使用列表元组集合作为参数，使用时只需要在列表元组集合前也加\*即可。例如：

list=[1,2,3,4]，test(\*list)=30。

**（4）关键字参数**

关键字参数也允许传入任意个参数，这些参数在函数内部会自动组装为一个dict字典，定义时只需要在参数名称前加\*\*。例如：

def test(\*\*number):

sum=0

for num in number:

sum=sum+number[num]

print(sum)。

test(a=1,b=2,c=3)=6。关键字参数函数可以使用字典作为参数，使用时只需要在字典前也加\*\*即可。例如：

dict={‘a’:1,’b’:3,’c’:5}，test(\*\*dict)=9

**（5）设置关键字参数的关键字**

如果要限制关键字参数的名字，就可以用命名关键字参数。例如，只接收c和d作为关键字参数，这种方式定义的函数如下：

def test(a,b,\*,c,d):

print(a,b,c,d)

如果函数定义中已经有了一个可变参数，后面跟着的命名关键字参数就不再需要一个特殊分隔符\*，例如：

def test7(a,b,\*numbers,d,e):

sum=a+b

for num in numbers:

sum=sum+num

sum=sum+d+e

print(sum)

test7(1,2,3,4,5,6,d=7,e=8)

这种情况下，关键字参数一定要传入，否则将会报错。

**（6）组合参数**

定义函数，可以用必选参数、默认参数、可变参数、关键字参数和命名关键字参数，这5种参数都可以组合使用，但需要注意参数定义的顺序必须是：必选参数、默认参数、可变参数、命名关键字参数、关键字参数。

## 递归函数

在函数内部，可以调用其他函数。如果一个函数在内部调用自身本身，那么这个函数就是递归函数。例如计算1\*2\*3\*4......\*n：

def test(n):

if n==1:

return 1

return n\*test(n-1)。

test(4)=4\*test(3)=4\*(3\*test(2))=4\*(3\*(2\*test(1)))=4\*(3\*(2\*(1)))=4\*3\*2\*1=24。

使用递归函数需要注意防止栈溢出，解决递归调用栈溢出的方法是通过尾递归优化， 尾递归是指在函数返回的时候，调用自身本身，并且return语句不能包含表达式。例如上述函数可优化为：

def test(n,result):

if n==1:

return result

return test(n-1,n\*result)

n-1,n\*result在调用函数前就会被计算，不会影响到函数。

# 高级特性

## 切片

切片根据索引可以取出列表元组指定范围的数，例如元组tuple=(‘a’,’b’,’c’,’d’,’e’)，取出这个元组的第1个到第3个数写法为tuple[0:3]，其中索引3不包括，结果为(‘b’,’c’,’d’)。如果想要从第n个值取到最后一个值，则这么写tuple[n:]。切片也支持用索引取出倒数的数，例如tuple[-3:]取最后3个数，结果是(‘c’,’d’,’e’)。切片还有其他的操作，例如每多少个中取一个：

liebiao=list(range(100))，liebiao[:10:2]：前十个每两个取一个，liebiao[::5]：每五个取一个。此外切片还可以对字符串进行操作，例如str=’abcdefg’，str[:4:3]=’abc’，str[::2]=’ac’。

## 迭代

如果给定一个list或tuple，可以通过for循环来遍历这个list或tuple，这种遍历称为迭代，例如:

list=list(range(0,10,2))

for i in list:

print(i)

结果为：0,2,4,6,8。

python可以进行迭代的对象不仅局限于list或tuple，dict、set、str也可以进行迭代。判断一个对象是否可以进行迭代，可以通过collections.abc模块的Iterable类型：

from collections.abc import Iterable

print(isinstance([‘a’,’b’,’c’],Iterable))

print(isinstance((‘a’,’b’,’c’),Iterable))

print(isinstance(‘abc’,Iterable))

结果为True,True,False。

enumerate(Iterable,start=开始下标) 函数用于将一个可遍历的数据对象组合为一个索引序列，同时列出数据和数据下标。需要注意的是开始下标并不是迭代数据的开始时游标，只是起到一个开始编号的作用，例如：

tuple=(‘a’,1,’b’,2)

for i,value in enumerate(tuple,start=2):

print(i,’:’,value)

结果为2:a,3:1,4:b,5:2。

## 列表生成式

生成一个list，常用的方法是利用list函数进行转换，例如：

list(range(10))

set={1,2,3,4}

tuple=(1,2,3,4)

list(set)

list(tuple)

也可以通过for循环来创建列表，例如：

[x for x in range(5)]=[0,1,2,3,4]

for循环可以使用多层，例如：

[m+n for m in range(5) for n in(5,10)]=[0+5,1+6,2+7,3+8,4+9]=[5,7,9,11,13]

在for循环中还可以使用判断语句，例如：

[pow(x,2) for x in range(10) if x%2==0]=[0,4,16,36,64]

如果想要在for循环中使用if......else，if......else必须写在for前面，例如：

[x if x>=0 else -x for x in range(-4,5)]=[4,3,2,1,0,1,2,3,4]

# 高阶函数

函数可以赋值给变量，例如：

function=abs

function(-10)=10

函数的参数可以接受变量，例如：

b=10

def test(x,y):

return x\*y

test(10,b)=100

当函数的变量为函数时，这种函数就称之为高阶函数，例如：

f=pow

def test(x,y,function):

function(x,y)

test(2,4,f)=16

## map()函数

map(function,Iterable)函数接收两个参数，一个是函数，另一个是可以进行迭代的对象，返回的是以迭代对象作为参数计算的函数结果并封装成一个Iterator迭代器，迭代器是惰性序列，可以通过list()或者tuple()函数返回一个list或者tuple，例如：

def test(x):

return x\*x

list(map(test,range(10)))=[0,1,4,9,16,25,36,49,64,81]

## reduce()函数

reduce()函数会对参数序列中元素进行累积，具体的积累方式是将一个数据集合中的数据进行下列操作：用传给 reduce 中的函数 function（有两个参数）先对集合中的第 1、2 个元素进行操作，得到的结果再与第三个数据用 function 函数运算，以此类推最后得到一个结果，例如：

def test(x,y):

return x\*y

reduce(test,range(1,5))=1\*2\*3\*4=24

## filter()筛选函数

filter(function,Iterable)筛选函数，函数用于过滤序列，过滤掉不符合条件的元素。同样的也是需要两个参数，第一个为函数，第二个为序列。序列的每个元素作为参数传递给函数进行判，然后返回 True 或 False，最后将返回 True 的元素放到新列表中，例如：

def test(x):

return x%2==0

list(test(range(10)))=[0,2,4,6,8]

## sorted()排序函数

sorted(Iterable, key=None, reverse=False)排序函数，用来排序可迭代的对象，结果返回一个list。Key：主要是用来进行比较的元素，只有一个参数，具体的函数的参数就是取自于可迭代对象中，指定可迭代对象中的一个元素来进行排序。reverse：排序规则，reverse = True 降序，reverse = False 升序（默认）。对一个列表进行排序，例如：

list=[9,2,6,5,1]

sorted(list)=[1,2,5,6,9]

也可以对字典的key、value、item进行排序，例如：

dict={'a':9,'h':2,'c':7,'e':5,'g':10}

sorted(dict.keys())=[a,c,e,g,h]

sorted(dict.values())=[2,5,7,9,10]

sorted(dict.items(),key=lambda x:x[1],reverse=True)=[('g', 10), ('a', 9), ('c', 7), ('e', 5), ('h', 2)]

list自带的函数sort()与sorted()函数很像，但是sort()只能应用在list，sorted()可以对所有可迭代的对象进行排序操作。sort()是对已经存在的列表进行操作，无返回值，而sorted()方法返回的是一个新的list，不会对传入的list产生影响。例如：

list.sort()

print(list)=[1,2,5,6,9]

sorted(list)=[1,2,5,6,9]

print(list)=[9,2,6,5,1]

## 匿名函数

lambda函数（匿名函数）：lambda argument\_list:expression。argument\_list是参数列表，它的结构与Python中函数(function)的参数列表是一样的（输入）。expression是一个关于参数的表达式，表达式中出现的参数需要在argument\_list中有定义，并且表达式只能是单行的（输出）。例如：

function=lambda x,y:x\*y

function(4,5)=20

# 模块

在计算机程序的开发过程中，随着程序代码越写越多，在一个文件里代码就会越来越长，越来越不容易维护。为了编写维护性比较的代码，一般会对函数进行分组，分别放到不同的文件里，这样，每个文件包含的代码就相对较少，很多编程语言都采用这种组织代码的方式。在Python中，一个.py文件就称之为一个模块。

使用模块最大的好处是大大提高了代码的可维护性，其次，编写代码不必从零开始。当一个模块编写完毕，就可以被其他地方引用。在编写程序的时候，也经常引用其他模块，包括Python内置的模块和来自第三方的模块。使用模块还可以避免函数名和变量名冲突，相同名字的函数和变量完全可以分别存在不同的模块中。因此，在编写模块时，不必考虑名字会与其他模块冲突。但是也要注意，尽量不要与内置函数名字冲突。

如果不同的人编写的模块名相同怎么办？为了避免模块名冲突，Python又引入了按目录来组织模块的方法，称为包（Package）。举个例子，比如有两个abc.py文件，这个时候可以通过包来组织模块。方法是选择两个包，比如mycompany1、mycompany2，然后把这两个文件分别存放到这两个包的目录下。那这时候，两个模块的名称就变成了mycompany1.abc、mycompany2.abc。需要注意的是每一个包目录下面都会有一个\_\_init\_\_.py的文件，这个文件是必须存在的，否则，Python就把这个目录当成普通目录，而不是一个包。\_\_init\_\_.py可以是空文件，也可以有Python代码，因为\_\_init\_\_.py本身就是一个模块，而它的模块名就是mycompany。类似的，可以有多级目录，组成多级层次的包结构。创建模块时要注意命名，不能和Python自带的模块名称冲突。例如，系统自带了sys模块，创建的模块就不可命名为sys.py，否则将无法导入系统自带的sys模块。

## 创建模块

创建模块其实就是建一个.py文件，通常情况下会把类似的功能代码写在同一个.py文件中，例如以下test.py文件就是一个模块：

# -\*- coding: utf-8 -\*-

class Moudle():

def func1(x,y):

return x\*y-(x+y)

def func2(x,y):

return pow(x,y)-(x+y)

if \_\_name\_\_ == '\_\_main\_\_':

print(Moudle.func1(4,2))

print(Moudle.func2(4,2))

首行信息指定了运行该文件代码使用的编码格式，该模块包含了两个函数。

## 使用模块

使用模块前，需要导入模块，导入的时候需要注意如果引用的模块在不在包里面，那么导入不能使用相对路径的导入（例如：from ..test import Test），否则会报找不到父包的错误信息。同时需要注意，如果导入的模块中有执行语句，那么一定要将执行语句写在if \_\_name\_\_ == '\_\_main\_\_':判断语句里面，否则引用这个模块时，去执行将会执行引用模块中的执行语句。例如，上述创建的模块如果没有把执行语句写在if \_\_name\_\_ == '\_\_main\_\_'，那么再其他地方引用该模块时，将会运行这两条执行语句。（\_\_name\_\_是python的内建变量，每个.py文件都有这个变量，该变量可以显示模块是被自己执行的还是其他模块调用的。如果是自己执行的，把么\_\_name\_\_就等于\_\_main\_\_，如果是被别的模块调用，则显示模块的绝对路径信息）。