# 第一章 python基础

## 1.python概述

Python是一种面向对象、解释型计算机程序设计语言，由Guido van Rossum于1989年发明，第一个公开发行版发行于1991年。

python官网 : [https://www.python.org/](https://www.python.org/" \t "/home/jason/文档\\x/_blank)

## 2.python的发展

### 2.1 python能为我们做什么

在web服务器编程，数据挖掘，自然语言处理，自动化测试等方面应用越来越多。在国内目前的趋势是部分取代java和PHP在服务器端的作用。目前流行的网站创建模式LAMP(Linux，apache，Mysql，Python/PHP/Perl)即可以用python来完成后端服务。

### 2.2 python哲学

Python崇尚优美、清晰、简单，是一个优秀并广泛使用的语言。优美、清晰、简单也是python一直强调的哲学理念。

### 2.3 python语言的特点

python语言的开放性赋予了其强大的生命 这与linux相同python是一门面向对象的语言，模块，变量，函数都看作是对象。有时python被认为是一门脚本语言，但是他比传统意义的脚本语言功能上要强大得多，甚至google等公司的一些大型项目都用python，所以很多人也把他看作一种高级动态编程语言。

很好的跨平台性，也可以很容易的和多种编程语言衔接（比如c c++ java），所以我们也称python为胶水语言。

python简单的特点，被用作学习的第一种编程语言很合适，性价比较高

## 3. python开发环境的搭建

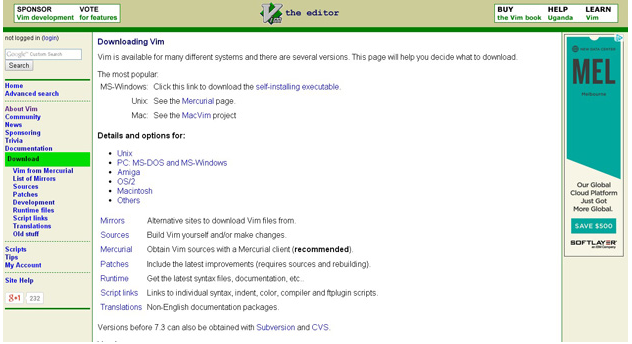
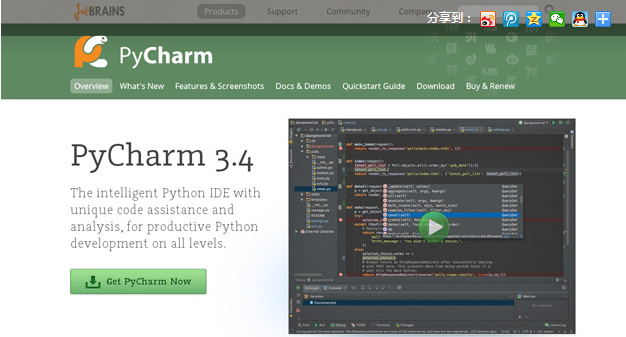
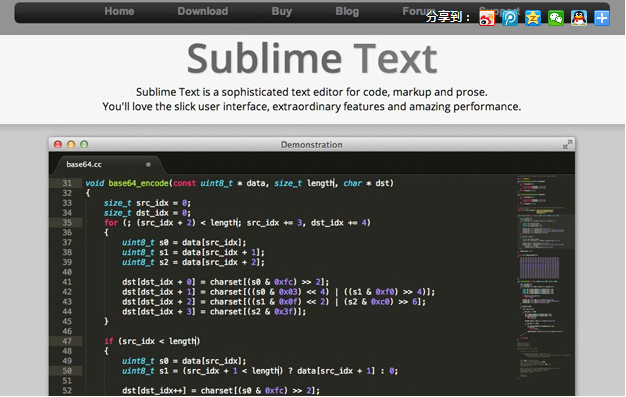
### 3.1.python版本介绍

python版本目前有python2.x与python 3.x,基本语法差别不是很大，python3在库方面有一些更新。

在系统支持上python支持 windows ,Mac OS X ,linux等系统，可以说python的跨平台性很强。

### 3.2 python常用开发工具

python在各个平台下都有比较优秀的开发工具，例举如下：

### 3.3 安装python

在这里分别为同学们介绍下python在windows和linux下的安装。

待添加:!!!

### 3.4编程语言类型介绍

1.编译型语言 如 c/c++语言   
 在程序执行之前，有一个单独的编译过程，将程序翻译成机器语言，以后执行这个程序的时候，就不用再进行翻译了。

2.解释性语言   
 运行的时候将程序翻译成机器语言，所以运行速度相对于编译型语言要慢。   
 脚本语言是一种解释性的语言,例如javascript,python等等,它不象cc++等可以编译成二进制代码,以可执行文件的形式存在。   
 脚本语言不需要编译，可以直接用，由解释器来负责解释。

3.半编译半解释性语言,如java,先生成字节码，python也是先生成字节码.pyc 。先编译生成字节码然后再由虚拟机负责解释。

4.交互式解释器：python shell等都有类似的交互解释器，用户可以直接在提示行后输入相应的语句并且运行出相应的结果，极大的方便了我们的学习。

## 4.python预备知识

### 4.1什么是程序

广义的讲，为了实现一个特定的目标而预先设计的一组可操作的工作步骤就可以称之为一个程序。对于计算机系统，简单的说，程序就是系统可以识别的一组有序的指令   
 程序存储在磁盘上，被加载到内存中，计算机系统从内存中逐条读取指令并执行

### 4.2计算机的存储结构

* 存储器可以简单的分为主存储器，外存储器和寄存器

1) 主存储器也称内存，是计算机的工作存储器。程序，待处理的数据和处理的结果都存储在内存中。内存通常以字节为单位对存储单元按顺序进行编址。内存读写速度比较快，但只能在加电时存储数据

2)外存储器是用来长期保存数据的大容量存储器，目前多为磁存储器或光存储器。读写速度慢于内存，可以在不加电的情况下长期保存数据

3) 寄存器是CPU内部的高速存储器，速度快，数目少。

### 4.3编程语言发展

第一阶段:机器语言   
 直接使用机器指令(0，1序列)来设计程序，可以 被计算机系统直接识别。和自然语言完全不同，难于记忆和理解，工作量大，效率低，无法移植

第二阶段:汇编语言   
 把机器指令符号化，即通过一组简单的的符号来表示机器指令，更接近于自然语言，更容易理解和使用。 不能被计算机系统直接识别，要转化成机器语言后才能执行。和机器语言一样，开发工作量大，无法移植

第三阶段:高级语言   
 和特定的计算机系统无关，更接近于人类自然语言。一条语句对应多条机器指令，工作量小，开发效率高。同样需要转换成机器代码后才能被计算机系统识别、执行。 JAVA , PYTHON, C, C++ ,C#

### 4.4进制转换

凡是按进位的方式计数的数制叫做进位计数制，简称进位制。   
 基数为2的进位制叫二进制 ,为8的进位制叫八进制 ,为10的进位制叫十进制 ,为16的进位制叫十六进制

### 4.5编码方式

计算机中数据的概念是广义的，机内除有前面提到的数值数据之外，还有文字、符号、图像、语言和逻辑信息等等，因为它们也都是以0、1形式存在，所以成为非数值数据   
 字符数据主要指数字、字母、通用符号、控制符号等等，在机器内都被变换成计算机能够识别的二进制编码形式。国际上被普遍采用的一种编码是美国国家信息交换标准代码，简称为ASCII   
 Unicode（统一码、万国码、单一码）是计算机科学领域里的一项业界标准,包括字符集、编码方案等。Unicode 是为了解决传统的字符编码方案的局限而产生的，它为每种语言中的每个字符设定了统一并且唯一的二进制编码，以满足跨语言、跨平台进行文本转换、处理的要求。1990年开始研发，1994年正式公布。其中UTF-8（8-bit Unicode Transformation Format）是一种针对Unicode的可变长度字符编码，又称万国码。由Ken Thompson于1992年创建。现在已经标准化为RFC 3629。UTF-8用1到6个字节编码UNICODE字符。用在网页上可以同一页面显示中文简体繁体及其它语言（如英文，日文，韩文）。

## 5.程序的运行

### 5.1通过命令提示行运行

通过命令行运行python即为交互模式，我们可以启动ipython 或者 python3 进入交互状态然后就可以写python语句了。每次启动我们认为是一个程序进程，交互状态和程序本身有一点

差别，如可以直接显示变量的值等。

退出: python2: quit()或者ctrl + D或者 exit(),如果安装了ipython插件,则不用加().

### 5.2通过脚本程序运行

通过脚本程序运行即我们之前提到的“程序”是我们将python程序写入到一个文件中，通过运行文件即可让程序按照我们预定的规则执行。

普通的python文件为.py格式的文件,可以通过编译的方法生成.pyc格式的字节码文件,该文件可增加文件的执行效率.

生成.pyc字节码文件的方法:

Import py\_compile

py\_compile.compile(“1,py”) # 1.py 为需要编译的文件名

生成.pyo格式优化代码的方法:

python -O -m py\_compile 1.py

### 5.3 python程序的执行过程

#### 1)python 程序的组成

* 程序由模块构成
* 模块包含语句
* 语句包含表达式
* 表达式建立并处理对象

python程序是按照语句顺次执行的。类，函数等语句块需要调用后才能执行具体内容。

#### 2) python关键字

1) 循环判断:

if; elif; else; for; while; bieak; continue; and; or; is; not in;

2) 函数模块类:

from; import; as; def; pass; lambda; return; class;

3) 异常:

try; except; finally; raise;

4) 其他:

del; global; with; assert; yield; nonlocal;

True; False; None; exec print (python3中, print () 为打印)

## 6数字和表达式

### 6.1 python的核心数据类型

1)数字（包含整形Int和浮点型Float） 字符串（String） 列表（List） 字典(Dict) 元组(Tuple) 集合(Set) 文件

2)其他类型：类类型(object);None(Nonetype);布尔型(Boolean)

3)编程单元模块类型：函数(function);模块(module);类(class)

### 6.2数字类型

1. 如果是长整形后面可以加一个L （python3中去掉了long类型所有的整形就是long 浮点型就是double）(float与double的区别?)
2. 在python中每个数都是一个对象，创建时存在内存中
3. python 中数字也支持科学计数法，复数等。使用科学技术法时,需要引入 math函数?,如何使用科学技术法.

### 6.3常量与变量

1)常量：不能被改变的量

2)变量：人为设定有的标识符，用来标记常量，可以被重复利用赋值

#### 3) 标识符命名规则

标识符是由程序员按照命名规则自行定义的词法符号，用于定义宏名、变量名、函数名和自定义类型名等。python的命名规则如下：

由字母,数字,下划线组成;数字不能作为开头;不能使用关键字;严格区分大小写.

但是在python的标识符命名中还有一些默认规则:

1) (\_x)格式的变量名, 不会被 from module import \* 语句导入

2) (\_\_x\_\_)格式的变量名,为系统的默认变量

3) (\_\_x)格式的变量名,为类的私有变量

#### 4) 变量的初始化和使用

变量使用前必须赋值（初始化）:

python属于弱类型变量： 对象有类型变量无类型，变量就相当于一个对象的标签

Python 中变量和常量是分开存放的，当把变量重新赋值的时候即变量存放的地址发生了改变。不用的常量仍然在内存中，有自动的垃圾处理机制，如果没有变量绑定这个常量则自动会被处理也可以通过del来处理，全局变量也随进程消亡.

## 7运算符及优先级

表达式是python程序的基本组成部分，下面我们先从最今本的四则运算来学习一下python中的表达式。

### 7.1简单的运算

+; -; \*; /;

赋值: a = 1 ; a,b = 1,2 ;

### 7.2除法

python3 中: / 为真除法,及保留商的小数部分; // 为假除法.及整除

### 7.3运算符及优先级

各个运算符的优先级如下图,优先级有低到高,

逻辑运算中的短路原则:

a>b or a>c and a<d

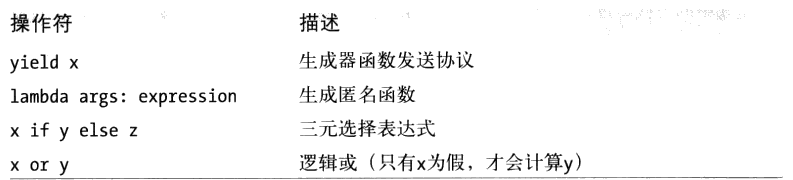
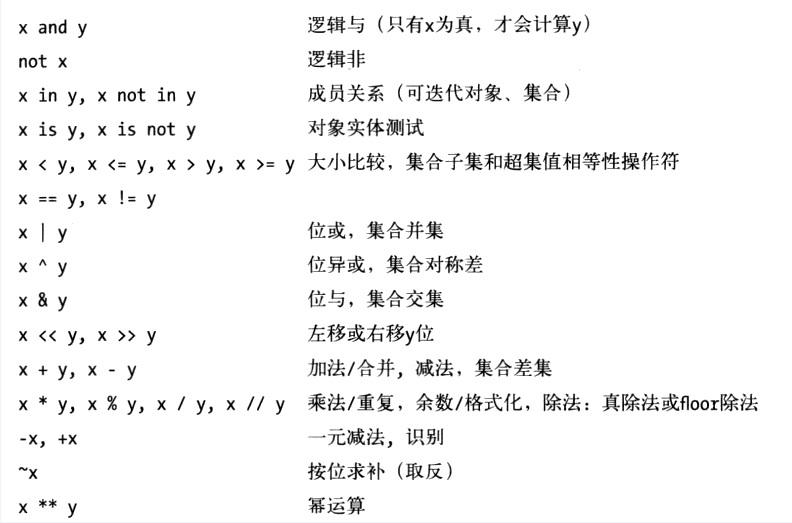
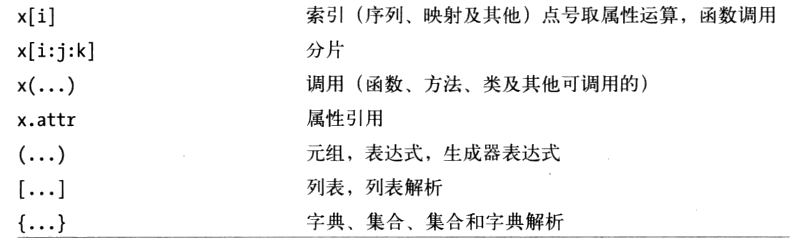
逻辑判断由左到右依次判断,

1)当第一个 or 条件的判断为True时, 不再继续判断, 整体判断结果为True

2)当 or 判断结果为False 时 向右继续执行

3)当 and 前的语句判断为 False时,不再继续判断.,整体判断结果为 False

4)当 and 前的语句判断为 True 时, 继续判断, 直到 所有的 and 的左右两边都为True时, 整体返回 True ,

## 8运算

### 8.1运算类型

* 数学运算 +; -; \*; /; //; %;
* 位运算 «:左移运算 »:右移运算 |:或运算 ^:异或运算 &:与运算 ~:非运算
* 逻辑运算 ==; ！= ; or; and; not; is; is not; in; not in

### 8.2特殊运算及一些技巧:

1) a,b两数交换的方法:

①: a^=b b^=a a^=b

②: a,b = b,a

③: tmp=a a=b b=tmp

2) 左移位运算:<< :换算为2进制数, 向左移动,右边用0补位

3) 右位移运算:>> : 换算为2进制数,向右移动,即去掉右面相应位数的数值.

4) 与运算:& :一假则假;

5) 或运算: | :一真则真;

6) 异或运算: ^ : 换算为2进制数,相同为0,不同为1

7) 取反 : ~ :

注： python 中支持 += 这种运算赋值写法，但是不支持++,-–运算

## **8.3 math模块和类型转换**

python 中除了可以用基本的运算符进行运算，还提供了专门用于运算的模块如math模块。我们可以通过help方法来查看math模块中有哪些方法。

import math # 引入math函数

math.pow(3,2) #幂运算, 结果为浮点型数字

math.sqrt(9) #开方运算

强制类型转换

在python中数据类型的强制转换通过一些内建函数来实现。有时候这种强制转换时不安全的，因为这是一种舍去精度的转换。

int(1.23) #强制转换为整型

str(123) # 强制转换为字符型,

# 第二章python 数据类型

## 1.字符串

### 1.1字符串常量

字符串 ： 零个或多个字符组成的有限串行   
 字符串常量：必须用引号引起来，用单引号双引号都可以，当一个字符串里面出现引号是,应与最外层的引号区分开来,如果内层单引号和双引号都有,则需要通过转义字符来区分开.  
 三引号 ：文本的原样保留，在程序编写如果字符串需要换行或者有些内容需要严格的原样输出的时候使用。还可以生成help目录。

### 1.2 变量和字符串

字符串是一种容器数据类型，他是由多个字符组成的，字符串是一种不可变数据类型。

### 1.3转义字符

转义字符：字母前加”\“来表示常见的那些不能显示的ASCII字符，称为转义字符，因为后面的字符，都不是它本来的ASCII字符意思了



### 1.4字符串操作

1. 字符串的拼接:+ 重复: \*

2. repr(),str() 和 ``, 强转为字符串, 3.x中已经不使用反撇符号

3. 字符串的切片操作:

用 [] 完成对字符串的操作，我们用 : 将字符串中的内容分为三部分   
第一部分为要取得字母下标，其中第一个下标为0   
有第二部分的时候表示取字符的范围，不包含后标记   
有第三部分的时候第三部分表示步长

str[I] #输出str中下标为I 的字符,

str[start[:stop:step]] # start:起始下标位置, stop: 终止下标位置(不 # 包含终止位置) step:步长

str[::-1] # 将str反转

## 2 输入输出

在python中我们的输入输出由两个内建函数完成。

内建函数，就是由语法规定存在的函数。程序员不必单独书写代码实现它，只需要调用既可，他们的实现，由编译器自动完成。

### 2.1 input

在2.x版本中有raw\_input,和input两个函数   
 这两个函数均能接收字符串 ，但 raw\_input() 直接读取控制台的输入（任何类型的输入它都可以接收）,将所有的输入看做为字符串,返回字符串类型

input(), 严格按照python语句标准输入内容,输入的数字具有自己的特性,返回数字的类型,及输入元组时,返回的类型就是元组.

在3.x的版本中只有input的返回值都是字符串类型，基本上和2.x中的raw\_input用法类似

在输入过程中当有连续输入的时候以回车作为分隔符

### 3.2 print

在 2.x中 print不加括号的，需要不换行时,在最后添加逗号.

在3.x中 print后加括号,需要不换行时: print(a,end=””)

1)直接输出: print a

2) 变量: 可以输出任何类型

3)格式化输出:

①:%: 标记转化说明符的开始,

② 转换标志:

- 表示左对齐,;

+:表示给正数加上+号;

“”:正数之前加空格;

0: 转化值位数不够时,用0补位

③ 最小字段宽度:转换后的字符串至少有该值指定的宽度,如果是\*,则宽度会从元组中辅助,

④ 在 . 后加精度值,如果转换的是实数,精度值为小数点后的位数,如果为字符串,则该数字为最大字段宽度,如果是\*,

⑤ 转换类型:

%d:整形; eg:print "%d %d"(%a,%b) 传入两个数 值,个数严格匹配

%f:浮点型 eg: a=1.23 print "%.1f"%a 输出 1.2

%e:科学计数法

%E:科学计数法(大写)

%o:八进制数

%x:十六进制数

%X:十六进制数(大写)

%r:字符串(使用repr转换任意python对象)

%s:字符串(受用str转换任意python对象)

## 3 注释

python中只有单行注释 用‘#’标记

如果一定有多行注释的需求也可以用三引号来完成，这样会认为引号中的内容为该模块的文档注释   
 良好的代码风格注释一般占代码量的15%以上

## 4列表 list

列表，元组，字典，集合还有字符串，我们可以称之为容器数据类型或者数据结构。所谓的容器数据类型（或者数据结构）是将基本数据类型形成一个集合整体，或者构建一定的模型，给操作带来了很大的便利。

### 4.2 定义 :

列表示一种容器类型，是任意对象的有序集合，可通过索引访问其中的元素。   
 命名规则:list=[],

列表是有序的,从下标0开始计算,列表中可以存储其他类型列表是可变数据类型,支持切片操作,列表当中可以嵌套列表,

### 4.2 列表操作 :

叠加: + :

重复: \*:

删除列表元素:del(list[I]):

索引:list.index(X)

切片:list[start:[:stop:step]] 或者 list[stop] list[::-1]: 将列表反转

3)列表方法

help(list): 查看list帮助文档

list.append(X): 将X 添加到list后,

list.extend(list1): 将list1追加到list后面

list.insert(I,X): 在下标为I的位置,添加X到list中

list.count(X):统计list中X的个数

list.reverse(): 将list反转

list.sort(): 将列表中的值按照从小到大排列,如果为字符,则按照字符的ascii码逐个比较,按照从小到大的顺序排列出来

list.pop(I): 将list下标为I的元素弹出

list.remove(X):将list中的X元素删除

len(list):计算list的长度

## 5元组 tuple ( )

1)定义   
 元组（tuple）是一种容器类型，是任意对象的有序集合，通过索引访问其中的元素。

长度固定，对象不可改变。

2)元组的操作

元祖的用法几乎与列表一直，唯一不同的是元祖不可修改，这在一些特殊需求的时候有意义。只是元祖的定义中用圆括号，在初始化的时候可以用括号也可以不用。

2)用途

元组的存在看似多余但是在有些地方不可替代   
1.作为键使用的时候   
2.作为一些内建函数或者方法的返回值， 不希望被修改

## 6字符串的高级操作

1)字符串格式化

在之前我们学习print的时候已经学习了一些字符串格式化方法。其实字符串本身也可以用这种方式格式化的表达。

2)字符串函数

字符串是常用的一种对象数据类型，操作方法很多。可以通过查看帮助文档学习使用一个新的方法。

str=”hello”

str.replace:(“l”,”a”)将 “l” 替换为”a”并输出, 但不改变str的值

str.splite(“l”): 将str 以 “l” 为分隔符分开为一个列表输出,不改变str

“\*”.join([list]): 将list中的各元素通过 ”\*” 连接到一起并输出,list中的只应为字符,

str.lower():

str.upper():

str.find(‘X’): 输出 X 在 str 中第一次出现的下标位置

## 7字典

字典（dict）在其他编程语言中又称作关联数组或者散列表。   
字典与列表不同通过键来实现元素存取，是一种无序的集合，容器可变，元素也是可变的。

创建字典:

d= {1:’a’,2:’b’}

字典中常见的方法:

d.keys() 以列表的形式将d的所有键列出来

d.values() 以列表的形式将d的所有值列出来

d.items() 将每一对键值作为元组,以列表的形式输出

d.copy() 生成一个d的副本,二者的操作互不影响

d.get(I) 输出键为I对应的值

d.updata(d2) 将d2 合并到d中 如果键值重名,则覆盖

len(d) 输出字典的长度

d.pop(I) 弹出键为I的值

d.clear() 清空d

del.d[I] 删除键为I的这组数据

del d 删除整个字典

## 8集合

集合：无序排列，可哈希的值集合。集合中可包含不同数据类型的，支持成员关系的测试

集合的创建:

s={1,2}

可以使用set创建集合： set（[1,2,3,4,5]）。使用这种方式创建的集合是可变的。使用{}来创建 （如果用这个方法建立的是空，那么默认为是字典而不是集合)还可以使用frozenset来创建集合，用法和set相同，只不过创建出的集合是不可变的，可以作为字典的键值。

集合不支持索引，元素获取，切片的操作，因为集合是无序的

集合的操作:

s.add(X) 添加数据X

s.pop() 弹出第一个数据

s.clear() 清空s

s.remove(X) 删除s中的X

集合的运算:

in: 1 in s 判断1 是否在s中

-: s1 - s2 将s1中与s2重复的值去掉,输出不重复的值

| : 并集 s1 | s2: 输出s1与s2的并集

&: 交集 s1 & s2: 输出s1与s2的交集

^: 补集 s1 ^ s2: 输出s1与s2的补集

## 9数据类型总结

1.bool:只有两个值:True,False, 逻辑运算得到的结果为bool类型

2.NoneType: None 空 占位变量

3.数字类型:

整型:int,不可变, 支持数学运算,位运算等

浮点型:float

4.字符串类型:有序的, 不可变; +\*运算, 属性方法,切片操作, in逻辑

5.列表: 有序的, 可变的; +\*运算, 属性方法,切片操作, in逻辑

6.元组: 有序的, 不可变; +\*运算, 属性方法,切片操作, in逻辑

7.字典: 无序的, 可变的; 属性方法, in逻辑

8.集合: 无序的 可变的; - | & ^运算, in逻辑

9.frozenset 无序的, 不可变; - | & ^运算, in逻辑

copy:

deepcopy:

语法:

import copy

l=[1,2]

l1=l 完全复制l的属性,l改变是,l1也跟着改变;

l2=copy.copy(l) 复制子元素的属性,当对l的子元素进行操作时,l2也执行该操作.

l3=copy.deepcopy(l) l3完全脱离l,l的任何操作,对于l3没有效果

# 第三章 语句

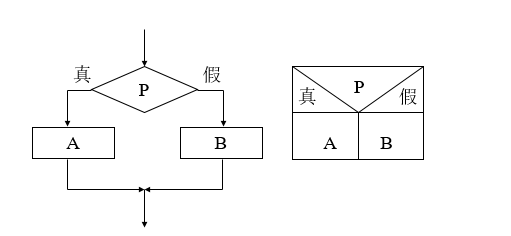
## 1.Python的缩进风格

在Python中所有的复合语句（也就是语句中嵌套语句）都有相同的一般形式，就是首行以冒号结尾，首行的下一行嵌套代码以缩进标识。

缩进的结束就是代码快的结束。在Python中用缩进的级别表示代码块。这是语法的规范，通常不同的代码快缩进为一个‘Tab’ ，而且在Python中缩进的错误为语法错误

Python语句在默认情况下是顺序执行的。

## 2.选择（分支）语句



### 2.1简单的if语句

if 逻辑判断1:

命令语句块1

### 2.2if…else…语句

if 逻辑判断1:

命令语句块1

else:

命令语句块2

### 2.3阶梯形式

if 逻辑判断1:

命令语句块1

elif 逻辑判断2:

命令语句块

else:

命令语句块3

### 2.4嵌套形式

if 逻辑判断1:

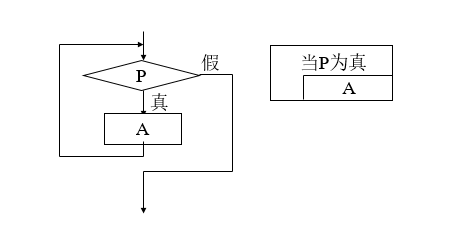
if 逻辑判断2:

命令语句块1

### 2.5if的条件运算

x if y else z 如果y为真,则返回x,否则返回z

## 3.while循环



### 3.1. while 语句格式

while 逻辑表达式1:

语句块1

每次循环首先判断while后面的逻辑表达式，如果表达式为真,则执行语句块，直到判断逻辑表达式为假终止循环。

### 3.2. while … else … 形式

while 逻辑判断1:

语句块1

else:

语句块2

只有当while语句正常结束的时候才会执行else后的语句，如果while语句被中断（外部信号或者break等）则不会执行else后的语句

## 4.for循环

### 4.1 for循环的语句形式

for 变量名 in 列表名:

语句块1

变量每次取列表中的一个值，然后执行语句块中的内容，直到列表中的值被取完为止。

for循环也有else的形式，同样只有for循环正常结束的时候才会执行else中的语句。

### 4.2 range（）生成列表

语法: 1.range(stop)

2.range(start,stop[,step])

for i in range(1,3): #从1开始直到3结束,不包括3

语句块1

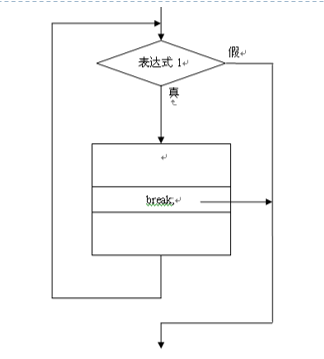
### 4.3 for 循环生成列表

[i for i in range(10)] # 生成 0-9的一个列表

## 5.辅助语句

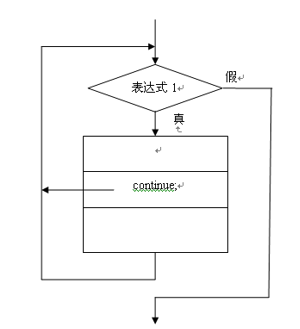
在语句控制部分还有些特殊语句用来控制语句块的执行。

### 5.1break语句

作用:从循环体内跳出循环体，即提前结束循环。   


### 5.2 continue语句

作用:结束本次循环,接着判定下一次是否执行循环。



### 5.3 pass语句

pass语句在Python中起到了一个占位的作用，当语句块中没有内容的时候语法规则是不允许什么都不写的，这时可以用pass语句占位。

### 5.4 return语句

return语句表示一个函数的结束，如果语句块在一个函数中，那么如果执行到return那么函数将会立即返回，即整个函数直接执行完毕。

## 6.zip和map

zip和map函数用来处理并行遍历。其中zip在Python2.x和python3.x版本中都支持而map是Python2.x中的函数。两者功能有相似的地方，只是map函数第一个参数希望得到一个函数变量或者None.

list1=[1,2,3]

list2=[‘a’,’b’,’c’]

list3=zip(list1,list2) #输出结果为:[(1, ‘a’), (2, ‘b’), (3, ‘c’)]

用这种方式将两个列表合成了一个列表，其中每一个元素分别是之前两个列表成对出现的。如果两个列表的长度不一样时使用zip函数会自动以较短的为准，而使用map时则会为较短的序列用None补齐。

### 用zip函数生成字典

??

## 第四章 函数

### 4.1函数的定义

函数是一个特定功能的代码模块,程序代码独立,通常有返回值,也可以是空值.

### 4.2函数的作用

将代码最大程度的重用,减少冗余

将复杂的系统分解为可管理部件,

### 内建函数

系统内部的,可以实现一定功能函数,即语法规定的函数.

查看内建函数和内置变量：

dir(\_\_builtins\_\_)

dir(\_\_builtin\_\_)

## 函数三要素

### 函数三要素

* 功能 参数 返回值

1. 功能

确定函数功能是一个函数的前提,功能,即函数最终要实现什么,决定了一个函数的复杂程度,运用什么方法,什么逻辑

2. 参数

参数,也叫参变量,也可成为形参,形参在函数调用之前不占用内存,只在调用时载入内存,参数为函数的对外接口.

3. 返回值

返回值是函数执行后以调用形式返回给调用者的结果,一个单纯功能的函数可以没有返回值,在Python中,没有返回值的函数默认返回None,如果有返回值,则用 return 标识.

## 设计函数

### 定义函数

Python 中,定义函数使用关键字 def

语法: def 函数名(参数列表):

语句块。

### 函数传参

#### 参数的传递

def func(a,b):

a += 1

b += 1

print a,b

func(1,2)

>>>2,3

#### 共享引用

1. 函数的形参变量只是在函数内起作用，这种参数的传递实际上是一种引用的共享，即把本地对象与形参变量相联系。

2. 对于不可变变量，当在函数内改变变量值时候只是让这个形参指向了一个新的对象。所以这种改变不会影响原来的本地变量的值。

3. 对于可变变量，如果在函数内修改了值，那么会对原来的引用产生影响。即原来变量的值也会发生改变。这样的做法是Python的一种机制。

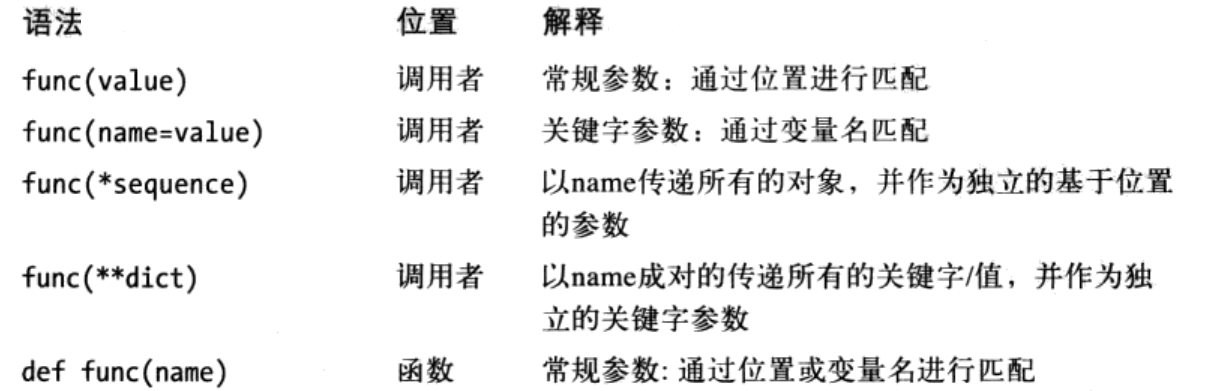
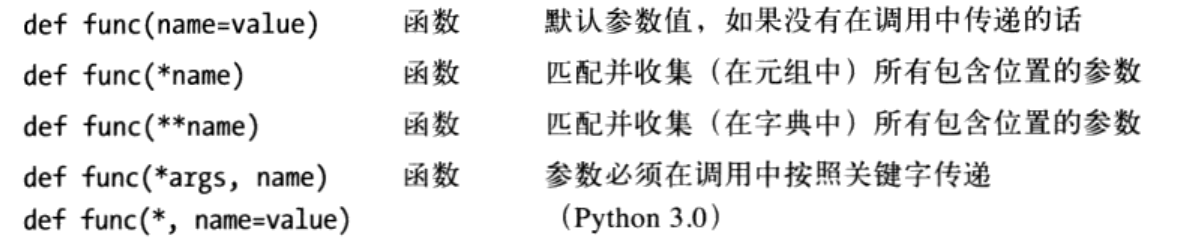
#### 避免可变参数的修改

1.直接传递可变对象的副本 a = a[:]

2.转化为不可变对象

#### 参数的匹配模式

待添加!!!!

### 函数调用和返回值

函数在调用的时候执行，并把相应的结果返回

如果函数没有返回值那么默认返回值为None。通常函数如果只是为了实现或者展示某个功能时,不需要返回值。

如果函数有返回值，用return进行返回。函数遇到return语句会结束。这时需要有个变量来接收相应的返回值。

python中可以有多个返回值，多个返回值用逗号隔开，返回结果形成一个元组，但是接收返回值的变量只能有1个。

## 作用域

当你在一个程序中使用变量名时，Python中改变或查找变量名都是在命名空间（一个保存变量的内存区域）中进行的。变量名被赋值的位置决定了这个变量名能被访问到的范围。

### 局部变量和全局变量

局部变量:  
 在函数内部定义的变量称为局部变量,在函数外定义的变量称为全局变量

1).在函数内定义的变量名只能够被函数内的代码使用，不能在函数外部引用

2) 当函数内的变量名与函数外的变量名冲突时,函数内使用的变量默认为函数中定义的变量名

### 变量的作用范围

1) 全局变量在文件范围内有效

2) 局部变量在函数内部有效

3) 类内部的本地变量在类范围内有效

4) 内嵌模块是全局作用域

变量的查找顺序:函数内(本地空间)-->类内(本地空间)-->模块内(全局空间)-->内置变量(内置空间)

## 函数的嵌套作用

### global的使用

global关键字用来在函数或者其他局部作用域中使用并可以修改全局变量,如果函数只是引用单并不对全局变量进行修改,则不需要使用global关键字.

a = 5

def func():

global a

a += 1

fun()

print(a)

### nonlocal的使用

def maker(N):

def action(X):

return X \*\* N

return action

f = maker(2)

print (f(3))

在Python中允许在一个函数内部进行函数的定义。我们有时候跟这种嵌套函数为工厂函数或者闭包。在及时生成事件处理或或者实时对不同情况进行反馈的程序中会有特别的作用。但是这种做法出现了函数的局部作用域嵌套的问题。

nonlocal关键字用来在函数或者其他作用域中使用外层(非全局)变量,只要是在函数嵌套中的内层函数使用外层函数的变量,如果不加nonlocal,可以打印,但是不能修改,添加之后即可修改:

def fun\_out():  
 a = 4

def fun\_in():

nonlocal a

a +=1

fun\_in()

print (a)

fun\_out()

## 特殊函数

### 递归函数

所谓递归函数是指一个函数的函数体中直接调用或间接调用了该函数自身的函数。这里的直接调用是指一个函数的函数体中含有调用自身的语句，间接调用是指一个函数在函数体里有调用了其它函数，而其它函数又反过来调用了该函数的情况。

递归函数调用的执行过程分为两个阶段：

递推阶段：从原问题出发，按递归公式递推从未知到已知，最终达到递归终止条件。   
回归阶段：按递归终止条件求出结果，逆向逐步代入递归公式，回归到原问题求解。

### 匿名函数

除了def语句外，Python还提供了一种生成函数对象的表达式形式，称之为lambda。这个表达式创建了一个之后能够调用的函数，它返回了这个函数，而不是将这个函数赋值给一个变量名，这也是lambda有时叫做匿名函数的原因。实际上，他们常常以一种行内进行函数定义的形式使用，或者用作推迟执行一些代码。

这里需要注意的是lambda是一个表达式而不是一个语句，它返回一个函数对象。

lambda在很多时候会使代码变得更加简洁