**Python**

1. **Python简介**
   1. **Python的历史和现状**

Python是一种面向对象、解释型、跨平台的动态脚本语言，由Guido van Rossum（荷兰）于1989年底发明。取python作该语言名字，是因为Guido本人是一个叫Monty Python的喜剧团体的爱好者

自诞生至今，它逐渐被广泛应用于处理系统管理任务和Web编程，已经成为最受欢迎的程序设计语言之一。知名的python应用有Google、Youtube、Dropbox和Instagram，国内的话则有豆瓣。

最新的python版本是3.4.0和2.7.8，语法有一定差别。鉴于目前使用较多的还是python2.x，而且linux和mac系统自带python2.x，因此建议初学者从python2.x学起。

**以下介绍均以python2.6.6为标准(Centos6系统自带版本)。**

* 1. **Python的特点**

Python的设计哲学是“优雅”、“明确”、“简单”。一个和其他大多数语言（如C）的区别就是，一个模块的界限，完全是由每行的首字符在这一行的位置来决定的（而C语言是用一对花括号{}来明确的定出模块的边界的，与字符的位置毫无关系）。这一点曾经引起过争议。因为自从C这类的语言诞生后，语言的语法含义与字符的排列方式分离开来，曾经被认为是一种程序语言的进步。不过不可否认的是，通过强制程序员们缩进（包括if，for和函数定义等所有需要使用模块的地方），Python确实使得程序更加清晰和美观。

虽然Python可能被粗略地分类为“脚本语言”，但实际上一些大规模软件开发计划例如Zope、Mnet及BitTorrent，Google也广泛地使用它。Python的支持者较喜欢称它为一种高级动态编程语言，原因是“脚本语言”泛指仅作简单程序设计任务的语言，如shellscript、VBScript等只能处理简单任务的编程语言，并不能与Python相提并论。

Python开发者的哲学是“用一种方法，最好是只有一种方法来做一件事”。

* 1. **Python的环境搭建和运行机制**
* **安装：**

Linux和Mac系统自带python解释程序，无需安装；Windows下安装python可以在官网<https://www.python.org/downloads/windows/>下载后直接运行安装。

* **运行：**

**交互式编程：**linux、mac和windows下直接在命令行的shell输入“python”回车，即可启动带提示符的python解释器。该模式下Python会在下一行立即给出输出。

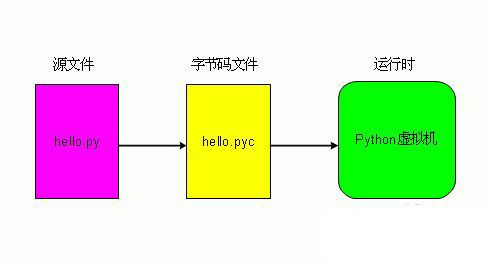
**脚本式编程：**新建一个“.py”后缀的文件，再以vi模式（Windows下使用记事本）进行编辑保存，然后使用命令“python xxx.py”解释运行。

* **IDE：**

如果你习惯使用IDE，可以使用官方的IDLE（Python GUI）；本人推荐SublimeText、PyCharm和PyDev（Eclipse插件）

* **运行机制：**

Python运行脚本的第一步，是将代码编译为“字节码”，之后将其转发到“python虚拟机”（PVM）中运行。如下图：



编译后的字节码文件会隐式存在于当前目录。所以，与PHP这种脚本语言不同，Python只在初次运行时初始化全部资源，随后的请求就不必再次初始化了。

1. **Python基础知识**
   1. **基础语法**
   2. **数据类型**
      1. **数字**
   * Python中的变量赋值前不需要事先声明，一旦赋值即创建该数值类型变量。创建一个数字类型的变量基本形式：

var1 = 306

var2 = 2.1

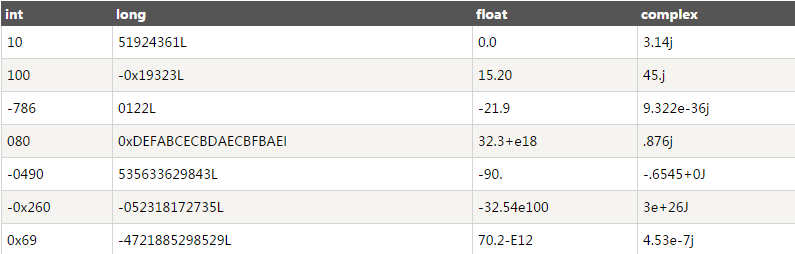
* + Python支持四种不同的数字类型：

整型(int) - 通常被称为是整型或整数，是正或负整数，不带小数点。

长整型(long) - 无限大小的整数，整数最后是一个大写或小写的L。

浮点型(float) - 浮点型由整数部分与小数部分组成，浮点型也可以使用科学计数法表示（2.5e2 = 2.5 x 102 = 250）

复数(complex) - 复数的虚部以字母J 或 j结尾 。如：2+3i



* Python内置了很多有用的数学函数，以下（部分函数使用前需导入math模块）：



* + 1. **字符串**
* 创建字符串很简单，只要为变量分配一个值即可。例如：

var1 = ‘Hello World!’

var2 = “Coremail”

* Python三引号：

python三引号允许一个字符串跨多行，字符串中可以包含换行符、制表符以及其他特殊字符。

三引号的语法是一对连续的单引号或者双引号（通常都是成对的用）。

>>> str = '''hi # 说明： “>>>”代表交互模式下的命令提示符（下同）

there'''

>>> str

'hi\nthere'

>>> print hi

hi

there

* 格式化字符串：

Python 支持格式化字符串的输出 。尽管这样可能会用到非常复杂的表达式，但最基本的用法是将一个值插入到一个有字符串格式符 %s 的字符串中。

如下实例：

print "My name is %s and weight is %d kg!" % ('ayi', 60)

以上输出结果：

My name is ayi and weight is 60 kg!

python字符串格式化符号如下:



* Unicode字符串：

Python 中定义一个 Unicode 字符串和定义一个普通字符串一样简单：

>>> u'Hello World !'

u'Hello World !'

引号前小写的"u"表示这里创建的是一个 Unicode 字符串。如果你想加入一个特殊字符，可以使用 Python 的 Unicode-Escape 编码。如下例所示：

>>> u'Hello\u0020World !'

u'Hello World !'

被替换的 \u0020 标识表示在给定位置插入编码值为 0x0020 的 Unicode 字符（空格符）。

* Python的字符串内建函数还有如下：

|  |  |
| --- | --- |
| **方法** | **描述** |
| string.capitalize() | 把字符串的第一个字符大写 |
| [string.center(width)](http://www.w3cschool.cc/python/att-string-center.html) | 返回一个原字符串居中,并使用空格填充至长度 width 的新字符串 |
| [string.count(str, beg=0, end=len(string))](http://www.w3cschool.cc/python/att-string-count.html) | 返回 str 在 string 里面出现的次数，如果 beg 或者 end 指定则返回指定范围内 str 出现的次数 |
| [string.decode(encoding='UTF-8', errors='strict')](http://www.w3cschool.cc/python/att-string-decode.html) | 以 encoding 指定的编码格式解码 string，如果出错默认报一个 ValueError 的 异 常 ， 除 非 errors 指 定 的 是 'ignore' 或 者'replace' |
| [string.encode(encoding='UTF-8', errors='strict')](http://www.w3cschool.cc/python/att-string-encode.html) | 以 encoding 指定的编码格式编码 string，如果出错默认报一个ValueError 的异常，除非 errors 指定的是'ignore'或者'replace' |
| [string.endswith(obj, beg=0, end=len(string))](http://www.w3cschool.cc/python/att-string-endswith.html) | 检查字符串是否以 obj 结束，如果beg 或者 end 指定则检查指定的范围内是否以 obj 结束，如果是，返回 True,否则返回 False. |
| [string.expandtabs(tabsize=8)](http://www.w3cschool.cc/python/att-string-expandtabs.html) | 把字符串 string 中的 tab 符号转为空格，默认的空格数 tabsize 是 8. |
| [string.find(str, beg=0, end=len(string))](http://www.w3cschool.cc/python/att-string-find.html) | 检测 str 是否包含在 string 中，如果 beg 和 end 指定范围，则检查是否包含在指定范围内，如果是返回开始的索引值，否则返回-1 |
| [string.index(str, beg=0, end=len(string))](http://www.w3cschool.cc/python/att-string-index.html) | 跟find()方法一样，只不过如果str不在 string中会报一个异常. |
| [string.isalnum()](http://www.w3cschool.cc/python/att-string-isalnum.html) | 如果 string 至少有一个字符并且所有字符都是字母或数字则返回 True,否则返回 False |
| [string.isalpha()](http://www.w3cschool.cc/python/att-string-isalpha.html) | 如果 string 至少有一个字符并且所有字符都是字母则返回 True,否则返回 False |
| [string.isdecimal()](http://www.w3cschool.cc/python/att-string-isdecimal.html) | 如果 string 只包含十进制数字则返回 True 否则返回 False. |
| [string.isdigit()](http://www.w3cschool.cc/python/att-string-isdigit.html) | 如果 string 只包含数字则返回 True 否则返回 False. |
| [string.islower()](http://www.w3cschool.cc/python/att-string-islower.html) | 如果 string 中包含至少一个区分大小写的字符，并且所有这些(区分大小写的)字符都是小写，则返回 True，否则返回 False |
| [string.isnumeric()](http://www.w3cschool.cc/python/att-string-isnumeric.html) | 如果 string 中只包含数字字符，则返回 True，否则返回 False |
| [string.isspace()](http://www.w3cschool.cc/python/att-string-isspace.html) | 如果 string 中只包含空格，则返回 True，否则返回 False. |
| [string.istitle()](http://www.w3cschool.cc/python/att-string-istitle.html) | 如果 string 是标题化的(见 title())则返回 True，否则返回 False |
| [string.isupper()](http://www.w3cschool.cc/python/att-string-isupper.html) | 如果 string 中包含至少一个区分大小写的字符，并且所有这些(区分大小写的)字符都是大写，则返回 True，否则返回 False |
| [string.join(seq)](http://www.w3cschool.cc/python/att-string-join.html) | Merges (concatenates)以 string 作为分隔符，将 seq 中所有的元素(的字符串表示)合并为一个新的字符串 |
| [string.ljust(width)](http://www.w3cschool.cc/python/att-string-ljust.html) | 返回一个原字符串左对齐,并使用空格填充至长度 width 的新字符串 |
| [string.lower()](http://www.w3cschool.cc/python/att-string-lower.html) | 转换 string 中所有大写字符为小写. |
| [string.lstrip()](http://www.w3cschool.cc/python/att-string-lstrip.html) | 截掉 string 左边的空格 |
| [string.maketrans(intab, outtab])](http://www.w3cschool.cc/python/att-string-maketrans.html) | maketrans() 方法用于创建字符映射的转换表，对于接受两个参数的最简单的调用方式，第一个参数是字符串，表示需要转换的字符，第二个参数也是字符串表示转换的目标。 |
| [max(str)](http://www.w3cschool.cc/python/att-string-max.html) | 返回字符串 *str* 中最大的字母。 |
| [min(str)](http://www.w3cschool.cc/python/att-string-min.html) | 返回字符串 *str* 中最小的字母。 |
| [string.partition(str)](http://www.w3cschool.cc/python/att-string-partition.html) | 有点像 find()和 split()的结合体,从 str 出现的第一个位置起,把 字 符 串 string 分 成 一 个 3 元 素 的 元 组 (string\_pre\_str,str,string\_post\_str),如果 string 中不包含str 则 string\_pre\_str == string. |
| [string.replace(str1, str2,  num=string.count(str1))](http://www.w3cschool.cc/python/att-string-replace.html) | 把 string 中的 str1 替换成 str2,如果 num 指定，则替换不超过 num 次. |
| [string.rfind(str, beg=0,end=len(string) )](http://www.w3cschool.cc/python/att-string-rfind.html) | 类似于 find()函数，不过是从右边开始查找. |
| [string.rindex( str, beg=0,end=len(string))](http://www.w3cschool.cc/python/att-string-rindex.html) | 类似于 index()，不过是从右边开始. |
| [string.rjust(width)](http://www.w3cschool.cc/python/att-string-rjust.html) | 返回一个原字符串右对齐,并使用空格填充至长度 width 的新字符串 |
| string.rpartition(str) | 类似于 partition()函数,不过是从右边开始查找. |
| [string.rstrip()](http://www.w3cschool.cc/python/att-string-rstrip.html) | 删除 string 字符串末尾的空格. |
| [string.split(str="", num=string.count(str))](http://www.w3cschool.cc/python/att-string-split.html) | 以 str 为分隔符切片 string，如果 num有指定值，则仅分隔 num 个子字符串 |
| [string.splitlines(num=string.count('\n'))](http://www.w3cschool.cc/python/att-string-splitlines.html) | 按照行分隔，返回一个包含各行作为元素的列表，如果 num 指定则仅切片 num 个行. |
| [string.startswith(obj, beg=0,end=len(string))](http://www.w3cschool.cc/python/att-string-startswith.html) | 检查字符串是否是以 obj 开头，是则返回 True，否则返回 False。如果beg 和 end 指定值，则在指定范围内检查. |
| [string.strip([obj])](http://www.w3cschool.cc/python/att-string-strip.html) | 在 string 上执行 lstrip()和 rstrip() |
| [string.swapcase()](http://www.w3cschool.cc/python/att-string-swapcase.html) | 翻转 string 中的大小写 |
| [string.title()](http://www.w3cschool.cc/python/att-string-title.html) | 返回"标题化"的 string,就是说所有单词都是以大写开始，其余字母均为小写(见 istitle()) |
| [string.translate(str, del="")](http://www.w3cschool.cc/python/att-string-translate.html) | 根据 str 给出的表(包含 256 个字符)转换 string 的字符,要过滤掉的字符放到 del 参数中 |
| [string.upper()](http://www.w3cschool.cc/python/att-string-upper.html) | 转换 string 中的小写字母为大写 |
| [string.zfill(width)](http://www.w3cschool.cc/python/att-string-zfill.html) | 返回长度为 width 的字符串，原字符串 string 右对齐，前面填充0 |
| [string.isdecimal()](http://www.w3cschool.cc/python/att-string-isdecimal.html) | isdecimal()方法检查字符串是否只包含十进制字符。这种方法只存在于unicode对象。 |

* + 1. **列表**
* 列表是Python中最基本的数据结构。

列表中的每个元素都分配一个索引，第一个索引是0，第二个索引是1，依此类推。列表的数据项不需要具有相同的类型

创建一个列表，只要把逗号分隔的不同的数据项使用方括号括起来即可。如下所示：

list1 = ['roger', 'km', 1990, 1991]

list2 = ["welcome", "to", "coremail"]

* 访问列表中的值：

使用下标索引来访问列表中的值，如下所示：

list1 = ['roger', 'km', 1990, 1991]

list2 = ["welcome", "to", "coremail"]

print list1[0],list2[2]

以上实例输出结果：

roger coremail

* 更新列表

你可以对列表的数据项进行重新赋值，也可以使用append()或extend()方法来添加列表项，如下示例：

#!/usr/bin/python

list1 = ['roger', 'km', 1990, 1991]

list2 = ["welcome", "to", "coremail"]

print "更新前的列表内容为 : "

print list1

print list2

print "更新后的列表内容为："

list2[2] = ‘YingShii’ #修改list2的‘coremail’为‘YingShi’

print list2

list1.append(‘lxj’) #list1末尾增加一个数据项‘lxj’

print list1

list1.append([‘ceci’,’sl’]) #list1末尾增加一个数据项（列表）：‘[‘ceci’,’sl’]’

print list1

list1.extend(‘lxj’) #list1末尾增加一个数据项‘lxj’

print list1

list1.extend([‘ceci’,’sl’]) # list1末尾增加一个数据项（列表）：‘[‘ceci’,’sl’]’

print list1

以上实例输出结果为：

更新前的列表内容为 :

['roger', 'km', 1990, 1991]

[‘welcome’, ‘to’, ‘coremail’]

更新后的列表内容为：

[‘welcome’, ‘to’, ‘YingShi’]

['roger', 'km', 1990, 1991,’lxj’]

['roger', 'km', 1990, 1991,[‘ceci’,’sl’]]

['roger', 'km', 1990, 1991,’l’,’x’,’j’]

['roger', 'km', 1990, 1991,’ceci’,’sl’]

* 删除列表元素

可以使用 del语句或remove()来删除列表的的元素，如下实例：

list1 =['roger', 'km', 1990, 1991]

print list1

del list1[2]

list1.remove(‘roger’)

print "删除列表元素之后 : "

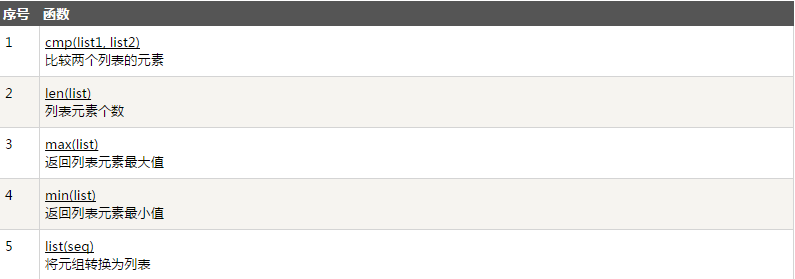
print list1

以上实例输出结果为：

['roger', 'km', 1990, 1991]

[ 'km', 1991]

* 更多列表操作的函数和方法如下表格：





* + 1. **字典**
    - 字典是另一种可变容器模型，且可存储任意类型对象，如其他容器模型。字典由键和对应值成对组成，类似于PHP关联数组或C语言哈希表。基本语法如下：

dict1 ={‘ayi’:1,’ceci’:2,’roger’:3,’sl’:4}

每个键与值用冒号隔开，每对用逗号分割，整体放在花括号中。

键必须独一无二，但值则不必。值可以取任何数据类型，但必须是不可变的，如字符串，数或元组。

* + - 访问字典里的值

根据键（key）访问字典里相应的值（value），如下实例：

dict1 ={‘ayi’:1,’ceci’:2,’roger’:3,’sl’:4}

print dict1[‘ayi’],dict1[‘sl’]

以上实例输出结果：

1 4

* + - 修改字典元:素

向字典添加新内容的方法是增加新的键/值对，修改或删除已有键/值对如下实例:

dict = {'Name': 'Lara', 'Age': 7, 'Class': 'First'}

dict['Age'] = 8 #修改已有元素

dict['School'] = "DPS School" #增加新元素

print "dict['Age']: ", dict['Age']

print "dict['School']: ", dict['School']

以上实例输出结果：

dict['Age']: 8

dict['School']: DPS School

* + - 删除字典元素

Del语句能删单一的元素，使用clear()方法可以清空字典。如下实例：

dict = {'Name': 'Zara', 'Age': 7, 'Class': 'First'}

del dict['Name'] # 删除键是'Name'的条目

print dict

dict.clear() # 清空词典所有条目

print dict

以上实例输出结果：

{'Age': 7, 'Class': 'First'}

{}

* + - 字典内置函数和方法

Python字典包含了以下内置函数：





* + 1. **元组**
       - 元组是类似于列表的序列数据类型，用圆括号“（）”括起来。

它和列表之间的主要区别是：列表中元素的值是可以改变的，而元组的元素则不能被更新，所以元组可以被认为是只读列表。

元组使用小括号，列表使用方括号。元组的创建很简单，只需要在括号中添加元素，并使用逗号隔开即可。如下实例：

tup1 = ('roger', 'km', 1990, 1991)

tup2 = (1, 2, 3, 4, 5 )

tup3 = () #空元组

tup1 = (1,) #注意：当元组只包含一个元素时，需要在元素后面添加逗号

* + - * 访问和修改元组

访问元组和访问列表的语法类似，如下：

>>> tup1 = ('roger', 'km', 1990, 1991)

>>> print tup1[0] #下标索引也是从0开始

roger

而元组的元素值是不可以修改的，只能进行元组的拼接组合，或者你可以使用del语句删除整个元组：

tup1 = (12, 34.56)

tup2 = ('abc', 'xyz')

# tup1[0] = 100 # 修改元组元素操作是非法的。

tup3 = tup1 + tup2 # 组合成一个新的元组

print tup3

del tup1

以上实例输出结果为：

(12, 34.56, 'abc', 'xyz')

* + - * 元组的内置函数



* + 1. **类型转换和索引切片**
       - Python的类型转换函数有如下：



* + - * Python还有很实用的切片功能，可以对序列和元组进行切割，或者实现更高级的功能。

从原理上分析切片运算：list的切片，内部是调用\_\_getitem\_\_，\_\_setitem\_\_,\_\_delitem\_\_和slice函数，而slice函数又是和range()函数相关的。给切片传递的键是一个特殊的slice对象，该对象拥有可描述所请求切片方位的属性，例如：

a = [ 1, 2, 3, 4, 5, 6 ]

x = a [ 1 : 5 ] # 等同于x = a.\_\_getitem\_\_( slice ( 1, 5, None ) )，获取序列的元素a[1],a[2],a[3],a[4],但不包含a[5]

print x

a [ 1 : 3 ] = [10, 11, 12 ] #等同于a.\_\_setitem\_\_( slice ( 1, 3, None ), [ 10, 11, 12 ] )，**赋予的元素数可以多于切片长度（慎用）**

print a

del a [ 1 : 4 ]# a.\_\_delitem\_\_( slice ( 1, 4, None ) )

print a

以上实例输出结果是：

[2, 3, 4, 5]

[1, 10, 11, 12, 4, 5, 6]

[1, 4, 5, 6]

还可以给切割加上步进值（step），如：

a = [ 1, 2, 3, 4, 5, 6 ]

x = a[0:6:2] #切片的步进值为2

print x

以上实例输出结果：

[1,3,5]

步进值可以是负数哦！有一个经典应用是：翻转字符串。如下：

>>> str='coremail.com'

>>> str[::-1] #步进值为-1

'moc.liameroc'

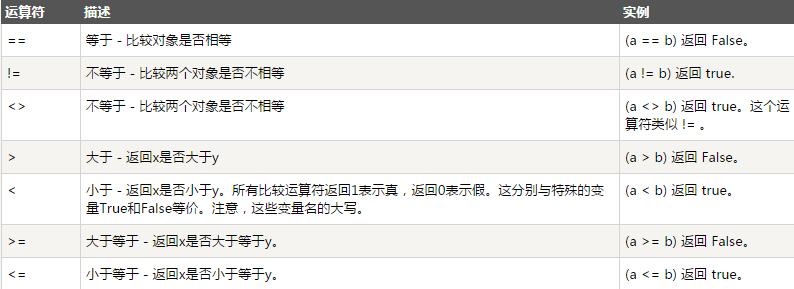
* 1. **运算符**
     1. **算术运算符**

以下假设变量a为10，变量b为20：



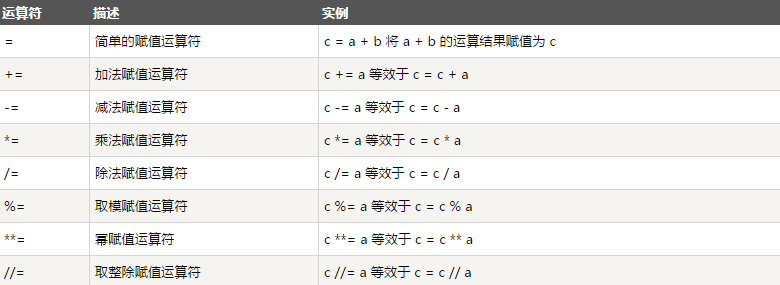
* + 1. **比较运算符**

以下假设变量a为10，变量b为20：



* + 1. **赋值运算符**

以下假设变量a为10，变量b为20：



* + 1. **位运算符**

按位运算符是把数字看作二进制来进行计算的。以下实例演示了Python所有位运算符的操作：

a = 60 # 60 = 0011 1100

b = 13 # 13 = 0000 1101

c = 0

c = a & b; # 12 = 0000 1100

print "Line 1 - Value of c is ", c

c = a | b; # 61 = 0011 1101

print "Line 2 - Value of c is ", c

c = a ^ b; # 49 = 0011 0001

print "Line 3 - Value of c is ", c

c = ~a; # -61 = 1100 0011

print "Line 4 - Value of c is ", c

c = a << 2; # 240 = 1111 0000

print "Line 5 - Value of c is ", c

c = a >> 2; # 15 = 0000 1111

print "Line 6 - Value of c is ", c

以上实例输出结果：

Line 1 - Value of c is 12

Line 2 - Value of c is 61

Line 3 - Value of c is 49

Line 4 - Value of c is -61

Line 5 - Value of c is 240

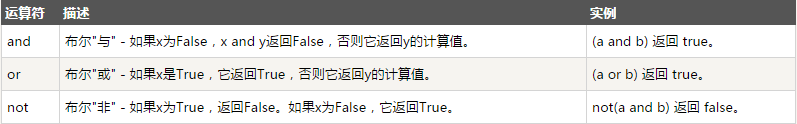
Line 6 - Value of c is 15

Python中的按位运算法则总结如下表（以上述程序的a=60,b=13为例）：



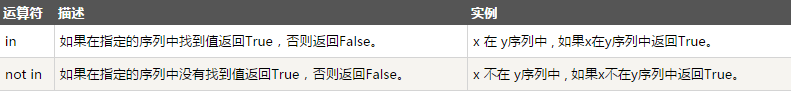
* + 1. **逻辑运算符**

Python语言支持逻辑运算符，以下假设变量a为10，变量b为20：



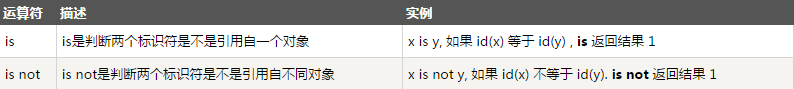
* + 1. **成员运算符**

Python还支持成员运算符，运算对象包括字符串，列表或元组：



* + 1. **身份运算符**

身份运算符用于比较两个对象的存储单元：



* + 1. **优先级**

以下表格列出了从最高到最低优先级的所有运算符：



* 1. **控制语句**
     1. **条件语句**

Python 编程中 if 语句用于控制程序的执行，基本形式为（else语句非必备）：

　if 判断条件：

执行语句……

　else：

执行语句……

Python程序语言指定任何非0和非空（null）值为true，0 或者 null为false。

if-else语句当然也可以嵌套使用，形式如下：

if 判断条件1:

执行语句1……

elif 判断条件2:

执行语句2……

elif 判断条件3:

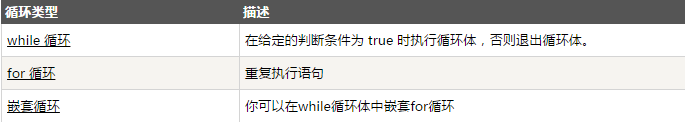
执行语句3……

else:

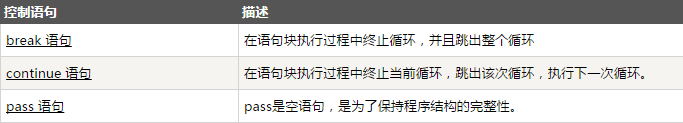
执行语句4……

* + 1. **循环语句**

Python提供了for循环和while循环（在Python中没有do..while循环）:



循环控制语句可以更改语句执行的顺序。Python支持以下循环控制语句：



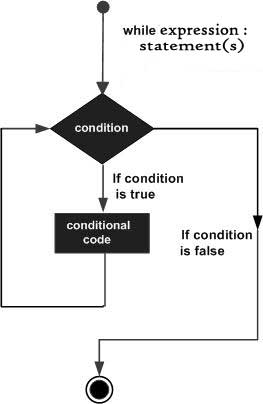
* + - 1. **While循环**

while 语句用于循环执行程序，即在某条件下，循环执行某段程序，以处理需要重复处理的相同任务。其基本形式为：

while 判断条件：

执行语句……

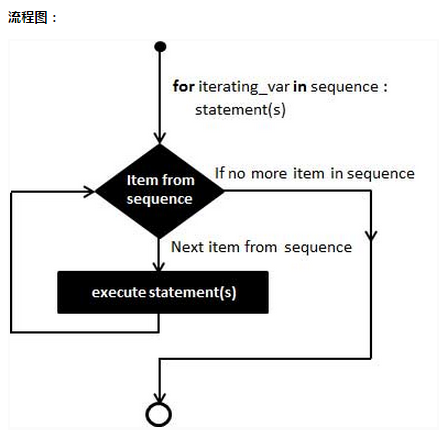
执行语句可以是单个语句或语句块。判断条件可以是任何表达式，任何非零、或非空（null）的值均为true。



* + - 1. **For循环**
      * for循环可以遍历任何序列的项目，如一个列表或者一个字符串。for循环的语法格式如下：

for var in sequence:

statements(s)



* + - * 另外一种执行循环的遍历方式是通过索引，如下实例：

names = ['km', 'roger','ayi']

for index in range(len(names)): #len(names) == 3，这里用range(3)也是一样的

print 'Current name :', names[index]

print "They are douB!23333"

以上实例输出结果为：

Current name: km

Current name: roger

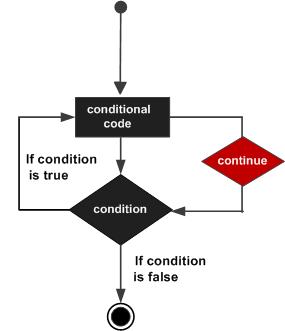
Current name: ayi

They are douB!23333

* + - 1. **循环控制**
      * **Continue语句**

continue 语句用来告诉Python跳过当前循环的剩余语句，然后继续进行下一轮循环；用在while和for循环中。

流程示意图：



看如下一个实例：

names = ['km','ayi','roger']

for index in range(len(names)):

if names[index] == 'ayi':

continue

print 'Current name :', names[index]

print 23333

以上实例输出结果为：

Current name : km

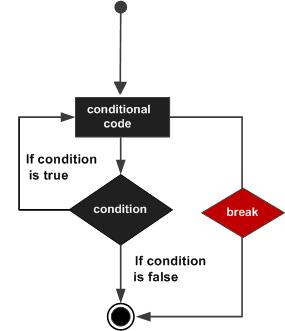
Current name : roger

23333

* + - * **Break语句**

break语句用来终止循环语句，即循环条件没有False条件或者序列还没被完全递归完，也会停止执行循环语句。如果您使用嵌套循环，break语句将停止执行最深层的循环，并开始执行下一行代码。

流程图：



看如下一个实例：

names = ['km','ayi','roger']

for index in range(len(names)):

if names[index] == 'ayi':

break

print 'Current name :', names[index]

print 23333

以上实例输出结果为：

Current name : km

23333

* + - * **Pass语句**

pass是空语句，是为了保持程序结构的完整性。

看如下一个实例：

names = ['km','ayi','roger']

for index in range(len(names)):

if names[index] == 'ayi':

pass

else:

print 'Current name :', names[index]

print 23333

以上实例输出结果为：

Current name : km

Current name : roger

23333

* 1. **函数**
     1. 函数的定义和调用
     2. 参数传递
     3. 匿名函数
     4. 变量作用域和命名空间
     5. 常用函数
  2. 模块和包
     1. 引入模块
     2. 定位模块
     3. 常用模块

1. 文件IO和异常处理
   1. 屏幕输出与键盘输入
   2. 文件的打开和关闭
   3. 文件的修改
   4. 异常机制
2. Python高级编程
   1. 面向对象
   2. 正则表达式
   3. 字符编码
   4. CGI编程
   5. GUI编程
   6. 函数编程
   7. 数据库编程
   8. 多线程编程
   9. XML解析
   10. HTML处理
   11. 脚本和流
   12. Web服务
   13. 性能优化和内存管理