# [如何学习一门新的语言](http://www.cnblogs.com/chgaowei/archive/2011/05/29/2061902.html)

这两天在看《编程人生》，这本书确实非常不错。而且看得也特别的轻松。其中有几个 人都谈到了如何学习新的语言，但是给我最深刻的是google的首席java架构师joshua bloch。正好最近我也在学习python，所以顺便总结一下如何学习一门新的语言。希望你能补充一些。

**心态**

这不但是学习一门新的语言最重要的，而是对任何的学习都是最重要的。下面是书中的描述，非常的精彩，特别是那个比喻：

“学 习一门新的语言的时候，要利用以前所学的语言的功底，但是也要保持开放的心态。有些人执着于一种理念：“这就是写所有程序必须遵循的方法”。我不是说那种 语言，但是某些语言，令人执着于这样的理念。当开始学习新语言的时候，他们会批评这种语言跟真正神的语言的所有的不同之处。当使用新语言时，他们极力使用 神的语言的方法去写。这样，你就会错过这个新语言真正的独特之处。   
这就像你本来只有一个榔头，有人给了你一个螺丝刀，你说“哎，这不是一把好榔头，但是我应该可以倒着拿螺丝刀，用螺丝刀来砸东西。”你得到了一个很烂的榔头，但事实上它确实一把很不错的螺丝刀。所以你应该对所有的事物保持开放和积极的心态。”

如果你的杯子满了，那他永远再也装不进水了。如果你认为你找到了银弹，那么你可能就要固步自封了。

对新的事物，方法保持一个开发而积极的心态，才能真正了解他，了解他的独特之处。

**了解他的历史，哲学观**

选择一门语言，往往选择了一种思维方式和哲学观。所以，了解一门语言的历史和哲学观非常重要。你要知道这门语言是谁创建的，为什么创建，如何发展起来的，适合那些领域，以及解决问题的哲学是什么。

那python来说，他的设计哲学是“用一种方法，最好是只有一种方法来做一件事”，而perl的设计哲学是“总有多种方法来做同一件事”。所以，我选择的是python。

了解这方面的知识的一个非常好的来源是百科网站。

**代码，代码，还是代码**

代码是学习一门语言的必经之路，可能也是最快的一种方法。

你不但要找一些优秀的代码来阅读，还要亲自动手来写代码。这个过程对学习语言来说是非常快的。另外，你一定要用语言去解决实际的问题，而不仅仅是写代码来验证语法。在解决问题的过程中，你可以学习它是如何解决问题的，而且会积累语言的经验。

**社区**

多去这个语言的社区逛逛吧，这里有很多人在讨论这种语言，和他们一起讨论你能够学到更多。

# [Python入门笔记：学习目录](http://www.cnblogs.com/BeginMan/archive/2013/03/11/2954819.html)

一、我为什么要学Python?

1.兴趣、好奇心

俗话说：“兴趣是最好的老师！”，有兴趣，好奇心，才能支撑着我从简到难地去学习Python语言。

2.Python的发展前景

2012--2013年编程语言排行榜上地位显赫，且年纪轻轻就如此登峰造极，可见非同小可，何况在国外她已经很火了……而国内经常性的啃别人剩下的馍，对于Python在国内的发展，不可小觑啊。

二、学习参考

1.《Dive Into Python》

2.《Core Python Programming》

3. Python官方文档

三、学习目录

[ 一、Python初级 ]

1.Python概述

2.Python基础(上篇)

3.Python基础(下篇)

4.Python对象(上篇)

5.Python对象(下篇)

6.Python数字

7.Python序列(上篇)

8.Python序列(中篇)

9.Python序列(下篇)

10.Python映像和集合(上篇)

11.Python映像和集合(下篇)

12.Python条件和循环(上篇)

13.Python条件和循环(下篇)

14.Python文件输入输出

15.Python错误和异常

16.Python函数

17.Python模块

18.PythonOOP

19.Python小结

[ 二、Python高级 ]

待续……

# [Python入门笔记(1)：Python概述](http://www.cnblogs.com/BeginMan/archive/2013/03/11/2954830.html)

一、Python特点

Python发展了十五年了，现在正是少年得意之时。它的特点如下：

1.高级编程语言

2.面向对象

3.可升级

所谓的可升级也就是说重用、提升、增强和完善你的代码，远远凌驾于Unix的shell或批处理。在《core python programming》解释：“可升级”这个术语用于衡量硬件的负载，为系统添加新硬件后带来性能的提升；我们试图用于软件，指的是Python提供了 基本的开发模块，你可以在她上面开发你的软件，而且这些需要扩展和增长时，Python的可插入性和模块化架构则是你的项目生机盎然，便于管理。

4.可扩展

怎么扩展：分多文件或多模块，分部处理，同时便于管理；c/c++编写(因为Python的标准实现是使用C语言完成，也称CPython，故用c /c++编写Python扩展)编写Python扩展。Python的java实现被称作Jython,还有IronPython针对.net或Mono 平台的C#实现。

5.可移植性

以为它是C写的，因此具有C的可移植性。

6.简单易学

Python的总体感觉是：关键字少、结构简单、便于阅读

7.健壮性和易维护性

异常处理机制，内存管理由Python解释器负责，这样程序员才不会纠结于内存管理问题，而是致力开发任务。

8.丰富性

各种API、标准库、扩展库和开发插件、工具等，使得Python开发高效、丰富

9.解释性和编译性

Python是一种解释型语言，没有编译这一项。Python实际上是字节编译，其结果生成一个近似机器语言的中间形式；

Python源文件用.py扩展名，当源文件被解释器加载或者显式地进行字节码编译时会被编译成字节码，由于调用解释器的方式不同，源文件会被编译成带有.pyc或pyo扩展名文件。

二、Python和Django环境搭建

见另一篇博文讲解:<http://www.cnblogs.com/BeginMan/archive/2013/03/11/2954346.html>

三、运行Python的三种方法：

1.最简单 的一种就是通过交互式的启动解释器，每输入一行便执行一行如(Python command line)

2.运行Python脚本，调用相关的脚本解释器

3.集成开发环境的图形用户界面，如：Python GUI、eclipse pydev

图形用户[界面](http://baike.baidu.com/view/240704.htm)（Graphical User Interface，简称 GUI，又称图形[用户接口](http://baike.baidu.com/view/1309799.htm)）是指采用图形方式显示的计算机操作用户[界面](http://baike.baidu.com/view/240704.htm)。与早期计算机使用的[命令行界面](http://baike.baidu.com/view/5483485.htm)相比，图形界面对于用户来说在视觉上更易于接受。

**动态类型与静态类型的编程语言：**

编程语言分为动态(类型)语言和静态(类型)语言。

动态语言是指：在运行期间才去做数据类型检查的语言，也就是说，在用动态类型的语言编程时，永远也不用给任何变量指定数据类型，该语言会在第一次赋值给变量时，在内部将数据类型记录下来。Python、Ruby、js就是典型的动态类型语言。

静态类型语言与动态类型语言刚刚相反：它的数据类型是在编译期间检查的，也就是说写程序时要声明所有变量的数据类型，C/C++是静态类型语言的典型代表，其他的静态类型语言有C#、Java等。

# [Python入门笔记(2)：基础(上)](http://www.cnblogs.com/BeginMan/archive/2013/04/11/3015028.html)

一、欲练此功，必先自宫

从现在起，就开始真正地接触她了，或许很多人像我一样，不知道如何去快速掌握一门新的编程语言，今天贴出来一些建议，方便大家看看，这也是学习之前 的十分重要的事情。虽然“欲练此功，必先自宫”太夸张，但是在学习之前，肯定要做好准备了。由于工作中使用Python，在工作前一个月就开始学习并投入 到实际项目中，时隔一个月的学习，今天重温一遍。我做的准备如下：

**1.工具准备**

这个在上一章学习笔记中已经做好了

**2.资料准备**

《Python 核心编程》  
 Python官方文档  
博客园、CSDN等社区

**3.入门教程准备**

大约在六月份之前完成Python入门笔记。

**4.心理准备**

关于如何学好一门新的编程语言，这里推荐阅读：http://www.cnblogs.com/chgaowei/archive/2011/05/29/2061902.html

**5.知识框架拆分准备**

通过对整个Python学习过程中的框架进行拆分，利用《Python核心编程》一书中，几乎涵盖了所有Python知识点，但是对于像我一样的新 手来说，我还是难以掌握她的整个学习框架、流程。如果我们大致看了这一本书，就能细分她的框架，分成模块来学，这样才能做到心中有数。下面是我拆分的框 架：(由于我们先从Python基础学习，所以这里不考虑高级部分)

[1]、Python基础部分：

(1).语法基础、(2)、数据类型、(3).Python对象、(4).字符串、元祖和列表、(5).映射和集合、(6).函数

[2]、流程控制

条件和循环

[3]、文件操作

[4]、模块

[5]、面向对象

[6]、异常处理

[7]、环境部署

好了，这些准备都到位了，现在就开始学习吧，没事的时候建议看看Python哲学

二、Python语法

我不知道自己之前是如何写第一个Python程序的，貌似是照着《Dive Into Python》这本令人在之前学起来蛋疼的教程中写的。一如既往地，咱们从"helloworld"开始看吧，哈哈，我们学的很多语言都是从helloworld开始的。

[复制代码](javascript:void(0);)

>>> 1 + 1

2

>>> print 'hello world'

hello world

>>> x = 1

>>> y = 2

>>> x + y

3

[复制代码](javascript:void(0);)

这里例子并不能说明什么语法，但是能让我们熟悉一些，比如，**Python 既是 动态类型定义语言 (因为它不使用显示数据类型声明) , 又是 强类型定义语言 (因为一旦一个变量具有一个数据类型, 它实际上就一直是这个类型了) 、一句程序结束后并不不以分号结尾、print 可以输出字符串等。那下面就细致讲解下python语法吧**

**1、注释**

和很多Unix脚本类似，Python的注释语句从#字符开始，如：

# print "hello world"

注意Python的#字符注释只能实现单行注释，或许很多人和我一样好奇，如果要注释掉一百多行代码，那么是不是敲一百多次#呢？？毋庸置疑，如果我们这样做，只能说明还是打酱油的雇的帮打酱油的人。

我们可以有如下小技巧：

(1).程序控制法：

在不影响整个程序的情况下，我们可以把一大堆要注释掉的代码段，利用if条件句进行间接注释，也就是if屏蔽了要注释的代码块的执行、可以使用 if False：、if None：、if 0：进行屏蔽处理，如下：

[复制代码](javascript:void(0);)

1 在eclipse中:

2 print 'BeginMan' #输出

3 if False: #屏蔽，if 后还可以是 0、None等一切布尔值为假的条件

4 print 'hello world'

5 print 'good man'

6 print 'sssss'

7 print 'ok!' #输出

[复制代码](javascript:void(0);)

则if 下的代码不会执行，也就没输出了。但是这个技巧呢，最好不要用，可能在很多代码存在的情况下存在潜伏的异常。

(2).编辑器技巧

如果在eclipse下，选中多行，可以使用快捷键ctrl+/注释或取消注释，这个强烈推荐，真是“工欲善其事，必先利其器”啊，对于编辑器，最好是掌握常用的快捷键，才能提高开发效率。

(3).doc string (文档字符串)

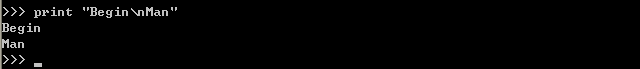
1 def foo(params):

2 """print a,b,c and return a """

三重引号(包含单引号又包含双引号)表示一个多行字符串。在开始与结束引号间的所有东西都被视为单个字符串的一部分, 包括硬回车和其它的引号字符。在三重引号中的任何东西都是这个函数的 doc string, 它们用来说明函数可以做什么。 如果存在 doc string, 它必须是一个函数要定义的第一个内容( 也就是说, 在冒号后面的第一个内容 )。 在技术上不要求给出函数的 doc string, 但应该这样做。 Python 带来一些额外的动机： doc string 在运行时可作为函数的属性。

**2、换行(\n)**

常见的换行符，在学其他编程语言中经常看到。



**3、反斜线(\)继续上一行**

也就是对一行一个语句的隔行，针对一些过长的语句可使用反斜杠\来分解成多行，如：

1 if a>0 and \

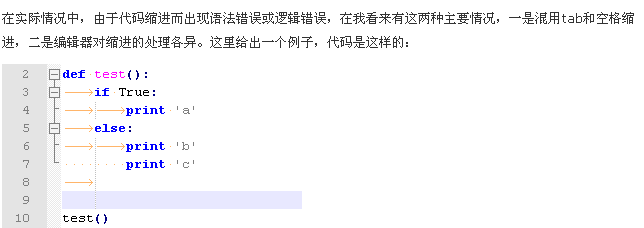
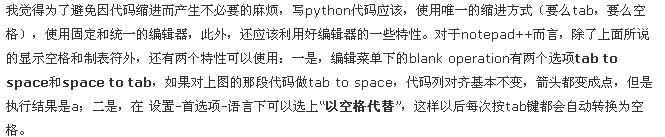
2 b<0 :

3 print 'ok!'

有两种例外情况可以不用反斜杠，一种是：小括号()、中括号[]、花括号{}；另外一种是三引号。针对可读性，不建议使用反斜杠。

**4、代码缩进**

Python 函数没有明显的 begin 和 end, 没有花括号, 用于标函数的开始和结束。 唯一的分隔符是一个冒号 (:), 接着代码本身是缩进的。例子可以参考上面的。注意，码缩进十分严格，如果不按规律办事，不小心的话就会出现语法错误，比如unexpected indent之类的。甚至有时也会出现逻辑错误。

参考：http://www.cnblogs.com/tt-0411/archive/2011/11/11/2245693.html  
  


**5、使用分号(;)可以在同一行书写多个语句**

如；import os; a=123; print a

注意句子末尾不要带;建议不要这样写，因为会大大降低代码的可读性。

**6、模块**

每个python脚本看做一个模块，磁盘的形式存放，可多次拆分成多个模块。使用import 导入。

三、Python的变量

注意，与C#、Java、C/C++等编程语言不一样，Python的变量没有类型，它的类型是在内存中根据对象所确定的。由于涉及到Python函数部分，在本节先不讲，在后面的部分会有讲解。这一节就首先学学变量赋值。

**Python是弱类型，无需明确指定变量类型，赋值的同时会自动声明类型**。如：

[复制代码](javascript:void(0);)

1 >>> anInt=10

2 >>> astr="Car"

3 >>> aFloat=-1.2+3

4 >>> aList=[1,2,3]

5 >>> aTuple=('aa','ss')

6 >>> print anInt,astr,aFloat,aList,aTuple

7 10 Car 1.8 [1, 2, 3] ('aa', 'ss')

[复制代码](javascript:void(0);)

我们需要注意以下几点：

1、在Python中是通过对象的引用而不是值来赋值给变量的。

2、赋值操作符主要是"="，同时也可使用增量赋值，如 x+=1。但是没有自增、自减操作符。

3、在C语言中，赋值语句可以当作表达式(可以返回值)，但是在Python中赋值语句不会返回值，如下面的就是非法的：

>>> x=1

>>> y=(x=x+1)

SyntaxError: invalid syntax

4、Python支持链式赋值、多重赋值、多元赋值

(1).链式赋值:

>>> y=x=x+1

>>> x,y

(2, 2)

(2).多重赋值

>>> x=y=z=1

>>> x,y,z

(1, 1, 1)

值为1的整型对象被创建，该对象的同一引用被赋值给了x,y,z

(3).多元赋值

为什么叫做多元赋值，在Python核心编程书中说，是作者自己其的名称，因为等号两边对象都是元祖。

>>> x,y,z=1,'ssss',True

>>> x,y,z

(1, 'ssss', True)

x,y,z=1,'as',3

(x,y,z)=1,'as',3

(x,y,z)=(1,'as',3) #推荐

其中，Python变量交换实在太优雅了，如果交换x和y的值，在其他语言，如C，我们或许用一个临时变量来完成工作。但是在Python中，你只需这样做就行了

>>> x,y=y,x

>>> x,y

('ssss', 1)

四、标识符

首先变量的命名规则跟其他主流语言如C一样，其次，Python关键字、标识符在Python文档中都有，必要时可翻阅。

 五、编码风格

[复制代码](javascript:void(0);)

1 #coding=utf-8 #编码格式

2 '''

3 Created on 2013-3-11

4 @author: BeginMan

5 '''

6 "this is a test module" #模块文档

7 import os #导入模块

8 import sys

9 s=True #全局变量(globalvariable)

10 class FooClass(object): #类的定义

11 "fooclass test" #类的文档对象

12 classAttribute=10 #类属性

13 def myFunction(self): #函数定义

14 "function test" #函数的文档对象

15 dataAttribute="BeginMan" #数据属性(实例属性)

16 pass

17

18 if \_\_name\_\_=="\_\_main\_\_": #主程序

19 inst=FooClass()

20 pass

21

[复制代码](javascript:void(0);)

 这里我们重点是Python的主程序。那么什么是Python的主程序呢，就是说无论这个模块是被别的模块导入还是作为脚本直接执行，都会执行这部分代码。通常这里不会有太多功能性代码，而是执行模式调用不同函数。

时刻记住一个事实，所有模块都有能力来执行代码，最高级别的Python语句，即没有缩进的代码，在模块被导入时就会执行。

其他：

如：垃圾回收机制，使我们更加专注于代码。

# [Python入门笔记(3)：基础(下)](http://www.cnblogs.com/BeginMan/archive/2013/04/12/3016323.html)

一、数字

在看《Python 核心编程》的时候，我就有点疑问，为什么没有把Python数字放在基础范围之内呢，可能作者先要通过Python对象，然后在深入到数字，这样会更加容易理解。这里有一个观点就是**“Python的一切皆对象”原则**，通过上一节的学习变量赋值，就能深刻体现这个原则，具体见下节的学习入门笔记。

由于数字在Python中用到的并不很多，就像学js的数字类型一样，我觉得最好放在基础篇之内，这样对于下节的对象有个铺垫的作用。

本节主要是学习Python的数字类型、操作、与数字有关的内建函数。

**1、数字类型**

 在上节中说，Python变量无须定义类型，这是由于它是动态类型的语言，在内存中根据所赋值动态地确定其类型。如果偏偏要在python中给变量分类的话，那么可分：**可变类型和不可变类型**。

这里整理了可变类型(mutable)和不可变类型(immutable)的相关知识：  
(或许我们在 学习了下一节才能对此有深刻理解，如果曾学习过C，它像C指针的艺术；如果学了C#，它像C#值类型与引用类型的性质；如果学了js，它的性质基本上同 Python；总的来说，一切的编程语言都围绕着基础核心，只是各自体现不一样而已，我们要学会融会贯通，举一反三。。。不过感觉好难哦。)

这里推荐一篇文章：[《如何学会600多种编程语言》](http://www.csdn.net/article/2013-03-20/2814575-programming-languages) -----------------------------------------------------------------------------------

不 可变类型(immutable):就是内存中的那可内容(value)是否可以被改变，如果是不可变则，在对对象本身操作的时候，必须在内存的另外地方再 申请一块区域(因为老区域#不可变#)，老的区域就丢弃了(如果还有其他ref，则ref数字减1，类似unix下的hard-link)。  
  
可变类型(mutable):对对象操作的时候，不需要再在其他地方申请内存，只需要在此对象后面连续申请(+/-)即可，也就是它的address会保持不变，但区域会变长或者变短。  
  
不可变类型有:string、interger、tuple：

[复制代码](javascript:void(0);)

1 >>> var1 = 1

2 >>> var2 = var1

3 >>> var1,var2

4 (1, 1)

5 >>> id(var1),id(var2) #同指同一内存区域，id相同

6 (21200672, 21200672)

7 >>> var1 += 1

8 >>> var1,var2

9 (2, 1)

10 >>> id(var1),id(var2)  
#由于var1,var2数据不可变的，对var1+1后，申请另外一块区域，id(var1)变了，id(var2)不变,对此区域ref数字-1

11 (21200660, 21200672)

12 >>>

[复制代码](javascript:void(0);)

可变类型：list,dict

[复制代码](javascript:void(0);)

1 >>> lst1 = [1,2,3]

2 >>> lst2 = lst1

3 >>> lst1,lst2

4 ([1, 2, 3], [1, 2, 3])

5 >>> id(lst1),id(lst2)

6 (28497360, 28497360)

7 >>> lst1.append(4)#可变类型的，append后，address会保持不变

8 >>> lst1,lst2

9 ([1, 2, 3, 4], [1, 2, 3, 4])

10 >>> id(lst1),id(lst2)#lst1,lst2一直指向同一区域

11 (28497360, 28497360)

12 >>>

13 在操作这种类型的时候要特别小心，dict的key不用可变类型的，函数传参的时候不用可变类型的。

14 eg:

15 >>> def dis(arg=[]):

16 ... arg.append('1')

17 ... print arg

18 ...

19 >>> dis()

20 ['1']

21 >>> dis()

22 ['1', '1']

23 >>>id(dis())

24 ['1', '1', '1']

25 505246615

26 >>>id(dis())

27 ['1','1','1','1']

28 505246615

[复制代码](javascript:void(0);)

参考：http://blog.chinaunix.net/uid-26249349-id-3080279.html

数字就是不可变类型，也就是说更改数字的值就会产生新的对象，当然这个我们无须考虑。

python数字类型可分为：整型、长整型、布尔型、双精度浮点型、十进制浮点型和复数

**(1)、整型：**

如：0101、8、-3、0X80、-0X78

**(2)、布尔型**

True、False

**(3)、长整型**

在整型后面加L(或l)，目前整型和长整型在慢慢统一。只有对它调用repr()函数才有机会看到L，调用str()函数则看不到L：

[复制代码](javascript:void(0);)

>>> aLong=11111111111111111111

>>> aLong (在命令行中，直接输入变量名显示值，这种情况实际上调用了repr()函数)

11111111111111111111L

>>> print aLong （在命令行中，如果在变量前面使用print 则调用str()函数）

11111111111111111111

>>>

[复制代码](javascript:void(0);)

**(4)、双精度浮点型**

类似C中的double。实际精度依赖Python解释器的编译器。

**(5)、复数**

[复制代码](javascript:void(0);)

1 >>> aComplex=2.1+3j

2 >>> aComplex

3 (2.1000000000000001+3j)

4 >>> aComplex.real

5 2.1000000000000001

6 >>> aComplex.imag

7 3.0

8 >>> aComplex.conjugate()

9 (2.1000000000000001-3j)

10 >>>

[复制代码](javascript:void(0);)

更多见python core programming

**2、操作符**

分别是：+、-、\*、/、%、\*\*、//（地板除）

很有意思的是，python的除法运算。

在js中实现以下代码：

[复制代码](javascript:void(0);)

1 <script type="text/javascript">

2 var a=1,b=2;

3 var c=0.1,d=0.2;

4 var result1=a/b;

5 var result2=c/d;

6 alert(result1);#0.5

7 alert(result2);#0.5

8 </script>

[复制代码](javascript:void(0);)

如果我们在Python中实现，则如下:

>>>1/2

0

>>>0.1/0.2

0.5

有些奇怪吧，Python是这样处理的；

**传统除法**：如果是整型除法，则传统除法会舍弃小数点部分，返回一个整型(地板除)；如果操作数之一是浮点型，则执行真正地除法。

**真正除法**：返回真实的商，在未来Python版本中会成为统一标准，不过现在我们通过from \_\_future\_\_ import division可以实现。

[复制代码](javascript:void(0);)

>>> from \_\_future\_\_ import division

>>> 1/2

0.5

>>> 1.0/2.0

0.5

[复制代码](javascript:void(0);)

**地板除**：从Python2.2开始，新操作符//，被称为地板除、即不管操作数是什么类型，总会舍弃小数部分，返回整数。

[复制代码](javascript:void(0);)

>>> 1//2

0

>>> 1.0//2.0

0.0

>>> -1//2

-1

[复制代码](javascript:void(0);)

其他操作符有兴趣可以在《Python 核心编程》和官网文档看看。

**3、与数字有关的内建函数**

关于更多内建函数，将会在后面学习中具体说明，今天先学习关于数字的内建函数。它们有：

**(1)、标准内建函数：**

cmp()、str()、type()对于任何类型都可以使用

[复制代码](javascript:void(0);)

>>> cmp(1,2) #比较大小

-1

>>> cmp(2,1)

1

>>> cmp(2,2)

0

>>> str(0xFF) #转换成字符串

'255'

>>> type(0xFF) #返回对象类型

<type 'int'>

>>> type('str')

<type 'str'>

[复制代码](javascript:void(0);)

**（2）、转换工厂函数：**

int()、long()、float()、bool()、complex()，它们都是转换成跟数字有关的函数。如：

[复制代码](javascript:void(0);)

>>> '111'

'111'

>>> int('222')

222

>>> long('222')

222L

>>> float('222')

222.0

>>> bool(1)

True

>>> complex(2)

(2+0j)

>>> bool(True)

True

>>> bool(454)

True

>>> bool(0)

False

[复制代码](javascript:void(0);)

**(3)、功能函数**

有abs()返回给定参数的绝对值：

>>> abs(-3)

3

coerce()数据类型对两个参数进行转换函数：

[复制代码](javascript:void(0);)

>>> coerce(1,2)

(1, 2)

>>> coerce(1,-2)

(1, -2)

>>> coerce(1,123L)

(1L, 123L)

>>> coerce(1.3,123L)

(1.3, 123.0)

>>> coerce(1j,123L)

(1j, (123+0j))

[复制代码](javascript:void(0);)

divmod()通过取余，返回商和余数的元祖：

>>> divmod(10,2)

(5, 0)

>>> divmod(10,3)

(3, 1)

pow()和\*\*都可以进行指数运算：

>>> 5\*\*2

25

>>> pow(5,2)

25

round()四舍五入：

[复制代码](javascript:void(0);)

>>> round(3)

3.0

>>> round(3.45)

3.0

>>> round(3.999999,1)

4.0

[复制代码](javascript:void(0);)

更多内建函数见Python官网文档。

# [Python零碎知识(1):strip lstrip rstrip使用方法](http://www.cnblogs.com/BeginMan/archive/2013/03/14/2958985.html)

一、原理介绍：

[复制代码](javascript:void(0);)

Python中的strip用于去除字符串的首尾字符，同理，lstrip用于去除左边的字符，rstrip用于去除右边的字符。

这三个函数都可传入一个参数，指定要去除的首尾字符。

需要注意的是，传入的是一个字符数组，编译器去除两端所有相应的字符，直到没有匹配的字符，比如：

theString = 'saaaay yes no yaaaass'

print theString.strip('say')

theString依次被去除首尾在['s'，'a'，'y']数组内的字符，直到字符在不数组内。所以，输出的结果为：

yes no

比较简单吧，lstrip和rstrip原理是一样的。

注意：当没有传入参数时，是默认去除首尾空格的。

theString = 'saaaay yes no yaaaass'

print theString.strip('say')

print theString.strip('say ') #say后面有空格

print theString.lstrip('say')

print theString.rstrip('say')

运行结果：

yes no

es no

yes no yaaaass

saaaay yes no  
注：这段解释来自：（pylemon's notebook）：http://www.cnblogs.com/pylemon/archive/2011/05/18/2050179.html

[复制代码](javascript:void(0);)

 二、实际应用

这里举一个demo，用于在pythonIED中控制输出菜单，选择相应的选项，操作对应的操作。如下：  
假设程序中已经存在newUser和oldUser两个函数，这里我只贴出了主要部分

[复制代码](javascript:void(0);)

1 #菜单控制界面

2 def showmenu():

3 prompt='''

4 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

5 Welcome to Python System!

6 -------------------------

7 |(N)ew User Login |

8 |(L)ogin your system |

9 |(Q)uit |

10 -------------------------

11 Enter Choice:

12 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

13 '''

14 done=False

15 while not done:

16 chosen=False

17 while not chosen:

18 try:

19 choice=raw\_input(prompt).strip()[0].lower() #取输入的字符串第一个字符

20 except(EOFError,KeyboardInterrupt):

21 choice='q' #抛出异常

22 print '\n You picked:[%s]' %choice

23 if choice not in 'nlq': #判断输出字符串中首字符是否属于'nlq'

24 print 'Invalid option,tyr again'

25 else:

26 chosen=True #一切正常则跳出循环

27 #根据输入信息进行控制

28 if choice=='q':done=True #如果输入'q'则跳出整个循环，不再执行

29 if choice=='n':newUser() #如果输入'n'则调用用户注册函数

30 if choice=='l':oldUser() #如果输入'l'则调用用户登陆函数

31

32 #主程序

33 if \_\_name\_\_=="\_\_main\_\_":

34 showmenu()

[复制代码](javascript:void(0);)

注：请注意红色部分代码

[**Python零碎知识(2):强大的zip**](http://www.cnblogs.com/BeginMan/archive/2013/03/14/2959447.html)

一、代码引导

首先看这一段代码：

[复制代码](javascript:void(0);)

1 >>> name=('jack','beginman','sony','pcky')

2 >>> age=(2001,2003,2005,2000)

3 >>> for a,n in zip(name,age):

4 print a,n

5

6 输出：

7 jack 2001

8 beginman 2003

9 sony 2005

10 pcky 2000

[复制代码](javascript:void(0);)

再看这一段代码：

[复制代码](javascript:void(0);)

1 all={"jack":2001,"beginman":2003,"sony":2005,"pcky":2000}

2 for i in all.keys():

3 print i,all[i]

4

5 输出：

6 sony 2005

7 pcky 2000

8 jack 2001

9 beginman 2003

[复制代码](javascript:void(0);)

发现它们之间的区别么？

最显而易见的是：第一种简洁、灵活、而且能顺序输入。

二、zip()函数

它是Python的内建函数，(与序列有关的内建函数有：sorted()、reversed()、enumerate()、zip()),其中 sorted()和zip()返回一个序列(列表)对象，reversed()、enumerate()返回一个迭代器(类似序列)

[复制代码](javascript:void(0);)

1 >>> type(sorted(s))

2 <type 'list'>

3 >>> type(zip(s))

4 <type 'list'>

5 >>> type(reversed(s))

6 <type 'listreverseiterator'>

7 >>> type(enumerate(s))

8 <type 'enumerate'>

[复制代码](javascript:void(0);)

那么什么是zip()函数 呢？

我们help(zip)看看：

[复制代码](javascript:void(0);)

1 >>> help(zip)

2 Help on built-in function zip in module \_\_builtin\_\_:

3

4 zip(...)

5 zip(seq1 [, seq2 [...]]) -> [(seq1[0], seq2[0] ...), (...)]

6

7 Return a list of tuples, where each tuple contains the i-th element

8 from each of the argument sequences. The returned list is truncated

9 in length to the length of the shortest argument sequence.

[复制代码](javascript:void(0);)

提示：不懂的一定多help

定义：zip([seql, ...])接受一系列可迭代对象作为参数，将对象中对应的元素打包成一个个tuple（元组），然后返回由这些tuples组成的list（列表）。若传入参数的长度不等，则返回list的长度和参数中长度最短的对象相同。

[复制代码](javascript:void(0);)

1 >>> z1=[1,2,3]

2 >>> z2=[4,5,6]

3 >>> result=zip(z1,z2)

4 >>> result

5 [(1, 4), (2, 5), (3, 6)]

6 >>> z3=[4,5,6,7]

7 >>> result=zip(z1,z3)

8 >>> result

9 [(1, 4), (2, 5), (3, 6)]

10 >>>

[复制代码](javascript:void(0);)

zip()配合\*号操作符,可以将已经zip过的列表对象解压

1 >>> zip(\*result)

2 [(1, 2, 3), (4, 5, 6)]

更近一层的了解：  
内容来源：http://www.cnblogs.com/diyunpeng/archive/2011/09/15/2177028.html   (博客园人才真多！)

[复制代码](javascript:void(0);)

\* 二维矩阵变换（矩阵的行列互换）

比如我们有一个由列表描述的二维矩阵

a = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]

通过python列表推导的方法，我们也能轻易完成这个任务

print [ [row[col] for row in a] for col in range(len(a[0]))]

[[1, 4, 7], [2, 5, 8], [3, 6, 9]]

另外一种让人困惑的方法就是利用zip函数：

>>> a = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]

>>> zip(\*a)

[(1, 4, 7), (2, 5, 8), (3, 6, 9)]

>>> map(list,zip(\*a))

[[1, 4, 7], [2, 5, 8], [3, 6, 9]]

zip函数接受任意多个序列作为参数，将所有序列按相同的索引组合成一个元素是各个序列合并成的tuple的新序列，新的序列的长度以参数中最短的序列为准。另外(\*)操作符与zip函数配合可以实现与zip相反的功能，即将合并的序列拆成多个tuple。

①tuple的新序列

>>>>x=[1,2,3],y=['a','b','c']

>>>zip(x,y)

[(1,'a'),(2,'b'),(3,'c')]

②新的序列的长度以参数中最短的序列为准.

>>>>x=[1,2],y=['a','b','c']

>>>zip(x,y)

[(1,'a'),(2,'b')]

③(\*)操作符与zip函数配合可以实现与zip相反的功能,即将合并的序列拆成多个tuple。

>>>>x=[1,2,3],y=['a','b','c']

>>>>zip(\*zip(x,y))

[(1,2,3),('a','b','c')]

[复制代码](javascript:void(0);)

**其他高级应用：**

[复制代码](javascript:void(0);)

1.zip打包解包列表和倍数

>>> a = [1, 2, 3]

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

>>> z = zip(a, b)

>>> z

[(1, 'a'), (2, 'b'), (3, 'c')]

>>> zip(\*z)

[(1, 2, 3), ('a', 'b', 'c')]

2. 使用zip合并相邻的列表项

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

>>> zip(\*([iter(a)] \* 2))

[(1, 2), (3, 4), (5, 6)]

>>> group\_adjacent = lambda a, k: zip(\*([iter(a)] \* k))

>>> group\_adjacent(a, 3)

[(1, 2, 3), (4, 5, 6)]

>>> group\_adjacent(a, 2)

[(1, 2), (3, 4), (5, 6)]

>>> group\_adjacent(a, 1)

[(1,), (2,), (3,), (4,), (5,), (6,)]

>>> zip(a[::2], a[1::2])

[(1, 2), (3, 4), (5, 6)]

>>> zip(a[::3], a[1::3], a[2::3])

[(1, 2, 3), (4, 5, 6)]

>>> group\_adjacent = lambda a, k: zip(\*(a[i::k] for i in range(k)))

>>> group\_adjacent(a, 3)

[(1, 2, 3), (4, 5, 6)]

>>> group\_adjacent(a, 2)

[(1, 2), (3, 4), (5, 6)]

>>> group\_adjacent(a, 1)

[(1,), (2,), (3,), (4,), (5,), (6,)]

3.使用zip和iterators生成滑动窗口 (n -grams)

>>> from itertools import islice

>>> def n\_grams(a, n):

... z = (islice(a, i, None) for i in range(n))

... return zip(\*z)

...

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

>>> n\_grams(a, 3)

[(1, 2, 3), (2, 3, 4), (3, 4, 5), (4, 5, 6)]

>>> n\_grams(a, 2)

[(1, 2), (2, 3), (3, 4), (4, 5), (5, 6)]

>>> n\_grams(a, 4)

[(1, 2, 3, 4), (2, 3, 4, 5), (3, 4, 5, 6)]

4.使用zip反转字典

>>> m = {'a': 1, 'b': 2, 'c': 3, 'd': 4}

>>> m.items()

[('a', 1), ('c', 3), ('b', 2), ('d', 4)]

>>> zip(m.values(), m.keys())

[(1, 'a'), (3, 'c'), (2, 'b'), (4, 'd')]

>>> mi = dict(zip(m.values(), m.keys()))

>>> mi

{1: 'a', 2: 'b', 3: 'c', 4: 'd'}

[复制代码](javascript:void(0);)