# 一 入门课程介绍

## 1.1 Python的特点

优雅、明确、简单

## 1.2 python适合的领域

web网站和各种网络服务

系统工具和脚本

作为“胶水”语言把其他语言开发的模块包装起来方便使用

## 1.3 python不适合的领域

贴近硬件的代码（首选C）

移动开发：IOS/Android有各自的开发语言（Obj C，Swift/Java）

游戏开发：C/C++ (因为需要高速的渲染)

## 1.4 Python和其他语言对比

类型 运行速度 代码量

C 编译为机器码 非常快 非常多

Java 编译为字节码 快 多

Python 解释执行 慢 少

## 1.5 Python源码不能加密

# 二 安装Python

## 2.1 选择Python版本

Python是跨平台的语言。

目前有两个版本，2.7和3.3，两个版本不兼容。

因为很多第三方库是基于2.7版，所以教程是基于2.7版。

## 2.2 windows上安装Python

2.2.1

Python官网：[www.python.org](http://www.python.org) 下载python 2.7版本安装。路径选择默认路径。

2.2.2

在环境变量Path中，添加 Python 安装路径，如：C:\Python27。

2.2.3

在命令提示符中输入python，进入python交互式命令环境。

如，输入100 + 200，可以得到结果300.

输入exit()，可以退出。

## 2.3 第一个Python程序

2.3.1 打印Hello World

print ‘Hello World’

2.3.2 在notepad++ 上编写python代码文件。

设置--首选项--新建--编码，选择UTF-8（无BOM），勾选应用于ANSI文件。

输入print ‘Hello World’ ，print前不能有空格。

保存为以.py结尾命名的文件。

2.3.3 运行python文件

在命令提示符中，切换到文件目录，输入：python 文件名（如 python hello.py），python解释器就会直接运行文件。

# 三 Python变量和数据类型

## 3.1 Python中数据类型

Python能够直接处理的数据类型：

1. 整数
2. 浮点数 也就是小数。可以用数学写法，如1.23、3.14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，1.23\*10^9就是1.23e9，或者12.3e8，0.000012可以写成1.2e-5，等等。
3. 字符串
4. 布尔值 True、False，注意大小写。布尔值可以用and、or、not运算。
5. 空值 用None表示。
6. 此外，python还提供了列表、字典等多种数据类型，还允许创建自定义数据类型。

## 3.2 Python之print语句

print语句可以跟上多个字符串，用“,”隔开。print会依次打印每个字符串，遇到逗号“,”会输出一个空格。

例：print ‘100 + 200 =’, 100 + 200 运行结果为：100 + 200 = 300

## 3.3 Python的注释

Python的注释以 # 开头，后面的文字直到行尾都算注释。

## 3.4 Python中什么是变量

变量是用一个变量名表示，变量名必须是大小写英文、数字和下划线的组合，且不能用数字开头。比如：

a = 1, t\_007 = ‘T007’

***Python 中同一个变量可以反复赋值，而且可以是不同类型的变量。***

***这种变量本身类型不固定的语言称之为动态语言***，与之对应的是静态语言。静态语言在定义变量时必须指定变量类型，如果赋值的时候类型不匹配，就会报错。例如java是静态语言。

## 3.5 Python中定义字符串

字符串用’’或””括起来，如果字符串包含’或”，需要对特殊字符进行“转义”。Python字符串用 \ 进行转义。常用的转义字符还有：

\n：换行；

\t：一个制表符；

\\：\字符本身。

## 3.6 Python中raw字符串与多行字符串

如果一个字符串包含很多需要转义的字符，对每一个字符都进行转义会很麻烦。为了避免这种情况，我们可以在字符串前面加个前缀r，表示这是一个raw字符串，里面的字符就不需要转义了。例如：r’\(~\_~)/ \(~\_~)/’

但是r’...’表示法不能表示多行字符串，也不能表示包含’和”的字符串。

如果要表示多行字符串，可以用’’’...’’’表示。

还可以在多行字符串前面添加r，把这个多行字符串也变成一个raw字符串。

## 3.7 Python中的Unicode字符串

字符串还有一个编码问题。

因为计算机只能处理数字，如果要处理文本，就必须先把文本转换为数字才能处理。最早的计算机在设计时采用8个比特（bit）作为一个字节（byte），所以，一个字节能表示的最大的整数就是255（二进制11111111=十进制255），0 - 255被用来表示大小写英文字母、数字和一些符号，这个编码表被称为ASCII编码，比如大写字母 A 的编码是65，小写字母 z 的编码是122。

如果要表示中文，显然一个字节是不够的，至少需要两个字节，而且还不能和ASCII编码冲突，所以，中国制定了GB2312编码，用来把中文编进去。

类似的，日文和韩文等其他语言也有这个问题。为了统一所有文字的编码，Unicode应运而生。Unicode把所有语言都统一到一套编码里，这样就不会再有乱码问题了。

Unicode通常用两个字节表示一个字符，原有的英文编码从单字节变成双字节，只需要把高字节全部填为0就可以。

因为Python的诞生比Unicode标准发布的时间还要早，所以最早的Python只支持ASCII编码，普通的字符串'ABC'在Python内部都是ASCII编码的。

Python在后来添加了对Unicode的支持，以Unicode表示的字符串用u'...'表示，比如：print u’中文’，注意：不加u，中文就不能正常显示。

如果中文字符串在Python环境下遇到UnicodeDecodeError，这是因为.py文件保存的格式有问题。可以在第一行添加注释

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

目的是告诉python解释器用UTF-8编码读取源代码。然后用notepad++另存为...并选择UTF-8格式保存。

## 3.8 Python中整数和浮点数

注：此为python2.7版本，3.3版本不同

整数的运算结果仍为整数，浮点数的运算结果仍为浮点数，整数和浮点数混合运算结果为浮点数。

## 3.9 Python中的布尔类型

有True和False两种值。

与运算：and

或运算：or

非运算：not

Python把0、空字符串’’和None看成False，其他数值和非空字符串都看成True。

**短路计算**：

1. 在计算a and b时，如果a是False，则根据与运算法则，整个结果必定为False，因此返回a；如果a是True，则整个计算结果必定取决于b，因此返回b。
2. 在计算a or b时，如果a是True，则根据或运算法则，整个计算结果必定为True，因此返回a；如果a是False，则整个计算结果必定取决于b，因此返回b。

所以python解释器在做布尔运算时，只要能提前确定计算结果，它就不会往后算了，直接返回结果。

例：

（1）a = 'python'

print 'hello,', a or 'world'

b = ''

print 'hello,', b or 'world'

运行结果：

hello,python

hello,world

1. a = True

print a and ‘a=T’ or ‘a=F’

运行结果：

‘a=T’

# 四 List和Tuple类型

## **4.1 Python创建list**

Python 内置的一种数据类型是列表：list。list是一种有序的集合，可以随时添加和删除其中的元素。

list是数学意义上的有序集合，也就是说，list中的元素是按照顺序排列的。

list对象：[‘Michael’, ’Bob’, ‘Tracy’]

由于Python是动态语言，所以list中包含的元素并不要求都必须是同一种数据类型，我们完全可以在list中包含各种数据：

L = [‘Michael’ , 100 , True]

一个元素也没有的list，就是空list： empty\_list = []

## 4.2 Python按照索引访问list

索引从0开始。

L = [‘Adam’, ‘Lisa’, ‘Bart’]

L[0] = ‘Adam’

## 4.3 Python之倒序访问list

用-1这个索引来表示最后一个元素。类似的，有-2，-3……

## 4.4 Python之添加新元素

1. append()方法：把新的元素添加到list的尾部。

L = [‘Adam’, ‘Lisa’, ‘Bart’]

L.append(‘Paul’)

L = [‘Adam’, ‘Lisa’, ‘Bart’, ‘Paul’]

1. insert()方法：两个参数，第一个参数是索引号，第二个参数是待添加的新元素。

L = [‘Adam’, ‘Lisa’, ‘Bart’]

L.insert(0, ’Paul’)

L = [‘Paul’, ‘Adam’, ‘Lisa’, ‘Bart’]

## 4.5 Python从list删除元素

1. pop()方法：删掉list的最后一个元素，并且它还返回这个元素。
2. pop(索引)：按照索引删除元素。

## 4.6 Python中替换元素

对list中的某一个索引赋值

L = [‘Adam’, ‘Lisa’, ‘Bart’]

L[2] = ‘Paul’

L = [‘Adam’, ‘Lisa’, ‘Paul’]

将Adam和Bart两个元素调换位置：

L[0], L[2] = L[2], L[0] 或 L[0], L[-1] = L[-1], L[0]

L = [‘Bart’, ‘Lisa’, ‘Adam’]

## 4.7 Python之创建tuple

Tuple是另一种有序的列表，中文翻译为“元组”。tuple和list非常类似，但是，tuple一旦创建完毕，就不能修改了。

t = ( ‘Adam’, ‘Lisa’, ‘Bart’ )

创建tuple用()。

Tuple没有append()、insert()、pop()等方法，无法修改。

访问t元素：t[0]、t[-1]、t[2]

不能按索引赋值：t[0]=’Paul’，这样写会报错。

补充：t = (0,1,2,3,4,5,6,7,8,9) 与 t = tuple(range(0,10)) 运行结果相同

## 4.8 Python之创建单元素tuple

tuple和list一样，可以包含0个、1个和任意多个元素。

包含0个元素的tuple，也就是空tuple，直接用()表示。

包含1个元素的tuple，不能直接用t=(1) 创建，因为这时()会被python解释器当作运算时的优先级，导致t的值成为整数1，而不是tuple元素。

Python规定，单元素的tuple要多加一个逗号“,”，避免歧义。

t = (1,)

## 4.9 Python之“可变”的tuple

t = (‘a’, ‘b’, [‘A’, ‘B’])

L = t[2]

L[0] = ‘X’

L[1] = ‘Y’

print t

(‘a’, ‘b’, [‘X’, ‘Y’])

此时，表面上看，tuple的元素确实变了，但其实变的不是tuple的元素，而是list的元素。

Tuple一开始指向的list并没有变成别的list。

Tuple 的不变，意思是，tuple中每个元素的指向永远不变，但指向的这个list本身是可变的。

要创建一个内容也不变的tuple，就必须保证tuple的每一个元素本身也不能变。

# 五 条件判断和循环

## 5.1 Python之if语句

Python代码的缩进规则：具有相同缩进的代码被视为代码块。

缩进要严格按照python的习惯写法：4个空格，不要使用tab，更不要混合tab和空格，否则很容易造成因为缩进引起的语法错误。

注意：if语句后接表达式，然后用 : 表示代码块开始。

如果在python交互环境下敲代码，要特别留意缩进，并且退出缩进需要多敲一行回车。

age = 20

if age >= 18:

print ‘your age is’, age

print ‘adult’

print ‘END’

## 5.2 Python之if-else

if age >= 18:

print ‘adult’

else:

print ‘teenager’

## 5.3 Python之if-elif-else

if age >= 18:

print ‘adult’

elif age >= 6:

print ‘teenager’

elif age >= 3:

print ‘kid’

else:

print ‘baby’

## 5.4 Python之for循环

L = [‘Adam’, ‘Lisa’, ‘Bart’]

for name in L:

print name

## 5.5 Python之while循环

While循环不会迭代list或tuple的元素，而是根据表达式判断循环是否结束。

N = 10

x = 0

while x < N:

print x

x = x + 1

## 5.6 Python之break退出循环

sum = 0

x = 1

while True:

sum = sum + x

x = x + 1

if x > 100:

break

print sum

注意：python不支持x++表达式。

## 5.7 Python之continue继续循环

在循环过程中，可以用continue跳过后续循环代码，继续下一次循环。

计算及格分数的平均分：

L = [75,98,45,78,59,90]

sum = 0.0

n = 0

for x in L:

if x < 60:

continue

sum = sum + x

n = n + 1

print sum/n

## 5.8 Python之多重循环

for x in [‘A’, ‘B’, ‘C’]:

for y in [‘1’,’2’,’3’]:

print x + y

x每循环一次，y就会循环3次。

# 六 Dict和Set类型

## 6.1 Python之什么是dict

dict用来存储key-value，用{}表示

d = {‘Adam’: 95, ‘Lisa’: 85, ‘Bart’: 59}

由于dict也是集合，可以用len()计算集合的大小。

len(d) -- 3

通过定义一个dict，然后print发现，dict是无序的。

## 6.2 Python之访问dict

可以使用d[key]的形式查找对应的value。

如果key不存在，会报错keyError。要避免KeyError发生，有两个办法：

1. 先判断key是否存在，用in操作符：

if ‘Paul’ in d:

print d[‘Paul’]

1. 使用dict本身提供的一个get方法，在key不存在的时候，返回None:

>>>print d.get(‘Bart’)

59

>>>print d.get(‘Paul’)

None

## 6.3 Python中dict的特点

dict的第一个特点是查找速度快，无论dict有10个元素还是10万个元素，查找速度都一样，而list的查找速度随着元素增加而逐渐下降。

dcit的缺点是占用内存大，还会浪费很多内容，list正好相反，占用内存小，但是查找速度慢。

由于dict是按key查找，所以在一个dict中，key不能重复。

dict的第二个特点是存储的key-value序对是没有顺序的。

dcit的第三个特点是作为key的元素必须不可变，Python的基本类型如字符串、整数、浮点数都是不可变的，都可以作为key。但是list是可变的，就不能作为key。

## 6.4 Python更新dict

添加新的key-value：d[‘Paul’] = 72

如果key已经存在，用新的value替换掉原来的。

## 6.5 Python之遍历dict

直接使用for循环可以遍历dict的key:

d = {‘Adam’: 95, ‘Lisa’: 85, ‘Bart’: 59}

for key in d:

print key

print d[key]

另一种写法：

for key, v in d.items()

print key, ‘:’, v

## 6.6 Python中什么是set

set持有一系列元素，和list很像，但是set的元素没有重复，而且是无序的。

创建set的方式是调用set()并传入一个list，list的元素将作为set的元素：

>>>s = set( [‘A’, ‘B’, ‘C’] )

如果传入的list包含重复元素，set会自动去掉重复的元素。

## 6.7 Python之访问set

用in操作符判断某个元素是否在set中：’A’ in s

## 6.8 Python之set的特点

Set的内部结构和dict很像，唯一区别是不存储value，因此，判断一个元素是否在set中速度很快。

Set存储的元素和dict的key类似，必须是不变对象，因此，任何可变对象是不能放入set中的。

最后，set存储的元素是没有顺序的。

## 6.9 Python之遍历set

for name in s:

print name

注意：for循环遍历set时，元素的顺序和list的顺序很可能时不同的，而且不同的机器上运行的结果也可能不同。

s = set( [('Adam', 95), ('Lisa', 85), ('Bart', 59)] )

for x in s:

print x[0], ':' , x[1]

与

for x, score in s:

print x, ‘:’ , score

运行结果相同。

## 6.10 Python之更新set

由于set存储的是一组不重复的无序元素，因此，更新set主要做两件事：

一是把新的元素添加到set中，二是把已有元素从set中删除。

1、添加元素，用add()方法：

s = set( [1,2,3] )

s.add(4)

如果添加的元素已经存在于set中, add()不会报错，但是不会加进去了。

2、删除元素，用remove()方法：

s.remove(3)

如果删除的元素不存在set中，remove()会报错。

# 七 函数

## 7.1 Python之什么是函数

函数就是最基本的一种代码抽象的方式。

Python不但能非常灵活地定义函数，而且本身内置了很多有用的函数，可以直接调用。

## 7.2 Python之调用函数

要调用一个函数，需要知道函数的名称和参数。

可以直接从Python的官网查看文档：

http://docs.python.org/2/library/functions.html#abs

abs(x)：求绝对值。例：abs(-100) --> 100

调用函数的时候，如果传入的参数数量不对，会报TypeError的错误，并且Python会明确的告诉你：abs()有且仅有1个参数。

如果传入的参数数量是对的，但参数类型不能被函数所接受，也会报TypeError的错误，并且给出错误信息：str是错误的参数类型。

cmp(x,y)：比较函数。x<y，返回-1；x==y，返回0；x>y，返回1.

Python内置的常用函数还包括数据类型转换函数：

int(‘123’) --> 123

int(12.34) --> 12

str(123) --> ‘123’

sum(list)：参数为一个list，返回list所有元素之和。

问题：请计算 1\*1 + 2\*2 + 3\*3 + ... + 100\*100。

第一种：

L = [1]

n = 2

while n <= 100:

L.append(n \* n)

n += 1

print sum(L)

第二种：

L = [x \* x for x in range(1,101)]

第三种：

L = [x \*\* 2 for x in range(1,101)]

## 7.3 Python之编写函数

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、冒号（类似javascript中定义function），然后，在缩进块中编写函数体，函数的返回值用return语句返回。

def my\_abs(x):

if x >= 0:

return x

else:

return -x

如果没有return语句，函数执行完毕后返回None。

return None可以简写为return。

## 7.4 Python函数之返回多值

import math

def move(x, y, step, angle):

nx = x + step \* math.cos(angle)

ny = y - step \* math.sin(angle)

return nx, ny

如上函数，给定一个坐标、位移和角度，就可以计算出新的坐标。#math包提供了sin()和cos()函数，用import引用它。这样就可以同时获得返回值：

>>>x,y = move(100, 100, 60, math.pi/6)

>>>print x, y

151.961524277 70.0

但其实这只是一种假象，Python函数返回的仍然是单一值：

>>>r = move(100, 100, 60, math.pi/6)

>>>print r

(151.961524277, 70.0)

事实上，**返回值是一个tuple。**

但是，在语法上，返回一个tuple可以省略括号，而多个变量可以同时接收一个tuple，按位置赋给对应的值，所以，Python的函数返回多值其实就是返回一个tuple，但写起来更方便。

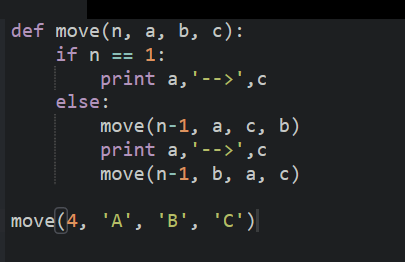
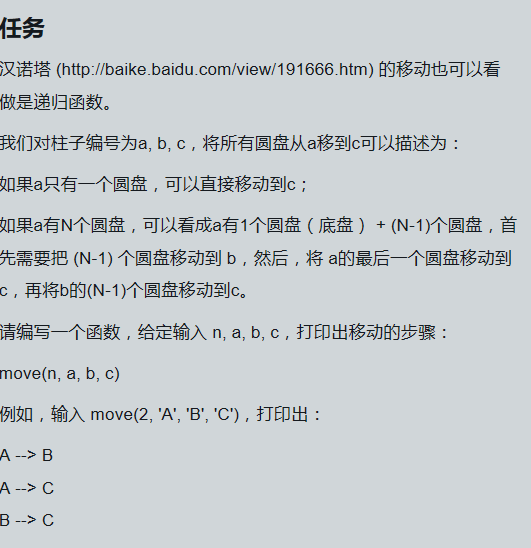
求平方根：math.sqrt()

## 7.5 Python之递归函数

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，这个函数就是递归函数。

递归函数的优点是定义简单，逻辑清晰。理论上，所有的递归函数都可以写成循环的方式，但循环的逻辑不如递归清晰。

**使用递归函数需要注意防止栈溢出。**在计算机中，函数调用是通过栈（stack）这种数据结构实现的，每当进入一个函数调用，栈就会加一层栈帧，每当函数返回，栈就会减一层栈帧。由于栈的大小不是无限的，所以，递归调用的次数太多，会导致栈溢出。



## 7.6 Python之定义默认参数

定义函数的时候，还可以有默认参数。

例如Python自带的int()函数，其实就有两个参数，我们既可以传一个参数，又可以传两个参数：

>>>int(‘123’)

123

>>>int(‘123’, 8)

83

int()函数的第二个参数是转换进制，如果不传，默认是十进制（base=10），如果传了，就用传入的参数。

可见，函数的默认参数的作用是简化调用。

定义一个函数，计算x的n次方，如果不传n，默认计算x的平方：

def power(x, n=2):

s = 1

while n > 0:

n = n - 1

s = s \* x

return s

由于函数的参数按从左到右的顺序匹配，所以默认参数只能定义在必需参数的后面。

## 7.7 Python之定义可变参数

可变参数是指一个函数能接受任意个参数。可变参数的名字前面有个\*号。

def fn(\*args):

print args

>>> fn()

()

>>> fn(‘a’)

(‘a’, )

>>> fn(‘a’, ‘b’)

(‘a’, ‘b’)

Python解释器会把传入的一组参数组装成一个tuple传递给可变参数，因此，在函数内部，直接把变量args看成一个tuple就好了。

定义可变参数的目的也是为了简化调用。

# 八 切片

## 8.1 对list进行切片

Python提供了切片（slice）操作符，用于取一个list中指定索引范围。

L = [‘a’, ‘b’, ‘c’, ‘d’]

>>> L[0:3]

[‘a’, ‘b’, ‘c’]

L[0:3]表示，从索引0开始取，直到索引3，但不包括索引3。

1、如果第一个索引是0，可以省略：

>>>L[:3]

[‘a’, ‘b’, ‘c’]

2、也可以从索引1开始：

>>>L[1 : 3]

[‘b’, ‘c’]

3、只用一个 : ，表示从头到尾：

>>>L[ : ]

[‘a’, ‘b’, ‘c’, ‘d’]

因此，L[:]实际上复制了一个新的list。

1. 切片操作还可以指定第三个参数：

>>> L[ ::2 ]

[‘a’, ‘c’]

第三个参数表示每N个取一个，上面的L[::2]会每两个元素取出一个来，也就是隔一个取一个。

>>> L = range(1, 101)

>>> print L[2::3] #3的倍数

>>> print L[4:50:5] #不大于50的5的倍数

把list换成tuple，切片操作完全相同，只是切片的结果也变成了tuple。

## 8.2 倒序切片

>>> L = [‘a’, ‘b’, ‘c’, ‘d’]

>>> L[-2: ]

[‘c’, ‘d’]

>>> L[: -2]

[‘a’, ‘b’]

>>> L[ -3 : -1]

[‘b’, ‘c’]

>>> L[ -4 : -1 : 2]

[‘a’, ‘c’]

倒序切片包含起始索引，不包含结束索引。

取出1-100的数列中，最后10个5的倍数：

>>> L = range(1, 101)

>>> print L[-46::5]

或者 >>> print L[4::5][-10: ]

## 8.3 对字符串切片

字符串’xxx’和Unicode字符串u’xxx’也可以看成是一种list，每个元素就是一个字符。因此，字符串也可以用切片操作，只是操作结果仍是字符串：

>>> ‘ABCDEFG’[ :3 ]

‘ABC’

>>> ‘ABCDEFG’[ -3: ]

‘EFG’

>>> ‘ABCDEFG’[ ::2 ]

‘ACEG’

Python没有针对字符串的截取函数，只需要切片一个操作就可以完成，非常简单。

字符串有个方法upper()可以把字符串变成大写：’abc’.upper()

# 九 迭代

## 9.1 什么是迭代

在Python中，给定一个list或tuple，可以通过for循环来遍历，称为迭代。用for...in来完成。

Python的for循环不仅可以用在list或tuple上，还可以作用在其他任何可迭代对象上。

集合指包含一组元素的数据结构，我们已经介绍的包括：

1. 有序集合：list，tuple，str和unicode；
2. 无序集合：set
3. 无序集合并且有key-value对：dict

而迭代是一个动词，指的是一种操作，在Python中，就是for循环。

迭代与按下标访问数组最大的不同是，后者是一种具体的迭代实现方式，而前者只关心迭代结果，根本不关心迭代内部是如何实现的。

## 9.2 索引迭代

Python中，迭代永远是取出元素本身，而非元素的索引。

想要在for循环中拿到索引，方法是使用enumerate()函数：

>>> L = [‘a’, ‘b’, ‘c’, ‘d’]

>>> for index, name in enumerate(L):

>>> print index, ‘-’, name

0 - a

使用enumerate()函数，可以在for循环中同时绑定索引index和元素name。但是，这不是enumerate()的特殊语法。实际上，enumerate()函数把list变成了类似

[(0, ‘a’), (1, ‘b’), (2, ’c’), (3, ‘d’)]

因此，迭代的每一个元素实际上是一个tuple。

>>> for t in enumerate(L):

>>> index = t[0]

>>> name = t[1]

如果我们知道每个tuple元素都包含两个元素，for循环可以简写为：

>>> for index, name in enumerate(L):

可见，索引迭代不是真的按索引访问，而是由enumerate()函数自动把每个元素变成(index, element)这样的tuple，再迭代，就同时获得了索引和元素本身。

zip()函数可以把两个list变成一个list：

>>> zip( [10, 20, 30], [‘A’, ‘B’, ‘C’] )

[(10, ‘A’), (20, ‘B’), (30, ‘C’)]

## 9.3 迭代dict的value

dict对象有一个values()方法，这个方法把dict转换成一个包含所有value的list，这样，我们迭代的就是dict的每一个value：

>>> d = { ‘Adam’ : 95, ‘Lisa’ : 85, ‘Bart’ : 59}

>>> print d.values()

[ 85, 95, 59]

>>> for v in d.values():

>>> print v

85

95

59

dict还有一个itervalues()方法，用itervalues()方法替代values()方法，迭代效果完全一样。

>>> print d.itervalues()

<dictionary-valueiterator object at 0x106adbb50>

>>> for v in d.itervalues():

>>> print v

85

95

59

两个方法的不同之处：

1. values()方法实际上把一个dict转换成了包含value的list。

2. 但是itervalues()方法不会转换，它会在迭代过程中依次从dict中取出value，所以itervalues()方法比values()方法节省了生成list所需的内存。

3. 打印itervalues()发现它返回一个<dictionary-valueiterator>对象，这说明在Python中，for循环可作用的迭代对象远不止list，tuple，str，unicode，dict等，任何可迭代对象都可以作用于for循环，而内部如何迭代我们通常并不用关心。

如果一个对象说自己可迭代，那我们就直接用for循环去迭代它，可见，迭代是一种抽象的数据操作，它不对迭代对象内部的数据有任何要求。

## 9.4 迭代dict的key和value

>>> d = { ‘Adam’ : 95, ‘Lisa’ : 85, ‘Bart’ : 59}

>>> print d.items()

[ (‘Lisa’, 85), (‘Adam’, 95), (‘Bart’, 59) ]

items()方法把dict对象转换成了包含tuple的list，我们对这个list进行迭代，可以同时获得key和value：

>>> for key, value in d.items():

>>> print key, ‘:’, value

Lisa : 85

Adam : 95

Bart : 59

和values()有一个itervalues()类似，items()也有一个对应的iteritems()，iteritems()不把dict转换成list，而是在迭代过程中不断给出tuple，所以，iteritems()不占用额外的内存。

# 十 列表生成式

## 10.1 生成列表

1、>>> range(1, 11)

[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

1. >>> [ x \* x for x in range(1, 11)]

[1, 4, 9, 16, 25, 36, 49, 64, 81, 100]

2）这种写法就是Python特有的列表生成式。利用列表生成式，可以以非常简洁的代码生成list。

写列表生成式时，要把生成的元素 x \* x 放在前面，后面跟for循环，就可以把list创建出来。

生成100以内的list[1,3,5,7,9,...]：

1. range(1, 100, 2)
2. range(1, 100)[ ::2 ]

## 10.2 复杂表达式

假设有如下的dict:

>>> d = { ‘Adam’: 95, ‘Lisa’: 85, ‘Bart’: 59 }

完全可以通过一个复杂的列表生成式把它变成一个HTML表格：

>>> tds = [ ‘<tr><td>%s</td><td>%s</td></tr>’ %(name, score) for name, score in d.iteritems() ]

>>> print ‘<table>’

>>> print ‘<tr><th>Name</th><th>Score></th></tr>’

>>> print ‘\n’.join(tds)

>>> print ‘</table>’

注：字符串可以通过 % 进行格式化，用指定的参数替代%s。字符串的join()方法可以把一个list拼接成一个字符串。

## 10.3 条件过滤

列表生成式的for循环后面还可以加上if判断。

>>> [ x \* x for x in range(1, 11) if x % 2 == 0 ]

[ 4, 16, 36, 64, 100 ]

有了if条件，只有if判断为True的时候，才把循环的当前元素添加到列表中。

注：isinstance(x, str) 可以判断变量x是否是字符串。

## 10.4 多层表达式

for循环可以嵌套，因此，在列表生成式中，也可以用多层for循环来生成列表。

>>> [ m + n for m in ‘ABC’ for n in ‘123’ ]

[ ‘A1’, ‘A2’, ‘A3’, ‘B1’, ‘B2’, ‘B3’, ‘C1’, ‘C2’, ‘C3’ ]