第 5 章 条件，循环和其他语句

1.print 和 import

1.使用逗号输出

```
>>>1,2,3  (1,2,3)  >>>print 1,2,3  1 2 3
```

2.把某件事作为另一件事导入

```
import module
from module import function
from module import function,anotherfunction
from module import *
import math as foobar
from math import sqrt as foobar
```

2.赋值魔法

1.序列解包

```
>>>x,y,z=1,2,3     >>>print x,y,z  1 2 3
>>>values=1,2,3    >>>values      (1,2,3)
>>>x,y,z=values    >>>x            1
```

2.链式赋值

```
>>> x=y=fun()
```

等于 x=func() x=y

不一定等价于： x=func() y=func()

3.增量赋值

```
>>> x += 1   foo *= 2
```

3.语句块

标准缩进量是 4 个空格

4.条件和条件语句

1.bool 变量的作用

解释为假的有：False，None，[] ,0,’’,{},() 相当与没有东西

但是它们本身并不相等。

2.复杂的条件

1.比较运算符

主要注意一下比较不兼容的类型。如： 3 > ‘edas’ 这种没什么意义，在 python3 中不允许不兼容的类型进行比较。注意 x <> y,不建议使用<>运算符.

2.is 同一性运算符

前面已经说过了。这里举个例：

```
x=y=[1,2,3]  z=[1,2,3]          x y 绑定在同一个列表上
x is y True   x is z False      z 是另一个列表
```

3.bool 运算符

```
and or not
```

和 C 一样，也是短路逻辑

x and y: 当 x 为假，立即返回，不判断 y 的真假

4.断言

```
assert
```

和 c 一样的用法

5.循环

1.while

2.for

range(0,4)　　[0,1,2,3]　　一次创建整个序列

xrange(0,4)　　xrange(0,4)　一次只创建一个数

python3 中 range()被转换成 xrange 风格

3.循环遍历字典

```
d={'1':'x','2':'y'}
for key in d:
    print key,d[key]
```

4.迭代工具

1.并行迭代

```
name=['x','y','z']
age=[1,2,3]
for i in range(len(name):
  print name[i],age[i]
```

zip 内建函数：

zip(name,age)　[('x',1),('y',2),('z',3)]

循环解包

```
for name,age in zip(name,age):
   print name,age
```

zip 可以用于不等长的序列，直到最短的序列用完为止。

2.编号迭代

enumerate 函数

```
for index,string in enumerate(strings):
    if 'xxx' in string:
        strings[index] = '[ssss]'
```

3.翻转和排序迭代

reversed 和 sorted

sorted[3,2,1,2]　[1,2,2,3] 返回一个列表

reversed　　返回一个迭代对象

5.跳出循环

1.break

2.continue　都和 C 一样

3.while True /break

6.列表推导式

列表推导式是利用其他列表创建新的列表

```
>>> [x*x for  x  in range(10)]

>>> boy=['a','b','c']
>>> girl=[bb','aa','cc']
>>> [ b+'+'+g for b in boy for g in girl if b[0]==g[0] ]
```

改进：

```
letterGirl={}
 for gs in girl:
    lettergirl.setdefault(gs)[0].append(gs)
 print [b+'+'+g for b in boy for g in letterGirl[b[0]]
```

7.pass del exec eval

pass: 程序什么事情不都做 因为空代码块是非法的

del: 对于 **del** 前面介绍过。如果 **x y** 都指向同一个列表，其实删除 **x** 并不一影响 **y** 的值。**del** 只是删除名称，并不删除列表本身，如果某个值不用的时候，**python** 负责内存的回收。

exec: 和 **C** 差不多的功能。执行字符串的命令。 **exec "print 'hello world'"** 不要乱用 **exec**，最好给它提供命名空间。如：

```
from math import sqrt
scope={}
exec 'sqrt=1' in scope
sqrt(4)  2
scope['sqrt']=1  把 scope 打印出来 scope.keys() 会发现里面有内建函数和值
 ['sqrt','__builtins__']
```

eval: 用于对字符串的表达式求值

如： **eval('2+3-5*0')**

eval(raw_input()) 等同于 **input()** 它们的区别前面已经讲了。

eval 也使用命名空间

```
scope={}
exec 'x=2' in scope
eval('x*x',scope)
4
```

表5-2 本章的新函数

函 数	描 述
chr(n)	当传入序号n时，返回n所代表的包含一个字符的字符串，(0≤n<256)
eval(source[, globals[, locals]])	将字符串作为表达式计算，并且返回值
enumerate(seq)	产生用于迭代的（索引，值）对
ord(c)	返回单字符字符串的int值
range([start,] stop[, step])	创建整数的列表
reversed(seq)	产生seq中值的反向版本，用于迭代
sorted(seq[, cmp][, key][, reverse])	返回seq中值排序后的列表
xrange([start,] stop[, step])	创造xrange对象用于迭代
zip(seq1, seq2,...)	创造用于并行迭代的新序列