Python Number(数字)

Python Number 数据类型用于存储数值。

数据类型是不允许改变的,这就意味着如果改变 Number 数据类型的值，将重新分配内存空间。

以下实例在变量赋值时 Number 对象将被创建：

```
var1 = 1
var2 = 10
```

您也可以使用del语句删除一些 Number 对象引用。

del语句的语法是：

```
del var1[,var2[,var3[....,varN]]]
```

您可以通过使用del语句删除单个或多个对象，例如：

```
del var
del var_a, var_b
```

Python 支持四种不同的数值类型：

- 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 integers) - 无限大小的整数，整数最后是一个大写或小写的L。
- 浮点型(floating point real values) - 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 numbers) - 复数由实数部分和虚数部分构成，可以用a + bj,或者complex(a,b)表示， 复数的实部a和虚部b都是浮点型。

int	long	float	complex
10	51924361L	0.0	3.14j
100	-0x19323L	15.20	45.j
-786	0122L	-21.9	9.322e-36j
080	0xDEFABCECBDAECBFBAEl	32.3+e18	.876j
-0490	535633629843L	-90.	-.6545+0J
-0x260	-052318172735L	-32.54e100	3e+26J
0x69	-4721885298529L	70.2-E12	4.53e-7j

- 长整型也可以使用小写"L"，但是还是建议您使用大写"L"，避免与数字"1"混淆。Python使用"L"来显示长整型。
- Python还支持复数，复数由实数部分和虚数部分构成，可以用a + bj,或者complex(a,b)表示， 复数的实部a和虚

部b都是浮点型

Python Number 类型转换

```
int(x [,base ])         将x转换为一个整数
long(x [,base ])        将x转换为一个长整数
float(x )               将x转换到一个浮点数
complex(real [,imag ])  创建一个复数
str(x )                 将对象 x 转换为字符串
repr(x )                将对象 x 转换为表达式字符串
eval(str )              用来计算在字符串中的有效Python表达式,并返回一个对象
tuple(s )               将序列 s 转换为一个元组
list(s )                将序列 s 转换为一个列表
chr(x )                 将一个整数转换为一个字符
unichr(x )              将一个整数转换为Unicode字符
ord(x )                 将一个字符转换为它的整数值
hex(x )                 将一个整数转换为一个十六进制字符串
oct(x )                 将一个整数转换为一个八进制字符串
```

Python math 模块、cmath 模块

Python 中数学运算常用的函数基本都在 math 模块、cmath 模块中。

Python math 模块提供了许多对浮点数的数学运算函数。

Python cmath 模块包含了一些用于复数运算的函数。

cmath 模块的函数跟 math 模块函数基本一致，区别是 cmath 模块运算的是复数，math 模块运算的是数学运算。

要使用 math 或 cmath 函数必须先导入：

```
import math
```

查看 math 查看包中的内容:

```
>>> import math
>>> dir(math)
['__doc__', '__file__', '__loader__', '__name__', '__package__', '__spec__', 'acos',
'acosh', 'asin', 'asinh', 'atan', 'atan2', 'atanh', 'ceil', 'copysign', 'cos', 'cosh',
'degrees', 'e', 'erf', 'erfc', 'exp', 'expm1', 'fabs', 'factorial', 'floor', 'fmod',
'frexp', 'fsum', 'gamma', 'gcd', 'hypot', 'inf', 'isclose', 'isfinite', 'isinf',
```

```
    'isnan', 'ldexp', 'lgamma', 'log', 'log10', 'log1p', 'log2', 'modf', 'nan', 'pi', 'pow',
     'radians', 'sin', 'sinh', 'sqrt', 'tan', 'tanh', 'tau', 'trunc']
>>>
```

下文会介绍各个函数的具体应用。

查看 cmath 查看包中的内容

```
>>> import cmath
>>> dir(cmath)
['__doc__', '__file__', '__loader__', '__name__', '__package__', '__spec__', 'acos',
'acosh', 'asin', 'asinh', 'atan', 'atanh', 'cos', 'cosh', 'e', 'exp', 'inf', 'infj',
'isclose', 'isfinite', 'isinf', 'isnan', 'log', 'log10', 'nan', 'nanj', 'phase', 'pi',
'polar', 'rect', 'sin', 'sinh', 'sqrt', 'tan', 'tanh', 'tau']
>>>
```

实例

```
>>> import cmath
>>> cmath.sqrt(-1)
1j
>>> cmath.sqrt(9)
(3+0j)
>>> cmath.sin(1)
(0.8414709848078965+0j)
>>> cmath.log10(100)
(2+0j)
>>>
```

Python数学函数

函数	返回值 (描述)
abs(x)	返回数字的绝对值，如abs(-10) 返回 10
ceil(x)	返回数字的上入整数，如math.ceil(4.1) 返回 5
cmp(x, y)	如果 x < y 返回 -1, 如果 x == y 返回 0, 如果 x > y 返回 1
exp(x)	返回e的x次幂(e x),如math.exp(1) 返回2.718281828459045
fabs(x)	返回数字的绝对值，如math.fabs(-10) 返回10.0
floor(x)	

	返回数字的下舍整数，如math.floor(4.9)返回 4
log(x)	如math.log(math.e)返回1.0,math.log(100,10)返回2.0
log10(x)	返回以10为基数的x的对数，如math.log10(100)返回 2.0
max(x1, x2,...)	返回给定参数的最大值，参数可以为序列。
min(x1, x2,...)	返回给定参数的最小值，参数可以为序列。
modf(x)	返回x的整数部分与小数部分，两部分的数值符号与x相同，整数部分以浮点型表示。
pow(x, y)	x**y 运算后的值。
round(x [,n])	返回浮点数x的四舍五入值，如给出n值，则代表舍入到小数点后的位数。
sqrt(x)	返回数字x的平方根

Python随机数函数

随机数可以用于数学，游戏，安全等领域中，还经常被嵌入到算法中，用以提高算法效率，并提高程序的安全性。

Python包含以下常用随机数函数：

函数	描述
choice(seq)	从序列的元素中随机挑选一个元素，比如random.choice(range(10))，从0到9中随机挑选一个整数。
randrange ([start,] stop [,step])	从指定范围内，按指定基数递增的集合中获取一个随机数，基数默认值为 1
random()	随机生成下一个实数，它在[0,1)范围内。
seed([x])	改变随机数生成器的种子seed。如果你不了解其原理，你不必特别去设定seed，Python会帮你选择seed。
shuffle(lst)	将序列的所有元素随机排序
uniform(x, y)	随机生成下一个实数，它在[x,y]范围内。

Python三角函数

Python包括以下三角函数：

函数	描述
acos(x)	返回x的反余弦弧度值。
asin(x)	返回x的反正弦弧度值。
atan(x)	返回x的反正切弧度值。
atan2(y, x)	返回给定的 X 及 Y 坐标值的反正切值。
cos(x)	返回x的弧度的余弦值。
hypot(x, y)	返回欧几里德范数 sqrt(x*x + y*y)。
sin(x)	返回的x弧度的正弦值。
tan(x)	返回x弧度的正切值。
degrees(x)	将弧度转换为角度,如degrees(math.pi/2) ， 返回90.0
radians(x)	将角度转换为弧度

Python数学常量

常量	描述
pi	数学常量 pi（圆周率，一般以π来表示）
e	数学常量 e，e即自然常数（自然常数）。

Python 字符串

字符串是 Python 中最常用的数据类型。我们可以使用引号（`'` 或 `"`）来创建字符串。

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 = "Python Runoob"
```

Python 访问字符串中的值

Python 不支持单字符类型，单字符在 Python 中也是作为一个字符串使用。

Python 访问子字符串，可以使用方括号来截取字符串，如下实例：

实例(Python 2.0+)

```
#!/usr/bin/python
var1 = 'Hello World!'
var2 = "Python Runoob"
print "var1[0]: ", var1[0]
print "var2[1:5]: ", var2[1:5]
```

以上实例执行结果：

```
var1[0]:  H
var2[1:5]:  ytho
```

Python 字符串连接

我们可以对字符串进行截取并与其他字符串进行连接，如下实例：

实例(Python 2.0+)

```
#!/usr/bin/python
# -*- coding: UTF-8 -*-
var1 = 'Hello World!'
print "输出 :- ", var1[:6] + 'Runoob!'
```

以上实例执行结果

```
输出 :-  Hello Runoob!
```

Python 转义字符

在需要在字符中使用特殊字符时，python 用反斜杠 `\` 转义字符。如下表：

转义字符	描述
\(在行尾时)	续行符
\\	反斜杠符号

\'	单引号
\"	双引号
\a	响铃
\b	退格(Backspace)
\e	转义
\000	空
\n	换行
\v	纵向制表符
\t	横向制表符
\r	回车
\f	换页
\oyy	八进制数，y 代表 0~7 的字符，例如：\012 代表换行。
\xyy	十六进制数，以 \x 开头，yy代表的字符，例如：\x0a代表换行
\other	其它的字符以普通格式输出

Python字符串运算符

下表实例变量 a 值为字符串 "Hello"，b 变量值为 "Python"：

操作符	描述	实例
+	字符串连接	>>>a + b 'HelloPython'
*	重复输出字符串	>>>a * 2 'HelloHello'
[]	通过索引获取字符串中字符	>>>a[1] 'e'
[:]	截取字符串中的一部分	>>>a[1:4] 'ell'
in	成员运算符 - 如果字符串中包含给定的字符返回 True	>>>"H" in a True
not in	成员运算符 - 如果字符串中不包	>>>"M" not in a

	含给定的字符返回 True	True
r/R	原始字符串 - 原始字符串：所有的字符串都是直接按照字面的意思来使用，没有转义特殊或不能打印的字符。 原始字符串除在字符串的第一个引号前加上字母"r"（可以大小写）以外，与普通字符串有着几乎完全相同的语法。	>>>print r'\n' \n >>> print R'\n' \n
%	格式字符串	请看下一章节

实例(Python 2.0+)

```
#!/usr/bin/python
# -*- coding: UTF-8 -*-
a = "Hello"
b = "Python"
print "a + b 输出结果：", a + b
print "a * 2 输出结果：", a * 2
print "a[1] 输出结果：", a[1]
print "a[1:4] 输出结果：", a[1:4]
if( "H" in a) :
print "H 在变量 a 中"
else :
print "H 不在变量 a 中"
if( "M" not in a) :
print "M 不在变量 a 中"
else :
print "M 在变量 a 中"
print r'\n'
print R'\n'
```

以上程序执行结果为：

```
a + b 输出结果： HelloPython
a * 2 输出结果： HelloHello
a[1] 输出结果： e
a[1:4] 输出结果： ell
H 在变量 a 中
M 不在变量 a 中
\n
```

```
\n
```

Python 字符串格式化

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。

在 Python 中，字符串格式化使用与 C 中 sprintf 函数一样的语法。

如下实例：

```
#!/usr/bin/python

print "My name is %s and weight is %d kg!" % ('Zara', 21)
```

以上实例输出结果：

```
My name is Zara and weight is 21 kg!
```

python 字符串格式化符号:

符 号	描述
%c	格式化字符及其ASCII码
%s	格式化字符串
%d	格式化整数
%u	格式化无符号整型
%o	格式化无符号八进制数
%x	格式化无符号十六进制数
%X	格式化无符号十六进制数（大写）
%f	格式化浮点数字，可指定小数点后的精度
%e	用科学计数法格式化浮点数
%E	作用同%e，用科学计数法格式化浮点数
%g	%f和%e的简写
%G	%F 和 %E 的简写
%p	用十六进制数格式化变量的地址

格式化操作符辅助指令:

符号	功能
*	定义宽度或者小数点精度
-	用做左对齐
+	在正数前面显示加号(+)
<sp>	在正数前面显示空格
#	在八进制数前面显示零('0')，在十六进制前面显示'0x'或者'0X'(取决于用的是'x'还是'X')
0	显示的数字前面填充'0'而不是默认的空格
%	'%%'输出一个单一的'%'
(var)	映射变量(字典参数)
m.n.	m 是显示的最小总宽度,n 是小数点后的位数(如果可用的话)

Python2.6 开始，新增了一种格式化字符串的函数 str.format()，它增强了字符串格式化的功能。

Python 三引号

Python 中三引号可以将复杂的字符串进行赋值。

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）。

```
 >>> hi = '''hi
there'''
>>> hi   # repr()
'hi\nthere'
>>> print hi  # str()
hi
there
```

三引号让程序员从引号和特殊字符串的泥潭里面解脱出来，自始至终保持一小块字符串的格式是所谓的WYSIWYG（所见即所得）格式的。

一个典型的用例是，当你需要一块HTML或者SQL时，这时当用三引号标记，使用传统的转义字符体系将十分费神。

```
errHTML = '''
```

```
<HTML><HEAD><TITLE>
Friends CGI Demo</TITLE></HEAD>
<BODY><H3>ERROR</H3>
<B>%s</B><P>
<FORM><INPUT TYPE=button VALUE=Back
ONCLICK="window.history.back()"></FORM>
</BODY></HTML>
'''
cursor.execute('''
CREATE TABLE users (
login VARCHAR(8),
uid INTEGER,
prid INTEGER)
''')
```

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

字符串方法是从python1.6到2.0慢慢加进来的——它们也被加到了Jython中。
这些方法实现了string模块的大部分方法，如下表所示列出了目前字符串内建支持的方法，所有的方法都包含了对Unicode的支持，有一些甚至是专门用于Unicode的。

方法	描述
string.capitalize()	把字符串的第一个字符大写
string.center(width)	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))	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')	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')	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))	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)	把字符串 string 中的 tab 符号转为空格，tab 符号默认的空格数是 8。
string.find(str, beg=0, end=len(string))	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.format()	格式化字符串
string.index(str, beg=0, end=len(string))	跟find()方法一样，只不过如果str不在 string中会报一个异常.
string.isalnum()	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()	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()	如果 string 只包含十进制数字则返回 True 否则返回 False.
string.isdigit()	如果 string 只包含数字则返回 True 否则返回 False.
string.islower()	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()	如果 string 中只包含数字字符，则返回 True，否则返回 False
string.isspace()	如果 string 中只包含空格，则返回 True，否则返回 False.
string.istitle()	如果 string 是标题化的(见 title())则返回 True，否则返回 False
string.isupper()	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)	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)	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()	转换 string 中所有大写字符为小写.
string.lstrip()	截掉 string 左边的空格
string.maketrans(intab, outtab)	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)	返回字符串 **str** 中最大的字母。
min(str)	返回字符串 **str** 中最小的字母。
string.partition(str)	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))	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))	类似于 find() 函数，返回字符串最后一次出现的位置，如果没有匹配项则返回 -1。
string.rindex(str, beg=0,end=len(string))	类似于 index()，不过是返回最后一个匹配到的子字符串的索引号。
string.rjust(width)	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()	删除 string 字符串末尾的空格.
string.split(str="", num=string.count(str))	以 str 为分隔符切片 string，如果 num 有指定值，则仅分隔 **num+1** 个子字符串
string.splitlines([keepends])	按照行('\r', '\r\n', '\n')分隔，返回一个包含各行作为元素的列表，如果参数 keepends 为 False，不包含换行符，如果为 True，则保留换行符。
string.startswith(obj, beg=0,end=len(string))	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])	在 string 上执行 lstrip()和 rstrip()
string.swapcase()	翻转 string 中的大小写

string.title()	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="")	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()	转换 string 中的小写字母为大写
string.zfill(width)	返回长度为 width 的字符串，原字符串 string 右对齐，前面填充0

Python 列表(List)

序列是Python中最基本的数据结构。序列中的每个元素都分配一个数字 - 它的位置，或索引，第一个索引是0，第二个索引是1，依此类推。

Python有6个序列的内置类型，但最常见的是列表和元组。

序列都可以进行的操作包括索引，切片，加，乘，检查成员。

此外，Python已经内置确定序列的长度以及确定最大和最小的元素的方法。

列表是最常用的Python数据类型，它可以作为一个方括号内的逗号分隔值出现。

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 = ['physics', 'chemistry', 1997, 2000]
list2 = [1, 2, 3, 4, 5 ]
list3 = ["a", "b", "c", "d"]
```

与字符串的索引一样，列表索引从0开始。列表可以进行截取、组合等。

访问列表中的值

使用下标索引来访问列表中的值，同样你也可以使用方括号的形式截取字符，如下所示：

实例(Python 2.0+)

```
#!/usr/bin/python
list1 = ['physics', 'chemistry', 1997, 2000]
list2 = [1, 2, 3, 4, 5, 6, 7 ]
print "list1[0]: ", list1[0]
print "list2[1:5]: ", list2[1:5]
```

以上实例输出结果：

```
list1[0]:  physics
list2[1:5]:  [2, 3, 4, 5]
```

更新列表

你可以对列表的数据项进行修改或更新，你也可以使用append()方法来添加列表项，如下所示：

实例(Python 2.0+)

```
#!/usr/bin/python
# -*- coding: UTF-8 -*-
list = [] ## 空列表
list.append('Google') ## 使用 append() 添加元素
list.append('Runoob')
print list
```

注意：我们会在接下来的章节讨论append()方法的使用

以上实例输出结果：

```
['Google', 'Runoob']
```

删除列表元素

可以使用 del 语句来删除列表的元素，如下实例：

实例(Python 2.0+)

```
#!/usr/bin/python
list1 = ['physics', 'chemistry', 1997, 2000]
print list1
del list1[2]
print "After deleting value at index 2 : "
print list1
```

以上实例输出结果：

```
['physics', 'chemistry', 1997, 2000]
After deleting value at index 2 :
['physics', 'chemistry', 2000]
```

注意：我们会在接下来的章节讨论remove()方法的使用

Python列表脚本操作符

列表对 + 和 * 的操作符与字符串相似。+ 号用于组合列表，* 号用于重复列表。

如下所示：

Python 表达式	结果	描述
len([1, 2, 3])	3	长度
[1, 2, 3] + [4, 5, 6]	[1, 2, 3, 4, 5, 6]	组合
['Hi!'] * 4	['Hi!', 'Hi!', 'Hi!', 'Hi!']	重复
3 in [1, 2, 3]	True	元素是否存在于列表中
for x in [1, 2, 3]: print x,	1 2 3	迭代

Python列表截取

Python 的列表截取实例如下：

```
>>>L = ['Google', 'Runoob', 'Taobao']
>>> L[2]
'Taobao'
>>> L[-2]
'Runoob'
>>> L[1:]
['Runoob', 'Taobao']
>>>
```

描述：

Python 表达式	结果	描述
L[2]	'Taobao'	读取列表中第三个元素
L[-2]	'Runoob'	读取列表中倒数第二个元素
L[1:]	['Runoob', 'Taobao']	从第二个元素开始截取列表

Python列表函数&方法

Python包含以下函数:

序号	函数
1	cmp(list1, list2) 比较两个列表的元素
2	len(list) 列表元素个数
3	max(list) 返回列表元素最大值
4	min(list) 返回列表元素最小值
5	list(seq) 将元组转换为列表

Python包含以下方法:

序号	方法
1	list.append(obj) 在列表末尾添加新的对象
2	list.count(obj) 统计某个元素在列表中出现的次数
3	list.extend(seq) 在列表末尾一次性追加另一个序列中的多个值（用新列表扩展原来的列表）
4	list.index(obj) 从列表中找出某个值第一个匹配项的索引位置
5	list.insert(index, obj) 将对象插入列表
6	list.pop([index=-1]) 移除列表中的一个元素（默认最后一个元素），并且返回该元素的值
7	list.remove(obj) 移除列表中某个值的第一个匹配项

8	list.reverse() 反向列表中元素
9	list.sort(cmp=None, key=None, reverse=False) 对原列表进行排序

Python 元组

Python 的元组与列表类似，不同之处在于元组的元素不能修改。

元组使用小括号，列表使用方括号。

元组创建很简单，只需要在括号中添加元素，并使用逗号隔开即可。

如下实例：

实例(Python 2.0+)

```
tup1 = ('physics', 'chemistry', 1997, 2000)
tup2 = (1, 2, 3, 4, 5 )
tup3 = "a", "b", "c", "d"
```

创建空元组

```
tup1 = ()
```

元组中只包含一个元素时，需要在元素后面添加逗号

```
tup1 = (50,)
```

元组与字符串类似，下标索引从0开始，可以进行截取，组合等。

访问元组

元组可以使用下标索引来访问元组中的值，如下实例:

实例(Python 2.0+)

```
#!/usr/bin/python
tup1 = ('physics', 'chemistry', 1997, 2000)
tup2 = (1, 2, 3, 4, 5, 6, 7 )
print "tup1[0]: ", tup1[0]
print "tup2[1:5]: ", tup2[1:5]
```

以上实例输出结果：

```
tup1[0]:  physics
```

```
tup2[1:5]:  (2, 3, 4, 5)
```

修改元组

元组中的元素值是不允许修改的，但我们可以对元组进行连接组合，如下实例:

实例(Python 2.0+)

```
#!/usr/bin/python
# -*- coding: UTF-8 -*-
tup1 = (12, 34.56)
tup2 = ('abc', 'xyz')
# 以下修改元组元素操作是非法的。
# tup1[0] = 100
# 创建一个新的元组
tup3 = tup1 + tup2
print tup3
```

以上实例输出结果：

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

删除元组

元组中的元素值是不允许删除的，但我们可以使用del语句来删除整个元组，如下实例:

实例(Python 2.0+)

```
#!/usr/bin/python
tup = ('physics', 'chemistry', 1997, 2000)
print tup
del tup
print "After deleting tup : "
print tup
```

以上实例元组被删除后，输出变量会有异常信息，输出如下所示：

```
('physics', 'chemistry', 1997, 2000)
After deleting tup :
Traceback (most recent call last):
  File "test.py", line 9, in <module>
```

```
    print tup
NameError: name 'tup' is not defined
```

元组运算符

与字符串一样，元组之间可以使用 + 号和 * 号进行运算。这就意味着他们可以组合和复制，运算后会生成一个新的元组。

Python 表达式	结果	描述
len((1, 2, 3))	3	计算元素个数
(1, 2, 3) + (4, 5, 6)	(1, 2, 3, 4, 5, 6)	连接
('Hi!',) * 4	('Hi!', 'Hi!', 'Hi!', 'Hi!')	复制
3 in (1, 2, 3)	True	元素是否存在
for x in (1, 2, 3): print x,	1 2 3	迭代

元组索引，截取

因为元组也是一个序列，所以我们可以访问元组中的指定位置的元素，也可以截取索引中的一段元素，如下所示：

元组：

```
L = ('spam', 'Spam', 'SPAM!')
```

Python 表达式	结果	描述
L[2]	'SPAM!'	读取第三个元素
L[-2]	'Spam'	反向读取，读取倒数第二个元素
L[1:]	('Spam', 'SPAM!')	截取元素

无关闭分隔符

任意无符号的对象，以逗号隔开，默认为元组，如下实例：

实例(Python 2.0+)

```
#!/usr/bin/python
print 'abc', -4.24e93, 18+6.6j, 'xyz'
x, y = 1, 2
print "Value of x , y : ", x,y
```

以上实例运行结果：

```
abc -4.24e+93 (18+6.6j) xyz
Value of x , y : 1 2
```

元组内置函数

Python元组包含了以下内置函数

序号	方法及描述
1	cmp(tuple1, tuple2) 比较两个元组元素。
2	len(tuple) 计算元组元素个数。
3	max(tuple) 返回元组中元素最大值。
4	min(tuple) 返回元组中元素最小值。
5	tuple(seq) 将列表转换为元组。

Python 字典(Dictionary)

字典是另一种可变容器模型，且可存储任意类型对象。

字典的每个键值 `key:value` 对用冒号 `:` 分割，每个键值对之间用逗号 `,` 分割，整个字典包括在花括号 `{}` 中 ,格式如下所示：

```
d = {key1 : value1, key2 : value2 }
```

注意：`dict` 作为 Python 的关键字和内置函数，变量名不建议命名为 **dict**。

键一般是唯一的，如果重复最后的一个键值对会替换前面的，值不需要唯一。

```
>>> tinydict = {'a': 1, 'b': 2, 'b': '3'}
>>> tinydict['b']
```

```
'3'
>>> tinydict
{'a': 1, 'b': '3'}
```

值可以取任何数据类型，但键必须是不可变的，如字符串，数字或元组。

一个简单的字典实例：

```
tinydict = {'Alice': '2341', 'Beth': '9102', 'Cecil': '3258'}
```

也可如此创建字典：

```
tinydict1 = { 'abc': 456 }
tinydict2 = { 'abc': 123, 98.6: 37 }
```

访问字典里的值

把相应的键放入熟悉的方括弧，如下实例:

实例

```
#!/usr/bin/python
tinydict = {'Name': 'Zara', 'Age': 7, 'Class': 'First'}
print "tinydict['Name']: ", tinydict['Name']
print "tinydict['Age']: ", tinydict['Age']
```

以上实例输出结果：

```
tinydict['Name']:  Zara
tinydict['Age']:  7
```

如果用字典里没有的键访问数据，会输出错误如下：

实例

```
#!/usr/bin/python
tinydict = {'Name': 'Runoob', 'Age': 7, 'Class': 'First'}
print "tinydict['Alice']: ", tinydict['Alice']
```

以上实例输出结果：

```
tinydict['Alice']:
Traceback (most recent call last):
  File "test.py", line 5, in <module>
    print "tinydict['Alice']: ", tinydict['Alice']
KeyError: 'Alice'
```

修改字典

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:

实例

```
#!/usr/bin/python
# -*- coding: UTF-8 -*-
tinydict = {'Name': 'Zara', 'Age': 7, 'Class': 'First'}
tinydict['Age'] = 8 # 更新
tinydict['School'] = "RUNOOB" # 添加
print "tinydict['Age']: ", tinydict['Age']
print "tinydict['School']: ", tinydict['School']
```

以上实例输出结果：

```
tinydict['Age']:  8
tinydict['School']:  RUNOOB
```

删除字典元素

能删单一的元素也能清空字典，清空只需一项操作。

显示删除一个字典用del命令，如下实例：

实例

```
#!/usr/bin/python
# -*- coding: UTF-8 -*-
tinydict = {'Name': 'Zara', 'Age': 7, 'Class': 'First'}
del tinydict['Name'] # 删除键是'Name'的条目
tinydict.clear() # 清空字典所有条目
del tinydict # 删除字典
print "tinydict['Age']: ", tinydict['Age']
print "tinydict['School']: ", tinydict['School']
```

但这会引发一个异常，因为用del后字典不再存在：

```
tinydict['Age']:
Traceback (most recent call last):
```

```
  File "test.py", line 10, in <module>
    print "tinydict['Age']: ", tinydict['Age']
NameError: name 'tinydict' is not defined
```

注：del() 方法后面也会讨论。

字典键的特性

字典值可以没有限制地取任何 python 对象，既可以是标准的对象，也可以是用户定义的，但键不行。

两个重要的点需要记住：

1）不允许同一个键出现两次。创建时如果同一个键被赋值两次，后一个值会被记住，如下实例：

实例

```
#!/usr/bin/python
tinydict = {'Name': 'Runoob', 'Age': 7, 'Name': 'Manni'}
print "tinydict['Name']: ", tinydict['Name']
```

以上实例输出结果：

```
tinydict['Name']:  Manni
```

2）键必须不可变，所以可以用数字，字符串或元组充当，所以用列表就不行，如下实例：

实例

```
#!/usr/bin/python
tinydict = {['Name']: 'Zara', 'Age': 7}
print "tinydict['Name']: ", tinydict['Name']
```

以上实例输出结果：

```
Traceback (most recent call last):
  File "test.py", line 3, in <module>
    tinydict = {['Name']: 'Zara', 'Age': 7}
TypeError: unhashable type: 'list'
```

字典内置函数&方法

Python字典包含了以下内置函数：

序号	函数及描述
1	cmp(dict1, dict2) 比较两个字典元素。
2	len(dict) 计算字典元素个数，即键的总数。
3	str(dict) 输出字典可打印的字符串表示。
4	type(variable) 返回输入的变量类型，如果变量是字典就返回字典类型。

Python字典包含了以下内置方法：

序号	函数及描述
1	dict.clear() 删除字典内所有元素
2	dict.copy() 返回一个字典的浅复制
3	dict.fromkeys(seq[, val]) 创建一个新字典，以序列 seq 中元素做字典的键，val 为字典所有键对应的初始值
4	dict.get(key, default=None) 返回指定键的值，如果值不在字典中返回default值
5	dict.has_key(key) 如果键在字典dict里返回true，否则返回false
6	dict.items() 以列表返回可遍历的(键, 值) 元组数组
7	dict.keys() 以列表返回一个字典所有的键
8	dict.setdefault(key, default=None)

	和get()类似, 但如果键不存在于字典中，将会添加键并将值设为default
9	dict.update(dict2) 把字典dict2的键/值对更新到dict里
10	dict.values() 以列表返回字典中的所有值
11	pop(key[,default]) 删除字典给定键 key 所对应的值，返回值为被删除的值。key值必须给出。 否则，返回default值。
12	popitem() 返回并删除字典中的最后一对键和值。