python脚本式编程教学

Python 基础语法

在本章中我们将来学习 Python 的基础语法，让你快速学会 Python 编程。

Python 保留字符

下面的列表显示了在Python中的保留字。这些保留字不能用作常数或变数，或任何其他标识符名称。

所有 Python 的关键字只包含小写字母。

and	exec	not
assert	finally	or
break	for	pass
class	from	print
continue	global	raise
def	if	return
del	import	try
elif	in	while
else	is	with
except	lambda	yield

行和缩进

学习 Python 与其他语言最大的区别就是，Python 的代码块不使用大括号 **{}** 来控制类，函数以及其他逻辑判断。python 最具特色的就是用缩进来写模块。

缩进的空白数量是可变的，但是所有代码块语句必须包含相同的缩进空白数量，这个必须严格执行。

以下实例缩进为四个空格:

```
if True:
    print ("True")
else:
    print ("False")
```

多行语句

Python语句中一般以新行作为语句的结束符。

但是我们可以使用斜杠（ \）将一行的语句分为多行显示，如下所示：

```
total = item_one + \
        item_two + \
        item_three
```

语句中包含 [], {} 或 () 括号就不需要使用多行连接符。如下实例：

```
days = ['Monday', 'Tuesday', 'Wednesday',
        'Thursday', 'Friday']
```

Python 引号

Python 可以使用引号(')、双引号(")、三引号(''' 或 """) 来表示字符串，引号的开始与结束必须是相同类型的。

其中三引号可以由多行组成，编写多行文本的快捷语法，常用于文档字符串，在文件的特定地点，被当做注释。

```
word = 'word'
sentence = "这是一个句子。"
paragraph = """这是一个段落。
包含了多个语句"""
```

Python注释

python中单行注释采用 # 开头。

```
**print** ("Hello, Python!") # 第二个注释
```

输出结果：

```
Hello, Python!
```

注释可以在语句或表达式行末：

```
name = "Madisetti" # 这是一个注释
```

python 中多行注释使用三个单引号(''')或三个双引号(""")。

Python空行

函数之间或类的方法之间用空行分隔，表示一段新的代码的开始。类和函数入口之间也用一行空行分隔，以突出函数入口的开始。

空行与代码缩进不同，空行并不是Python语法的一部分。书写时不插入空行，Python解释器运行也不会出错。但是空行的作用在于分隔两段不同功能或含义的代码，便于日后代码的维护或重构。

记住：空行也是程序代码的一部分。

Python算术运算符

以下假设变量： **a=10，b=20**：

运算符	描述	实例
+	加 - 两个对象相加	a + b 输出结果 30
-	减 - 得到负数或是一个数减去另一个数	a - b 输出结果 -10
*	乘 - 两个数相乘或是返回一个被重复若干次的字符串	a * b 输出结果 200
/	除 - x除以y	b / a 输出结果 2
%	取模 - 返回除法的余数	b % a 输出结果 0
**	幂 - 返回x的y次幂	a**b 为10的20次方， 输出结果 100000000000000000000
//	取整除 - 返回商的整数部分（**向下取整**）	`>>> 9//2 4 >>> -9//2 -5`

Python比较运算符和C语言完全相同，这里不列举了

Python赋值运算符

以下假设变量a为10，变量b为20：

运算符	描述	实例
=	简单的赋值运算符	c = a + b 将 a + b 的运算结果赋值为 c
+=	加法赋值运算符	c += a 等效于 c = c + a
-=	减法赋值运算符	c -= a 等效于 c = c - a
*=	乘法赋值运算符	c *= a 等效于 c = c * a
/=	除法赋值运算符	c /= a 等效于 c = c / a
%=	取模赋值运算符	c %= a 等效于 c = c % a
**=	幂赋值运算符	c ** =a 等效于 c = c ** a
//=	取整除赋值运算符	c //= a 等效于 c = c // a

Python位运算和C语言一致，在这里也跳过

Python逻辑运算符

Python语言支持逻辑运算符，以下假设变量 a 为 10, b为 20:

运算符	逻辑表达式	描述	实例
and	x and y	布尔"与" - 如果 x 为 False，x and y 返回 False，否则它返回 y 的计算值。	(a and b) 返回 20。
or	x or y	布尔"或" - 如果 x 是非 0，它返回 x 的值，否则它返回 y 的计算值。	(a or b) 返回 10。
not	not x	布尔"非" - 如果 x 为 True，返回 False 。如果 x 为 False，它返回 True。	not(a and b) 返回 False

Python成员运算符

除了以上的一些运算符之外，Python还支持成员运算符，测试实例中包含了一系列的成员，包括字符串，列表或元组。

运算符	描述	实例
in	如果在指定的序列中找到值返回 True，否则返回 False。	x 在 y 序列中 , 如果 x 在 y 序列中返回 True。
not in	如果在指定的序列中没有找到值返回 True，否则返回 False。	x 不在 y 序列中 , 如果 x 不在 y 序列中返回 True。

Python运算符优先级

以下表格列出了从最高到最低优先级的所有运算符：

运算符	描述
**	指数 (最高优先级)
~ + -	按位翻转, 一元加号和减号 (最后两个的方法名为 +@ 和 -@)
* / % //	乘，除，取模和取整除
+ -	加法减法
>> <<	右移，左移运算符
&	位 'AND'
^ |	位运算符
<= < > >=	比较运算符
<> == !=	等于运算符
= %= /= //= -= += *= **=	赋值运算符
is is not	身份运算符
in not in	成员运算符
not and or	逻辑运算符

Python 条件语句

Python条件语句是通过一条或多条语句的执行结果（True或者False）来决定执行的代码块。

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

Python 编程中 if 语句用于控制程序的执行，基本形式为：

```
if 判断条件：
    执行语句……
else：
    执行语句……
```

其中"判断条件"成立时（非零），则执行后面的语句，而执行内容可以多行，以缩进来区分表示同一范围。

else 为可选语句，当需要在条件不成立时执行内容则可以执行相关语句。

当判断条件为多个值时，可以使用以下形式：

```
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……
```

由于 python 并不支持 switch 语句，所以多个条件判断，只能用 elif 来实现，如果判断需要多个条件需同时判断时，可以使用 or （或），表示两个条件有一个成立时判断条件成功；使用 and （与）时，表示只有两个条件同时成立的情况下，判断条件才成功。

Python 循环语句

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

循环类型	描述
while 循环	在给定的判断条件为 true 时执行循环体，否则退出循环体。
for 循环	重复执行语句
嵌套循环	你可以在while循环体中嵌套for循环

Python While 循环语句

Python 编程中 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……
```

---

while循环使用 else 语句

在 python 中，while … else 在循环条件为 false 时执行 else 语句块：

Python for 循环语句

Python 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。

语法：

for循环的语法格式如下：

```
for iterating_var in sequence:
   statements(s)
```

for循环使用 else 语句

在 python 中，for … else 表示这样的意思，for 中的语句和普通的没有区别，else 中的语句会在循环正常执行完（即 for 不是通过 break 跳出而中断的）的情况下执行，while … else 也是一样。

循环控制语句

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：

控制语句	描述
break 语句	在语句块执行过程中终止循环，并且跳出整个循环
continue 语句	在语句块执行过程中终止当前循环，跳出该次循环，执行下一次循环。
pass 语句	pass是空语句，是为了保持程序结构的完整性。

Python 变量类型

变量存储在内存中的值，这就意味着在创建变量时会在内存中开辟一个空间。

基于变量的数据类型，解释器会分配指定内存，并决定什么数据可以被存储在内存中。

因此，变量可以指定不同的数据类型，这些变量可以存储整数，小数或字符。

标准数据类型

在内存中存储的数据可以有多种类型。

例如，一个人的年龄可以用数字来存储，他的名字可以用字符来存储。

Python 定义了一些标准类型，用于存储各种类型的数据。

变量赋值实例：

```
a=b=c=1
aa,bb,cc=11,22,33
str1,num1,bool1="114",514,3.14
```

Python有五个标准的数据类型：

- Numbers（数字）
- String（字符串）
- List（列表）
- Tuple（元组）
- Dictionary（字典）

Python数据类型转换

有时候，我们需要对数据内置的类型进行转换，数据类型的转换，你只需要将数据类型作为函数名即可。

以下几个内置的函数可以执行数据类型之间的转换。这些函数返回一个新的对象，表示转换的值。

函数	描述
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
set(s)	转换为可变集合
dict(d)	创建一个字典。d 必须是一个序列 (key,value)元组。
frozenset(s)	转换为不可变集合
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字符串

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，yy代表的字符，例如：\o12代表换行
\xyy	十六进制数，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 - 如果字符串中不包含给定的字符返回 True	>>>"M" not in a 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	看下面

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(+)
	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)

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）。

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。

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

方法	描述
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()，不过是从右边开始.
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+ 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

方法	描述
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（列表） 是 Python 中使用**最频繁**的数据类型。

列表可以完成大多数集合类的数据结构实现。它支持字符，数字，字符串甚至可以包含列表（即嵌套）。

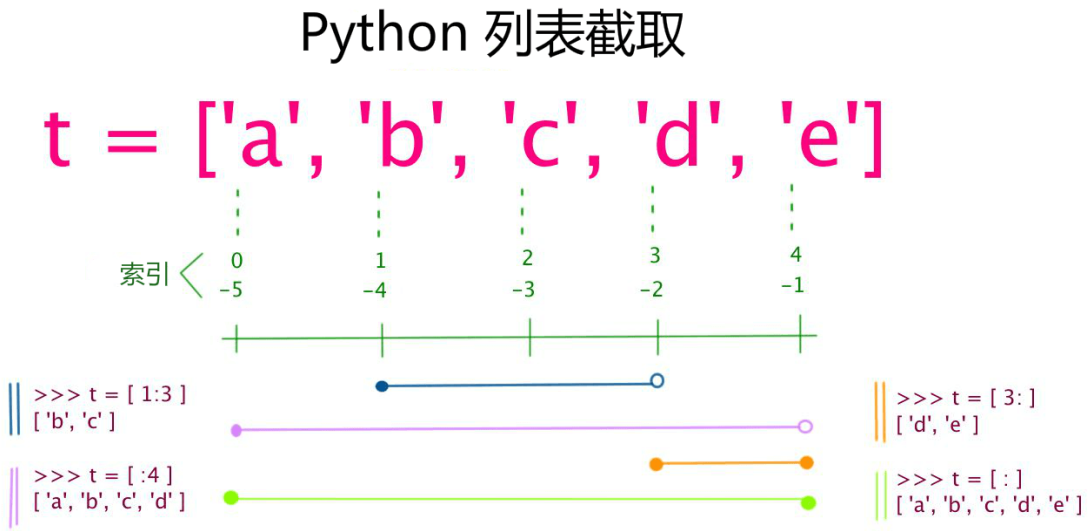
创建和使用列表

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"]
```

列表用 **[ ]** 标识，是 python 最通用的复合数据类型。**(就像数组)**

列表中值的切割也可以用到变量 **[头下标:尾下标]** ，就可以截取相应的列表，从左到右索引默认 0 开始，从右到左索引默认 -1 开始，下标可以为空表示取到头或尾。



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。

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。

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。

字典跳过，想学的请自学

Python 函数

函数是组织好的，可重复使用的，用来实现单一，或相关联功能的代码段。

函数能提高应用的模块性，和代码的重复利用率。你已经知道Python提供了许多内建函数，比如print()。但你也可以自己创建函数，这被叫做用户自定义函数。

定义一个函数

你可以定义一个由自己想要功能的函数，以下是简单的规则：

- 函数代码块以 **def** 关键词开头，后接函数标识符名称和圆括号**()**。
- 任何传入参数和自变量必须放在圆括号中间。圆括号之间可以用于定义参数。
- 函数的第一行语句可以选择性地使用文档字符串—用于存放函数说明。
- 函数内容以冒号起始，并且缩进。
- **return [表达式]** 结束函数，选择性地返回一个值给调用方。不带表达式的return相当于返回None。

```
def functionname( parameters ):
   "函数_文档字符串"
   function_suite
   return [expression]
```

函数调用

定义一个函数只给了函数一个名称，指定了函数里包含的参数，和代码块结构。

这个函数的基本结构完成以后，你可以通过另一个函数调用执行，也可以直接从Python提示符执行。

如下实例调用了printme（）函数

```
# 定义函数
def printme( str ):
   "打印任何传入的字符串"
   print str
   return

# 调用函数
printme("我要调用用户自定义函数!")
printme("再次调用同一函数")
```

Python 内置函数

		内置函数		
abs()	divmod()	input()	open()	staticmethod()
all()	enumerate()	int()	ord()	str()
any()	eval()	isinstance()	pow()	sum()
basestring()	execfile()	issubclass()	print()	super()
bin()	file()	iter()	property()	tuple()
bool()	filter()	len()	range()	type()
bytearray()	float()	list()	raw_input()	unichr()
callable()	format()	locals()	reduce()	unicode()
chr()	frozenset()	long()	reload()	vars()
classmethod()	getattr()	map()	repr()	xrange()
cmp()	globals()	max()	reverse()	zip()
compile()	hasattr()	memoryview()	round()	**import**()
complex()	hash()	min()	set()	
delattr()	help()	next()	setattr()	
dict()	hex()	object()	slice()	
dir()	id()	oct()	sorted()	exec 内置表达式