目录

[Python3基础语法 1](#_Toc520390548)

[Python3基本数据类型 2](#_Toc520390549)

[Python3高级数据类型 5](#_Toc520390550)

[String字符串 5](#_Toc520390551)

[List列表 8](#_Toc520390552)

[Tuple元组 9](#_Toc520390553)

[Set集合 9](#_Toc520390554)

[Dictionary字典 10](#_Toc520390555)

[Python3解释器 11](#_Toc520390556)

[Python3运算符 11](#_Toc520390557)

[Python3 条件控制 13](#_Toc520390558)

[Python3 循环语句 13](#_Toc520390559)

[Python3 迭代器与生成器 14](#_Toc520390560)

[Python3 函数 15](#_Toc520390561)

[参数传递 15](#_Toc520390562)

[参数 16](#_Toc520390563)

[匿名函数和lambda 16](#_Toc520390564)

[变量作用域 16](#_Toc520390565)

[Python3 模块mod 17](#_Toc520390566)

[Python3 输入和输出 19](#_Toc520390567)

[print()函数 19](#_Toc520390568)

[format()函数 19](#_Toc520390569)

[读取键盘输入 20](#_Toc520390570)

[创建文件读写对象file 20](#_Toc520390571)

[pickle模块与数据序列化 21](#_Toc520390572)

[Python3 错误和异常 21](#_Toc520390573)

[Python3 面向对象 23](#_Toc520390574)

[Python3 字符串匹配与正则表达式 25](#_Toc520390575)

[Python3标准库概览 28](#_Toc520390576)

[os模块 28](#_Toc520390577)

[glob模块 28](#_Toc520390578)

[sys模块 28](#_Toc520390579)

[math模块 28](#_Toc520390580)

[random模块 28](#_Toc520390581)

红色 重点，多为与Java语言不同之处

# Python3基础语法

标识符

* 第一个字符必须是字母表中字母或下划线 **\_** 。

以双下划线\_\_为起始和结束的变量很多已经被赋予系统级定义。

* 标识符的其他的部分由字母、数字和下划线组成。
* 标识符对大小写敏感。

在 Python 3 中，非 ASCII 标识符也是允许的了。

python保留字

保留字即关键字，我们不能把它们用作任何标识符名称。Python 的标准库提供了一个 keyword 模块，可以输出当前版本的所有关键字：

>>> import keyword

>>> keyword.kwlist

['False', 'None', 'True', 'and', 'as', 'assert', 'break', 'class', 'continue', 'def', 'del', 'elif', 'else', 'except', 'finally', 'for', 'from', 'global', 'if', 'import', 'in', 'is', 'lambda', 'nonlocal', 'not', 'or', 'pass', 'raise', 'return', 'try', 'while', 'with', 'yield']

注释

Python中单行注释以 **#** 开头

多行注释可以用多个 # 号，还有 ''' 和 """，即一对3个连续的单引号或双引号之间的内容视为注释块

代码块的行与缩进

python最具特色的就是使用缩进来表示代码块，不需要使用大括号 **{}** 。

缩进的空格数是可变的，但是同一个代码块的语句必须包含相同的缩进空格数。

多行语句

Python 通常是一行写完一条语句，但如果语句很长，我们可以使用反斜杠(\)来实现多行语句，在 [], {}, 或 () 中的多行语句，不需要使用反斜杠(\)。也可以使用 **""""""** 或者 **''''''**包裹语句块跨越多行。

同一行显示多条语句

Python可以在同一行中使用多条语句，语句之间使用分号(;)分割。

import 与 from...import

在 python 用 import 或者 from...import 来导入相应的模块。

将整个模块(somemodule)导入，格式为： import somemodule

从某个模块中导入某个函数,格式为： from somemodule import somefunction

从某个模块中导入多个函数,格式为： from somemodule import firstfunc, secondfunc, thirdfunc

将某个模块中的全部函数导入，格式为： from somemodule import \*

# Python3基本数据类型

Python 中的变量不需要声明。每个变量在使用前都必须赋值，变量赋值以后该变量才会被创建。可以理解为第一次为变量赋值即为对其的声明或指派assignment，可以使用常量或其他变量及表达式来完成该赋值过程。等号（=）用来给变量赋值。等号（=）运算符左边是一个变量名,等号（=）运算符右边是存储在变量中的值。变量就是变量，它没有类型，我们所说的"类型"是变量所指的内存中对象的类型。一个变量可以通过赋值指向不同类型的对象。

多个变量赋值

Python允许你同时为多个变量赋值。例如：

a = b = c = 1

以上实例，创建一个整型对象，值为 1，从后向前赋值，三个变量都指向同一个内存地址。

也可以为多个对象指定多个变量。例如：

a, b, c = 1, 2, "runoob"

以上实例，两个整型对象 1 和 2 的分配给变量 a 和 b，字符串对象 "runoob" 分配给变量 c。

多变量同时赋值时，实际上先计算右边表达式，然后同时赋值给左边，举例：

a, b = b, a+b

等价于：

n = b

b = a + b

a = n

标准数据类型

Python3 中有六个标准的数据类型：

* Number（数字）  
  python中数字有四种类型：

①int (整数), 如 1, 只有一种整数类型 int，表示为长整型，没有 python2 中的 Long。可使用0x和0o作为前缀表示十六进制或八进制整数。

②bool (布尔), 如 True。

③float (浮点数), 如 1.23、3E-2

④complex (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中无double类型，格式化输出时d为整数标识。

* String（字符串）
* List（列表）
* Tuple（元组）
* Set（集合）
* Dictionary（字典）

Python3 的六个标准数据类型中：

* 不可变数据（3 个）：Number（数字）、String（字符串）、Tuple（元组）；
* 可变数据（3 个）：List（列表）、Dictionary（字典）、Set（集合）。

注：在交互模式中，最后被输出的表达式结果被赋值给变量 **\_**。此处， **\_**变量应被用户视为只读变量。可用于后续代码中直接引用。

删除变量引用

可以使用del语句删除一些对象引用。del语句的语法是：

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

可以通过使用del语句删除单个或多个对象。例如：del var\_a, var\_b

数学函数

|  |  |
| --- | --- |
| 函数 | 返回值 ( 描述 ) |
| [abs(x)](http://www.runoob.com/python3/python3-func-number-abs.html) | 返回数字的绝对值，如abs(-10) 返回 10 |
| [ceil(x)](http://www.runoob.com/python3/python3-func-number-ceil.html) | 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。 Python 3 已废弃 。使用 使用 (x>y)-(x<y) 替换。 |
| [exp(x)](http://www.runoob.com/python3/python3-func-number-exp.html) | 返回e的x次幂(ex),如math.exp(1) 返回2.718281828459045 |
| [fabs(x)](http://www.runoob.com/python3/python3-func-number-fabs.html) | 返回数字的绝对值，如math.fabs(-10) 返回10.0 |
| [floor(x)](http://www.runoob.com/python3/python3-func-number-floor.html) | 返回数字的下舍整数，如math.floor(4.9)返回 4 |
| [log(x)](http://www.runoob.com/python3/python3-func-number-log.html) | 如math.log(math.e)返回1.0,math.log(100,10)返回2.0 |
| [log10(x)](http://www.runoob.com/python3/python3-func-number-log10.html) | 返回以10为基数的x的对数，如math.log10(100)返回 2.0 |
| [max(x1, x2,...)](http://www.runoob.com/python3/python3-func-number-max.html) | 返回给定参数的最大值，参数可以为序列。 |
| [min(x1, x2,...)](http://www.runoob.com/python3/python3-func-number-min.html) | 返回给定参数的最小值，参数可以为序列。 |
| [modf(x)](http://www.runoob.com/python3/python3-func-number-modf.html) | 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)](http://www.runoob.com/python3/python3-func-number-pow.html) | x\*\*y 运算后的值。 |
| [round(x [,n])](http://www.runoob.com/python3/python3-func-number-round.html) | 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)](http://www.runoob.com/python3/python3-func-number-sqrt.html) | 返回数字x的平方根。 |

随机数函数

|  |  |
| --- | --- |
| 函数 | 描述 |
| [choice(seq)](http://www.runoob.com/python3/python3-func-number-choice.html) | 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])](http://www.runoob.com/python3/python3-func-number-randrange.html) | 从指定范围内，按指定基数递增的集合中获取一个随机数，基数缺省值为1 |
| [random()](http://www.runoob.com/python3/python3-func-number-random.html) | 随机生成下一个实数，它在[0,1)范围内。 |
| [seed([x])](http://www.runoob.com/python3/python3-func-number-seed.html) | 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)](http://www.runoob.com/python3/python3-func-number-shuffle.html) | 将序列的所有元素随机排序 |
| [uniform(x, y)](http://www.runoob.com/python3/python3-func-number-uniform.html) | 随机生成下一个实数，它在[x,y]范围内。 |

三角函数

|  |  |
| --- | --- |
| 函数 | 描述 |
| [acos(x)](http://www.runoob.com/python3/python3-func-number-acos.html) | 返回x的反余弦弧度值。 |
| [asin(x)](http://www.runoob.com/python3/python3-func-number-asin.html) | 返回x的反正弦弧度值。 |
| [atan(x)](http://www.runoob.com/python3/python3-func-number-atan.html) | 返回x的反正切弧度值。 |
| [atan2(y, x)](http://www.runoob.com/python3/python3-func-number-atan2.html) | 返回给定的 X 及 Y 坐标值的反正切值。 |
| [cos(x)](http://www.runoob.com/python3/python3-func-number-cos.html) | 返回x的弧度的余弦值。 |
| [hypot(x, y)](http://www.runoob.com/python3/python3-func-number-hypot.html) | 返回欧几里德范数 sqrt(x\*x + y\*y)。 |
| [sin(x)](http://www.runoob.com/python3/python3-func-number-sin.html) | 返回的x弧度的正弦值。 |
| [tan(x)](http://www.runoob.com/python3/python3-func-number-tan.html) | 返回x弧度的正切值。 |
| [degrees(x)](http://www.runoob.com/python3/python3-func-number-degrees.html) | 将弧度转换为角度,如degrees(math.pi/2) ， 返回90.0 |
| [radians(x)](http://www.runoob.com/python3/python3-func-number-radians.html) | 将角度转换为弧度 |

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）。 |

数据类型转换

|  |  |
| --- | --- |
| 函数 | 描述 |
| int(x [,base]) | 将x转换为一个整数 |
| float(x) | 将x转换到一个浮点数 |
| complex(real [,imag]) | 创建一个复数 |
| [str(x)](http://www.runoob.com/python3/python-func-str.html) | 将对象 x 转换为字符串 |
| [repr(x)](http://www.runoob.com/python3/python-func-repr.html) | 将对象 x 转换为表达式字符串 |
| [eval(str)](http://www.runoob.com/python3/python-func-eval.html) | 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)](http://www.runoob.com/python3/python3-func-tuple.html) | 将序列 s 转换为一个元组 |
| [list(s)](http://www.runoob.com/python3/python3-att-list-list.html) | 将序列 s 转换为一个列表 |
| [set(s)](http://www.runoob.com/python3/python-func-set.html) | 转换为可变集合 |
| [dict(d)](http://www.runoob.com/python3/python-func-dict.html) | 创建一个字典。d 必须是一个序列 (key,value)元组。 |
| [frozenset(s)](http://www.runoob.com/python3/python-func-frozenset.html) | 转换为不可变集合 |
| [chr(x)](http://www.runoob.com/python3/python-func-chr.html) | 将一个整数转换为一个字符 |
| [ord(x)](http://www.runoob.com/python3/python-func-ord.html) | 将一个字符转换为它的整数值 |
| [hex(x)](http://www.runoob.com/python3/python-func-hex.html) | 将一个整数转换为一个十六进制字符串 |
| [oct(x)](http://www.runoob.com/python3/python-func-oct.html) | 将一个整数转换为一个八进制字符串 |

# Python3高级数据类型

## String字符串

* python中单引号和双引号使用完全相同。
* 使用三引号('''或""")可以指定一个多行字符串。
* 反斜杠\可以用来转义，使用r或R可以让反斜杠不发生转义。。 如 r"this is a line with \n" 则\n会显示，并不是换行。
* 按字面意义级联字符串，如"this " "is " "string"会被自动转换为this is string。
* 字符串可以用 + 运算符连接在一起，用 \* 运算符重复，即 \* 理解为乘号，后接希望 \* 前字符串的重复次数。
* Python 中的字符串有两种索引方式，从左往右以 0 开始，从右往左以 -1 开始。
* Python中的字符串不能改变。向一个索引位置赋值，比如word[0] = 'm'会导致错误。
* Python 没有单独的字符类型，一个字符就是长度为 1 的字符串。
* 字符串的截取的语法格式如下：变量[头下标:尾下标]。截取结果的索引区间为[头下标,尾下标)，即不包括尾下标索引的元素。若省略其中某一下标，则视为从头或至尾。
* 在Python3中，所有的字符串都是Unicode字符串。

Python转义字符

|  |  |
| --- | --- |
| 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字符串格式化（旧式）

旧式字符串格式化以%为关键标志，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) |

此处只给出一个使用例子，具体其他使用方法见官方文档。

>>> print ("我叫 %s 今年 %d 岁!" % ('小明', 10))

我叫 小明 今年 10 岁!

python3中可以使用format函数和相关的新型格式化机制，详见输入输出部分知识。

Python 的字符串内建函数

|  |  |
| --- | --- |
| 序号 | 方法及描述 |
| 1 | [capitalize()](http://www.runoob.com/python3/python3-string-capitalize.html)将字符串的第一个字符转换为大写 |
| 2 | [center(width, fillchar)](http://www.runoob.com/python3/python3-string-center.html)返回一个指定的宽度 width 居中的字符串，fillchar 为填充的字符，默认为空格。 |
| 3 | [count(str, beg= 0,end=len(string))](http://www.runoob.com/python3/python3-string-count.html)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 |
| 4 | [bytes.decode(encoding="utf-8", errors="strict")](http://www.runoob.com/python3/python3-string-decode.html)Python3 中没有 decode 方法，但我们可以使用 bytes 对象的 decode() 方法来解码给定的 bytes 对象，这个 bytes 对象可以由 str.encode() 来编码返回。 |
| 5 | [encode(encoding='UTF-8',errors='strict')](http://www.runoob.com/python3/python3-string-encode.html)以 encoding 指定的编码格式编码字符串，如果出错默认报一个ValueError 的异常，除非 errors 指定的是'ignore'或者'replace' |
| 6 | [endswith(suffix, beg=0, end=len(string))](http://www.runoob.com/python3/python3-string-endswith.html)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. |
| 7 | [expandtabs(tabsize=8)](http://www.runoob.com/python3/python3-string-expandtabs.html)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 。 |
| 8 | [find(str, beg=0 end=len(string))](http://www.runoob.com/python3/python3-string-find.html)检测 str 是否包含在字符串中，如果指定范围 beg 和 end ，则检查是否包含在指定范围内，如果包含返回开始的索引值，否则返回-1 |
| 9 | [index(str, beg=0, end=len(string))](http://www.runoob.com/python3/python3-string-index.html)跟find()方法一样，只不过如果str不在字符串中会报一个异常. |
| 10 | [isalnum()](http://www.runoob.com/python3/python3-string-isalnum.html)如果字符串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 |
| 11 | [isalpha()](http://www.runoob.com/python3/python3-string-isalpha.html)如果字符串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 |
| 12 | [isdigit()](http://www.runoob.com/python3/python3-string-isdigit.html)如果字符串只包含数字则返回 True 否则返回 False.. |
| 13 | [islower()](http://www.runoob.com/python3/python3-string-islower.html)如果字符串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 |
| 14 | [isnumeric()](http://www.runoob.com/python3/python3-string-isnumeric.html)如果字符串中只包含数字字符，则返回 True，否则返回 False |
| 15 | [isspace()](http://www.runoob.com/python3/python3-string-isspace.html)如果字符串中只包含空白，则返回 True，否则返回 False. |
| 16 | [istitle()](http://www.runoob.com/python3/python3-string-istitle.html)如果字符串是标题化的(见 title())则返回 True，否则返回 False |
| 17 | [isupper()](http://www.runoob.com/python3/python3-string-isupper.html)如果字符串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 |
| 18 | [join(seq)](http://www.runoob.com/python3/python3-string-join.html)以指定字符串作为分隔符，将 seq 中所有的元素(的字符串表示)合并为一个新的字符串 |
| 19 | [len(string)](http://www.runoob.com/python3/python3-string-len.html)返回字符串长度 |
| 20 | [ljust(width[, fillchar])](http://www.runoob.com/python3/python3-string-ljust.html)返回一个原字符串左对齐,并使用 fillchar 填充至长度 width 的新字符串，fillchar 默认为空格。 |
| 21 | [lower()](http://www.runoob.com/python3/python3-string-lower.html)转换字符串中所有大写字符为小写. |
| 22 | [lstrip()](http://www.runoob.com/python3/python3-string-lstrip.html)截掉字符串左边的空格或指定字符。 |
| 23 | [maketrans()](http://www.runoob.com/python3/python3-string-maketrans.html)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。 |
| 24 | [max(str)](http://www.runoob.com/python3/python3-string-max.html)返回字符串 str 中最大的字母。 |
| 25 | [min(str)](http://www.runoob.com/python3/python3-string-min.html)返回字符串 str 中最小的字母。 |
| 26 | [replace(old, new [, max])](http://www.runoob.com/python3/python3-string-replace.html)把 将字符串中的 str1 替换成 str2,如果 max 指定，则替换不超过 max 次。 |
| 27 | [rfind(str, beg=0,end=len(string))](http://www.runoob.com/python3/python3-string-rfind.html)类似于 find()函数，不过是从右边开始查找. |
| 28 | [rindex( str, beg=0, end=len(string))](http://www.runoob.com/python3/python3-string-rindex.html)类似于 index()，不过是从右边开始. |
| 29 | [rjust(width,[, fillchar])](http://www.runoob.com/python3/python3-string-rjust.html)返回一个原字符串右对齐,并使用fillchar(默认空格）填充至长度 width 的新字符串 |
| 30 | [rstrip()](http://www.runoob.com/python3/python3-string-rstrip.html)删除字符串字符串末尾的空格. |
| 31 | [split(str="", num=string.count(str))](http://www.runoob.com/python3/python3-string-split.html)num=string.count(str)) 以 str 为分隔符截取字符串，如果 num 有指定值，则仅截取 num 个子字符串 |
| 32 | [splitlines([keepends])](http://www.runoob.com/python3/python3-string-splitlines.html)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。 |
| 33 | [startswith(str, beg=0,end=len(string))](http://www.runoob.com/python3/python3-string-startswith.html)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。 |
| 34 | [strip([chars])](http://www.runoob.com/python3/python3-string-strip.html)在字符串上执行 lstrip()和 rstrip() |
| 35 | [swapcase()](http://www.runoob.com/python3/python3-string-swapcase.html)将字符串中大写转换为小写，小写转换为大写 |
| 36 | [title()](http://www.runoob.com/python3/python3-string-title.html)返回"标题化"的字符串,就是说所有单词都是以大写开始，其余字母均为小写(见 istitle()) |
| 37 | [translate(table, deletechars="")](http://www.runoob.com/python3/python3-string-translate.html)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 deletechars 参数中 |
| 38 | [upper()](http://www.runoob.com/python3/python3-string-upper.html)转换字符串中的小写字母为大写 |
| 39 | [zfill (width)](http://www.runoob.com/python3/python3-string-zfill.html)返回长度为 width 的字符串，原字符串右对齐，前面填充0 |
| 40 | [isdecimal()](http://www.runoob.com/python3/python3-string-isdecimal.html)检查字符串是否只包含十进制字符，如果是返回 true，否则返回 false。 |

## List列表

列表中元素的类型可以不相同，它支持数字，字符串甚至可以包含列表（所谓嵌套）。列表是写在方括号([])之间、用逗号分隔开的元素列表。如：

list = [ 'abcd', 786 , 2.23, 'runoob', 70.2 ] #只需给列表变量起个名即可，无需声明列表类型

列表同样可以被索引和截取，列表被截取后返回一个包含所需元素的新列表，但所有元素仍为原列表对应元素的引用。加号（+）是列表连接运算符，星号（\*）是重复操作。所有操作方法完全同字符串。

与Python字符串不一样的是，列表中的元素是可以改变的。可以重新为某个元素赋值，也可以截取某个区间重新赋值，若未对某些元素赋值（提供的常量列表长度不及截取的区间）或干脆使用空列表[]赋值将直接删除原列表中对应元素，而没有赋空值的说法。

删除列表元素

可以使用 del 语句来删除列表的元素，方法同删除变量引用，如del list[2]。也可以删去整个列表。但删除所有列表的元素结果不同于删去整个列表，前者的结果是一个空列表，如del list[:]。

Python列表函数&方法

Python包含以下函数:

|  |  |
| --- | --- |
| 序号 | 函数 |
| 1 | [len(list)](http://www.runoob.com/python3/python3-att-list-len.html)列表元素个数 |
| 2 | [max(list)](http://www.runoob.com/python3/python3-att-list-max.html)返回列表元素最大值 |
| 3 | [min(list)](http://www.runoob.com/python3/python3-att-list-min.html)返回列表元素最小值 |
| 4 | [list(seq)](http://www.runoob.com/python3/python3-att-list-list.html)将元组转换为列表 |

Python包含以下方法:

|  |  |
| --- | --- |
| 序号 | 方法 |
| 1 | [list.append(obj)](http://www.runoob.com/python3/python3-att-list-append.html)在列表末尾添加新的对象 |
| 2 | [list.count(obj)](http://www.runoob.com/python3/python3-att-list-count.html)统计某个元素在列表中出现的次数 |
| 3 | [list.extend(seq)](http://www.runoob.com/python3/python3-att-list-extend.html)在列表末尾一次性追加另一个序列的多个值（用新列表扩展原列表） |
| 4 | [list.index(obj)](http://www.runoob.com/python3/python3-att-list-index.html)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)](http://www.runoob.com/python3/python3-att-list-insert.html)将对象插入列表 |
| 6 | [list.pop([index=-1]])](http://www.runoob.com/python3/python3-att-list-pop.html)移除列表中一个元素（默认最后一个），并且返回该元素的值 |
| 7 | [list.remove(obj)](http://www.runoob.com/python3/python3-att-list-remove.html)移除列表中某个值的第一个匹配项 |
| 8 | [list.reverse()](http://www.runoob.com/python3/python3-att-list-reverse.html)反向列表中元素 |
| 9 | [list.sort(cmp=None, key=None, reverse=False)](http://www.runoob.com/python3/python3-att-list-sort.html)对原列表进行排序 |
| 10 | [list.clear()](http://www.runoob.com/python3/python3-att-list-clear.html)清空列表 |
| 11 | [list.copy()](http://www.runoob.com/python3/python3-att-list-copy.html)复制列表 |

列表推导式

列表推导式提供了从序列创建列表的简单途径。通常将一些操作应用于某个序列的每个元素，用其获得的结果作为生成新列表的元素，或者根据确定的判定条件创建子序列。

每个列表推导式都在一个表达式之后有零到多个 for 或 if 子句。返回结果是一个根据表达从其后的 for 和 if 上下文环境中生成出来的列表。如果希望表达式推导出一个元组，就必须使用括号。

举例：

[表达式 for 语句 if 语句 for 语句 if 语句…]

#总结果返回一个列表，表达式结果为列表的元素，后续的语句用以设定表达式中变量的遍历范围与条件。可以省略if条件，但一般每个for语句和一个表达式中的变量对应。

注意：表达式也可以是列表，即使用列表推导式可以完成嵌套列表创建。

## Tuple元组

元组（tuple）与列表使用方法几乎相同，不同之处在于元组的元素不能修改。元组写在小括号 **()** 里，元素之间用逗号隔开。元组与字符串类似，可以被索引和截取，其实，可以把字符串看作一种特殊的元组。虽然tuple的元素不可改变，但它可以包含可变的对象，比如list列表。

元组也可以使用类似列表推导式的机制。

元组可以通过运算符+进行连接组合。可以使用del删除整个元组，但不能删除单个元素。

构造包含 0 个或 1 个元素的元组比较特殊，所以有一些额外的语法规则：  
tup1 = () # 空元组  
tup2 = (20,) # 一个元素，需要在元素后添加逗号，否则括号会被视为算术运算符

无歧义的情况下，元组定义可以省略括号，如：  
tup3 = 1, 2, 3 #合法，但是不是好习惯，此例也说明列表定义不能省略[]括号。

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 | len(tuple)计算元组元素个数。 |
| 2 | max(tuple)返回元组中元素最大值。 |
| 3 | min(tuple)返回元组中元素最小值。 |
| 4 | tuple(seq)将列表转换为元组。 |

## Set集合

集合（set）是一个无序不重复元素的序列。基本功能是进行成员关系测试和删除重复元素。可以使用大括号 **{ }** 或者 **set()** 函数创建集合，注意：创建一个空集合必须用**set()** 而不是 **{ }**，因为 **{ }** 是用来创建一个空字典。集合中重复的元素被自动去掉，包括定义时就录入的重复元素。

创建格式：  
parame = {value01,value02,...}  
或者parame = set(value)

set可以使用以下运算符进行集合运算，包括差集-，并集|，交集&，反交集^。

集合也可以使用类似列表推导式的机制。

集合的基本操作

1、添加元素

语法格式如下：s.add( x )

将元素 x 添加到集合 s 中，如果元素已存在，则不进行任何操作。还有一个方法，也可以添加元素，且参数可以是列表，元组，字典等，语法格式如下：s.update( x ) # x 可以有多个，用逗号分开。

2、移除元素

语法格式如下：s.remove( x )

将元素 x 添加到集合 s 中移除，如果元素不存在，则会发生错误。此外还有一个方法也是移除集合中的元素，且如果元素不存在，不会发生错误。格式如下所示：s.discard( x )

我们也可以设置随机删除集合中的一个元素，语法格式如下：s.pop()

3、计算集合元素个数

语法格式如下：len(s)

4、清空集合

语法格式如下：s.clear() # 清空集合 s。

5、判断元素是否在集合中存在

语法格式如下：x in s

判断元素 s 是否在集合 x 中存在，存在返回 True，不存在返回 False。

## Dictionary字典

字典是一种映射类型，字典用"{ }"标识，它是一个无序的键(key) : 值(value)对集合。

键(key)必须使用不可变类型，如字符串，数字或元组。值可以使用任何数据类型。

在同一个字典中，键(key)必须是唯一的。值不必唯一。创建时如果同一个键被赋值两次，后一个值会被记住。

使用格式举例：  
dict = {} #创建一个空字典  
dict['one'] = "1 - 菜鸟教程" #插入一个键值对，索引为键  
tinydict = {'name': 'runoob','code':1, 'site': 'www.runoob.com'} #创建有初值的字典

构造函数dict()使用方法举例：  
dict([('Runoob', 1), ('Google', 2), ('Taobao', 3)]) #以键值对元组列表为参数类型  
dict(Runoob=1, Google=2, Taobao=3) #关键字参数，一般常见最简单的方式  
上述两个语句都创建字典{'Taobao': 3, 'Runoob': 1, 'Google': 2}

字典也可以使用类似列表推导式的机制。但显然必须是如下格式：  
{表达式1:表达式2 for 语句 if 语句 for 语句 if 语句…}

修改字典

插入新项与修改已有项的方法一致，都是向字典中的键赋值即可。

删除字典元素

根据字典元素可变的特性，显然也可以使用del关键词删除字典的元素，以致整个字典。可使用字典类型成员函数clear()来清空字典内容。

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 | len(dict)计算字典元素个数，即键的总数。 |
| 2 | str(dict)输出字典，以可打印的字符串表示。 |
| 3 | 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 | [radiansdict.clear()](http://www.runoob.com/python3/python3-att-dictionary-clear.html)删除字典内所有元素 |
| 2 | [radiansdict.copy()](http://www.runoob.com/python3/python3-att-dictionary-copy.html)返回一个字典的浅复制 |
| 3 | [radiansdict.fromkeys()](http://www.runoob.com/python3/python3-att-dictionary-fromkeys.html)创建新字典，以seq中元素做键，val为所有键对应的初始值 |
| 4 | [radiansdict.get(key, default=None)](http://www.runoob.com/python3/python3-att-dictionary-get.html)返回指定键的值，若值不在字典中返回default值 |
| 5 | [key in dict](http://www.runoob.com/python3/python3-att-dictionary-in.html)如果键在字典dict里返回true，否则返回false |
| 6 | [radiansdict.items()](http://www.runoob.com/python3/python3-att-dictionary-items.html)以列表返回可遍历的(键, 值) 元组数组 |
| 7 | [radiansdict.keys()](http://www.runoob.com/python3/python3-att-dictionary-keys.html)以列表返回一个字典所有的键 |
| 8 | [radiansdict.setdefault(key, default=None)](http://www.runoob.com/python3/python3-att-dictionary-setdefault.html)和get()类似, 但若键不存在，将添加键并将值设为default |
| 9 | [radiansdict.update(dict2)](http://www.runoob.com/python3/python3-att-dictionary-update.html)把字典dict2的键/值对更新到dict里 |
| 10 | [radiansdict.values()](http://www.runoob.com/python3/python3-att-dictionary-values.html)以列表返回字典中的所有值 |
| 11 | [pop(key[,default])](http://www.runoob.com/python3/python3-att-dictionary-pop.html)删除给定键对应的值，返回被删的值。key值必须给出。 否则，返回default值。 |
| 12 | [popitem()](http://www.runoob.com/python3/python3-att-dictionary-popitem.html)随机返回并删除字典中的一对键和值(一般删除末尾对)。 |

# Python3解释器

交互式编程

交互式编程即使用类似命令提示符形式的界面与操作方式来编程。

我们可以在命令提示符中输入"Python"命令来启动Python解释器，在 python 提示符（以>>>为标志）中输入python语句，然后按回车键查看运行效果。

脚本式编程

即将python语句编写为一个.py后缀的文件，使用python3命令来执行脚本。注意python3只是其中一种解释器而已。

# Python3运算符

Python算术运算符

|  |  |
| --- | --- |
| **运算符** | **描述** |
| + | 加 - 两个对象相加 |
| - | 减 - 得到负数或是一个数减去另一个数 |
| \* | 乘 - 两个数相乘或是返回一个被重复若干次的字符串 |
| / | 除 - x 除以 y |
| % | 取模 - 返回除法的余数 |
| \*\* | 幂 - 返回x的y次幂 |
| // | 取整除 - 返回商的整数部分，但是不一定是整型 |

Python比较运算符

|  |  |
| --- | --- |
| **运算符** | **描述** |
| == | 等于 - 比较对象是否相等 |
| != | 不等于 - 比较两个对象是否不相等 |
| > | 大于 - 返回x是否大于y |
| < | 小于 - 返回x是否小于y。所有比较运算符返回1表示真，返回0表示假。这分别与特殊的变量True和False等价。注意，这些变量名的大写。 |
| >= | 大于等于 - 返回x是否大于等于y。 |
| <= | 小于等于 - 返回x是否小于等于y。 |

Python赋值运算符

|  |  |
| --- | --- |
| **运算符** | **描述** |
| = | 简单的赋值运算符 |
| += | 加法赋值运算符 |
| -= | 减法赋值运算符 |
| \*= | 乘法赋值运算符 |
| /= | 除法赋值运算符 |
| %= | 取模赋值运算符 |
| \*\*= | 幂赋值运算符 |
| //= | 取整除赋值运算符 |

Python位运算符

|  |  |
| --- | --- |
| **运算符** | **描述** |
| & | 按位与运算符：参与运算的两个值,如果两个相应位都为1,则该位的结果为1,否则为0 |
| | | 按位或运算符：只要对应的二个二进位有一个为1时，结果位就为1。 |
| ^ | 按位异或运算符：当两对应的二进位相异时，结果为1 |
| ~ | 按位取反运算符：对数据的每个二进制位取反,即把1变为0,把0变为1。**~x** 类似于 **-x-1** |
| << | 左移动运算符：运算数的各二进位全部左移若干位，由"<<"右边的数指定移动的位数，高位丢弃，低位补0。 |
| >> | 右移动运算符：把">>"左边的运算数的各二进位全部右移若干位，">>"右边的数指定移动的位数 |

Python逻辑运算符

|  |  |
| --- | --- |
| 运算符 | 描述 |
| and | 布尔"与" - 如果 x 为 False，x and y 返回 False，否则它返回 y 的计算值。 |
| or | 布尔"或" - 如果 x 是 True，它返回 x 的值，否则它返回 y 的计算值。 |
| not | 布尔"非" - 如果 x 为 True，返回 False 。如果 x 为 False，它返回 True。 |

Python成员运算符

|  |  |
| --- | --- |
| 运算符 | 描述 |
| in | 如果在指定的序列中找到值返回 True，否则返回 False。可用于字符串。 |
| not in | 如果在指定的序列中没有找到值返回 True，否则返回 False。可用于字符串。 |

Python身份运算符

|  |  |
| --- | --- |
| 运算符 | 描述 |
| is | is 是判断两个标识符是不是引用自一个对象 |
| is not | is not 是判断两个标识符是不是引用自不同对象 |

**注：**   
id() 函数可用于获取对象内存地址。

is 用于判断两个变量引用对象是否为同一个， == 用于判断引用变量的值是否相等。

Python运算符优先级

|  |  |
| --- | --- |
| **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 |
| and or not | 逻辑运算符 |

# Python3 条件控制

if 语句

Python中if语句的一般形式如下所示：

if condition\_1:

statement\_block\_1

elif condition\_2:

statement\_block\_2

else:

statement\_block\_3

·Python 中用 elif 代替了 else if，所以if语句的关键字为：if – elif – else。也可以通过if语句嵌套来实现复杂的条件语句。

·在Python中没有switch – case语句。

# Python3 循环语句

while 循环

Python中while语句的一般形式：

while 判断条件:

语句

else:

语句

或

while 判断条件: 单行语句

·同样需要注意冒号和缩进。另外，在Python中没有do..while循环。

·可以使用CTRL+C来退出当前的无限循环。

·判断条件可以加或不加括号()

for 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。

for循环的一般格式如下：

for <variable> in <sequence>:

<statements>

else:

<statements>

range()函数

如果你需要遍历数字序列，可以使用内置range()函数。它会生成序列。

range(x) #返回一个[0,x)区间的整数序列

range(x,y) #返回一个[x,y)区间的整数序列

range(x,y,z) #返回一个[x,y)区间的以z为步长的整数序列

·区间可以是负数区间，步长也可以为负，此时x与y是起止点

break和continue语句及循环中的else子句

break 语句可以跳出 for 和 while 的循环体。如果你从 for 或 while 循环中终止，任何对应的循环 else 块将不执行。

continue语句被用来告诉Python跳过当前循环块中的剩余语句，然后继续进行下一轮循环。

循环语句可以有 else 子句，它在穷尽列表(以for循环)或条件变为 false (以while循环)导致循环终止时被执行,但循环被break终止时不执行。

pass 语句

Python pass是空语句，是为了保持程序结构的完整性。pass 不做任何事情，一般用做占位语句。

遍历技巧

以下几个函数返回一个处理过后的序列，常用于遍历：

①items()

for k, v in knights.items():

返回字典的键值对序列，分别赋给两个变量

②enumerate()

for i, v in enumerate(['tic', 'tac', 'toe']):

返回索引编号和对应值序列

③zip()

for q, a in zip(questions, answers):

同时遍历两个或更多序列，分别赋给每个变量

 ④reversed()

for i in reversed(range(1, 10, 2)):

将序列倒序

⑤sorted()

for f in sorted(set(basket)):

将序列排序，一般按照首字母

# Python3 迭代器与生成器

Iterator迭代器

迭代器对象从集合的第一个元素开始访问，直到所有的元素被访问完结束。迭代器只能往前不会后退。迭代器有两个基本的方法：iter() 和 next()。字符串，列表或元组对象都可用于创建迭代器。

举例：

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

>>> it = iter(list) # 创建迭代器对象

>>> print (next(it)) # 输出迭代器的下一个元素

1

迭代器对象本身也是个序列，可用于in关键词，虽然这一般和使用其本体集合对象无区别。

Generator生成器

在 Python 中，使用了 yield 的函数被称为生成器（generator）。跟普通函数不同的是，生成器是一个返回迭代器对象的函数，只能用于迭代操作，更简单点理解生成器就是一个迭代器。

在调用生成器运行的过程中，每次遇到 yield 时函数会暂停并保存当前所有的运行信息，返回 yield 的值（此时yield的作用和使用像是一个临时的return）, 并在下一次执行 next() 方法时从当前位置继续运行（第一次执行next()函数从头开始执行）。当函数结束或return时，迭代器到达尾部。

举例：

def fibonacci(n): # 生成器函数 - 斐波那契

a, b, counter = 0, 1, 0

while True:

if (counter > n): return

yield a

a, b = b, a + b

counter += 1

f = fibonacci(10) # f 是一个迭代器，由生成器返回生成

while True:

try:

print (next(f), end=" ")

except StopIteration:

sys.exit()

# Python3 函数

定义一个函数

Python 定义函数使用 def 关键字，一般格式如下：

def 函数名（参数列表）:

函数体

## 参数传递

可更改(mutable)与不可更改(immutable)对象

在 python 中，strings, tuples, 和 numbers 是不可更改的对象，而 list,dict 等则是可以修改的对象。

* 不可变类型：变量赋值 a=5 后再赋值 a=10，这里实际是新生成一个 int 值对象 10，再让 a 指向它，而 5 被丢弃，不是改变a的值，相当于新生成了a。
* 可变类型：变量赋值 la=[1,2,3,4] 后再赋值 la[2]=5 则是将 list la 的第三个元素值更改，本身la没有动，只是其内部的一部分值被修改了。

python 函数的参数传递：

* 不可变类型：类似 c++ 的值传递，如 整数、字符串、元组。如fun（a），传递的只是a的值，没有影响a对象本身。比如在 fun（a）内部修改 a 的值，只是修改另一个复制的对象，不会影响 a 本身。
* 可变类型：类似 c++ 的引用传递，如 列表，字典。如 fun（la），则是将 la 真正的传过去，修改后fun外部的la也会受影响

python 中一切都是对象，严格意义我们不能说值传递还是引用传递，我们应该说传不可变对象和传可变对象。

## 参数

以下是调用函数时可使用的参数种类：

①必需参数：必需参数须以正确的顺序传入函数。调用时的数量必须和声明时的一样。这是最一般的调用方式，直接按顺序写实参变量即可调用。

②关键字参数：使用关键字参数允许函数调用时参数的顺序与声明时不一致，因为 Python 解释器能够用参数名匹配参数值。使用“形参=实参”的方式为每个参数赋值。

③默认参数：道理同C++，函数声明时为形参赋默认值，即可在调用时省略该参数传递从而使用默认参数。

④不定长参数：类似Java中的不定长参数

接收不定长参数的函数定义的基本语法如下：

def functionname([formal\_args,] \*var\_args\_tuple ): #正式参数不是必要的，不定长参数由\*标识

#"函数\_文档字符串"，Python中习惯在函数体内首行写文档注释

function\_suite

return [expression]

加了星号 **\*** 的参数会以元组(tuple)的形式导入，存放所有未命名的变量参数，即不定长参数。如果在函数调用时没有赋值给不定长参数，它就是一个空元组。

另外，也可以使用 \*\* 标识不定长参数，加了两个星号 **\*\*** 的参数会以字典的形式导入。

声明函数时，参数中星号 **\*** 可以单独出现，例如:

def f(a,b,\*,c):

return a+b+c #如果单独出现星号， **\*** 后的参数必须用关键字传入。

## 匿名函数和lambda

python 使用 lambda 来创建匿名函数。lambda的主体是一个表达式，而不是一个代码块，仅仅只能写一行并封装有限的逻辑进去。lambda 函数拥有自己的命名空间，且不能访问自己参数列表之外或全局命名空间里的参数。

lambda 函数的语法只包含一个语句，如下：

functionname = lambda [arg1 [,arg2,.....argn]]:expression

#[]表示所有arg均可省略，而不是参数外需加[]。语句占一行。调用方法与一般函数一致。

## 变量作用域

程序的变量并不是在哪个位置都可以访问的，访问权限决定于这个变量是在哪里赋值的。

Python的作用域一共有4种，分别是：

* L （Local） 局部作用域：语句块内
* E （Enclosing） 闭包函数（内嵌函数）外的函数中：嵌套语句块外层
* G （Global） 全局作用域：
* B （Built-in） 内建作用域：

以 L –> E –> G –>B 的规则查找。

Python 中只有模块（module），类（class）以及函数（def、lambda）才会引入新的作用域，其它的代码块（如 if/elif/else/、try/except、for/while等）是不会引入新的作用域的，也就是说这些语句内定义的变量，外部也可以访问。这与Java有差别。

全局变量和局部变量

局部变量只能在其被声明的函数内部访问，而全局变量可以在整个程序范围内访问。一般情况下，在函数或局部域内部无法直接使用外部或全局变量，需先用关键字声明，见下节。但可声明同名的局部变量以使用，函数体内的对同名变量的操作对对应的全局变量不影响。

global 和 nonlocal关键字

当内部作用域想修改外部作用域的变量时，就要用到global和nonlocal关键字了。在函数体内用global关键字再次声明全局变量，如global x，无需赋值生效，即可在函数体内使用全局变量。

同理，如果要在内嵌作用域中修改外层嵌套作用域（enclosing 作用域，外层非全局作用域）中的变量则需要 nonlocal 关键字了，用法相同，只是关键字不同。

# Python3 模块mod

包

包是一种管理 Python 模块命名空间的形式，采用"点模块名称"。比如一个模块的名称是 A.B， 那么他表示一个包 A中的子模块 B 。实际上，python中的包概念和Java中的基本一致。

目录只有包含一个叫做 \_\_init\_\_.py 的文件才会被认作是一个包，主要是为了避免一些滥俗的名字（比如叫做 string）不小心的影响搜索路径中的有效模块。

模块

以.py为后缀的文件都被视为一个模块，模块文件中可以有全局变量、函数、类以及裸程序代码（不处于任何函数中的执行与操作命令）。主模块的名字永远是"\_\_main\_\_"，一个Python应用程序的主模块，应当总是使用绝对路径引用。

import导入模块

想使用 Python 源文件，只需在另一个源文件里执行 import 语句，语法如下：

import module1[, module2[,... moduleN]

#module为模块名，且包含其所在的包路径，类似Java

一个模块只会被导入一次，不管你执行了多少次import。这样可以防止导入模块被一遍又一遍地执行。（不同于C++）

模块除了方法定义，还可以包括可执行的代码。这些代码一般用来初始化这个模块。这些代码只有在第一次被导入时才会被执行。

每个模块有各自独立的符号表，在模块内部为所有的函数当作全局符号表来使用。所以，模块的作者可以放心大胆的在模块内部使用这些全局变量，而不用担心把其他用户的全局变量搞花。

使用形如import item.subitem.subsubitem的导入形式有一个限制：除了最后一项，都必须是包，而最后一项则可以是模块或者是包，但是不可以是类，函数或者变量的名字。亦即若只想导入模块中的一个函数，只能使用from import方式。

from … import 语句

from 包名/模块名 import name1[, name2[, ... nameN]]

这个命令实质上与import功能相同，只是如下节所述，这两种语句都是导入了import关键词后的一个名称而已。对于单纯的import命令，导入一个模块一般必须带有完整的包路径，那么后续引用该模块也必须带有完整的包路径，即和import关键词后的内容相同。但from import方式却可以将包路径与模块名分开书写，使得import关键词后只有单纯的模块名，此后即可单独使用该模块名，更方便。

两相比较，使用from import这种方法永远不会有错，且适用更广。事实上，这也是推荐的方法。除非是你要导入的子模块有可能和其他包的子模块重名，此时需要保留完整的包路径来引用模块。

把一个模块的所有内容全都导入到当前的命名空间也是可行的，只需使用如下声明：

from模块名 import \*

#不推荐用。因为盲目引入的其它来源的命名，很可能覆盖了已有的定义。

注：

①那些由单一下划线（\_）开头的名字无论如何不会被导入到另一个模块中。

②from 包名 import \*的方式在windows系统下，导入包内全部模块会出现问题。因为Windows是一个大小写不区分的系统。在这类平台上，没有人敢担保一个叫做 ECHO.py 的文件导入为模块 echo 还是 Echo 甚至 ECHO。  
因此，使用这样的导入语句遵循如下规则：如果包定义文件 \_\_init\_\_.py 存在一个叫做 \_\_all\_\_ 的列表变量，那么在使用 from package import \* 的时候就把这个列表中的所有名字作为子包名导入。如果 \_\_all\_\_ 没有定义，那么使用from package import \*这种语法的时候，就不会导入包里的任何子模块。只是把包和它里面定义的所有内容导入进来（可能运行\_\_init\_\_.py里定义的初始化代码）。

隐式与相对导入

举例：

from . import echo #当前路径

from .. import formats #上一层目录下

from ..filters import equalizer #返回上一层，再进入另一目录下

无论是隐式的还是显式的相对导入都是从当前模块开始的。

引用模块内容

单纯使用import加模块名的方式导入模块，这样做并没有把直接定义在fibo中的函数名称导入到当前命名空间里，只是把模块fibo的名字导入了。这样在使用其中的函数时，仍然要带模块名，并按照形如类的方式调用，使用.点号访问模块中的函数，形如访问静态成员函数。

对于import和from import语句，可以理解为都只是导入了一个import关键词后面的名称，将其纳入当前命名空间，而不是其中的所有内容。

注：可以为外部模块函数重命名或称为赋予本地名。如：

fib = fibo.fib

#之后便可在本地直接使用fib()来调用fibo模块中的fib()函数，否则须以fibo.fib()形式调用

\_\_name\_\_属性

一个模块被另一个程序第一次引入时，其主程序或纯代码部分将运行。如果我们想在模块被引入时，模块中的某一程序块不执行，我们可以用\_\_name\_\_属性来使该程序块仅在该模块自身运行时执行。

例如：

#!/usr/bin/python3

# Filename: using\_name.py

if \_\_name\_\_ == '\_\_main\_\_':

print('程序自身在运行')

else:

print('我来自另一模块')

注：\_\_name\_\_ 与 \_\_main\_\_ 底下是双下划线。

dir() 函数

内置的函数 dir() 以模块名为参数时，可以找到该模块内定义的所有名称。以一个字符串列表的形式返回。如果没有给定参数，那么 dir() 函数会罗列出当前模块定义的所有名称，亦即当前命名空间中所有以各种形式被使用的名称。

dir函数的主要作用就是返回一个模块中所有函数与变量名的列表。

被del命令完全删除的名称将不在dir()函数返回结果中显示。

help()函数

以模块名为参数，返回一个由模块文档注释行组成的手册页面。

# Python3 输入和输出

## print()函数

print()用于控制台输出，多个变量同时输出可以使用逗号分隔作为参数，如print(x,y)。

print 默认输出是换行的，如果要实现不换行要在变量末尾加上 end=""，如print( x, end=" " )。实际上，end的默认值应该是换行符，照此法可以在输出的结尾添加任何合法字符。

repr()函数

repr()的使用形如类型转换，参数置于括号中。其结果为字符串，但与str()类型转换不同。

①repr()返回的是变量的直接书写形式，如字符串为参数返回”’xxx’”，元组作参返回”(x,y,z)”

②因上述原因，repr()不对转义字符进行解释，直接输出字符。与字符串前置r或R一致。

③repr()函数仍然会先计算参数表达式的结果，再进行输出

格式对齐函数

rjust(x)：字符串对象成员函数，x为输出的字符数，原字符串不足的字符以空格代替且置于字符串前。即右对齐功能。

ljust(x)：类似上例

center()：

zfill(x)：字符串对象成员函数，类似rjust()但是仅适用于纯数字字符串，用于在数字前补0。这个函数会妥善处理负号，即0不会置于负号前。且负号与小数点均占一字符位。若x值比原字符串位数还小，则函数不做任何处理返回原串。

## format()函数

使用形式举例：

print('{}网址： "{}!"'.format('菜鸟教程', 'www.runoob.com'))

format()即字符串对象的成员函数，未经转义的{}括号及其里面的字符 (称作格式化字段) 将会被 format() 中的参数替换。

（1）格式化字段

①在括号中可以使用数字作为索引用于指向传入对象在 format() 参数中的位置，以0起始

②括号中可以使用自定义关键字，然后在format()中使用关键字参数来匹配位置。关键字对应法和数字索引对应法可以结合使用，只要不冲突。

③括号中可以使用以下几个关键字：!a (使用 ascii()), !s (使用 str()) 和 !r (使用 repr())。等效于将这些函数作用于format的参数，再输出。

④数字索引与关键字参数之后可以带冒号:后接可选的其他的更细致的格式选项，如{0:num1.num2f}。0为索引示例；num1与num2分别表示小数点前后的位数精度，二者均可省略，且num2与小数点.共存；f为输出类型标识示例，即浮点类型，也可以是d整数，s字符串等，也可省略采用默认，即s。

格式化字段可以与字典变量搭配出特别用法：

举例：

>>> table = {'Google': 1, 'Runoob': 2, 'Taobao': 3}

>>> print('Runoob: {0[Runoob]:d}; Google: {0[Google]:d}; Taobao: {0[Taobao]:d}'.format(table))

Runoob: 2; Google: 1; Taobao: 3

或如下写法作用与结果相同：

>>> table = {'Google': 1, 'Runoob': 2, 'Taobao': 3}

>>> print('Runoob: {Runoob:d}; Google: {Google:d}; Taobao: {Taobao:d}'.format(\*\*table))

Runoob: 2; Google: 1; Taobao: 3

注：旧式字符串格式化知识见python3高级数据类型中字符串string类型知识。

## 读取键盘输入

Python提供了 input() 内置函数从标准输入读入一行文本，默认的标准输入是键盘。input 可以接收一个Python表达式作为输入，并将运算结果返回。

举例：

#!/usr/bin/python3

str = input("请输入："); #input函数可以带一个参数用以提示用户输入

print ("你输入的内容是: ", str)

## 创建文件读写对象file

open() 将会返回一个 file 对象，基本语法格式如下:

open(filename, mode)

* filename：包含了你要访问的文件名称的字符串值。是一个路径，如"/tmp/foo.txt"
* mode：是一个字符(串)，决定了打开文件的模式：只读，写入，追加等。所有可取值见如下的完全列表。这个参数是非强制的，默认文件访问模式为只读(r)。

不同模式打开文件的完全列表：

|  |  |
| --- | --- |
| 模式 | 描述 |
| r | 以只读方式打开文件。文件的指针将会放在文件的开头。这是默认模式。 |
| rb | 以二进制格式打开一个文件用于只读。文件指针将会放在文件的开头。 |
| r+ | 打开一个文件用于读写。文件指针将会放在文件的开头。 |
| rb+ | 以二进制格式打开一个文件用于读写。文件指针将会放在文件的开头。 |
| w | 打开一个文件只用于写入。如果该文件已存在则打开文件，并从开头开始编辑，即原有内容会被删除。如果该文件不存在，创建新文件。 |
| wb | 以二进制格式打开一个文件只用于写入。如果该文件已存在则打开文件，并从开头开始编辑，即原有内容会被删除。如果该文件不存在，创建新文件。 |
| w+ | 打开一个文件用于读写。如果该文件已存在则打开文件，并从开头开始编辑，即原有内容会被删除。如果该文件不存在，创建新文件。 |
| wb+ | 以二进制格式打开一个文件用于读写。如果该文件已存在则打开文件，并从开头开始编辑，即原有内容会被删除。如果该文件不存在，创建新文件。 |
| a | 打开一个文件用于追加。如果该文件已存在，文件指针将会放在文件的结尾。也就是说，新的内容将会被写入到已有内容之后。如果该文件不存在，创建新文件进行写入。 |
| ab | 以二进制格式打开一个文件用于追加。如果该文件已存在，文件指针将会放在文件的结尾。也就是说，新的内容将会被写入到已有内容之后。如果该文件不存在，创建新文件进行写入。 |
| a+ | 打开一个文件用于读写。如果该文件已存在，文件指针将会放在文件的结尾。文件打开时会是追加模式。如果该文件不存在，创建新文件用于读写。 |
| ab+ | 以二进制格式打开一个文件用于追加。如果该文件已存在，文件指针将会放在文件的结尾。如果该文件不存在，创建新文件用于读写。 |

文件对象的方法

file 对象使用 open 函数来创建，下表列出了 file 对象常用的函数：

|  |  |
| --- | --- |
| 序号 | 方法及描述 |
| 1 | [file.close()](http://www.runoob.com/python3/python3-file-close.html)关闭文件。关闭后文件不能再进行读写操作。 |
| 2 | [file.flush()](http://www.runoob.com/python3/python3-file-flush.html)刷新文件内部缓冲，直接把内部缓冲区的数据立刻写入文件, 而不是被动的等待输出缓冲区写入。 |
| 3 | [file.fileno()](http://www.runoob.com/python3/python3-file-fileno.html)返回一个整型的文件描述符(file descriptor FD 整型), 可以用在如os模块的read方法等一些底层操作上。 |
| 4 | [file.isatty()](http://www.runoob.com/python3/python3-file-isatty.html)如果文件连接到一个终端设备返回 True，否则返回 False。 |
| 5 | [file.next()](http://www.runoob.com/python3/python3-file-next.html)返回文件下一行。 |
| 6 | [file.read([size])](http://www.runoob.com/python3/python3-file-read.html)从文件读取指定的字节数，如果未给定或为负则读取所有。 |
| 7 | [file.readline([size])](http://www.runoob.com/python3/python3-file-readline.html)读取整行，包括 "\n" 字符。 |
| 8 | [file.readlines([sizeint])](http://www.runoob.com/python3/python3-file-readlines.html)读取所有行并返回列表，若给定sizeint>0，返回总和大约为sizeint字节的行, 实际读取值可能比 sizeint 较大, 因为需要填充缓冲区。 |
| 9 | [file.seek(offset[, whence])](http://www.runoob.com/python3/python3-file-seek.html)设置文件当前位置 |
| 10 | [file.tell()](http://www.runoob.com/python3/python3-file-tell.html)返回文件当前位置。 |
| 11 | [file.truncate([size])](http://www.runoob.com/python3/python3-file-truncate.html)从文件的首行首字符开始截断，截断文件为 size 个字符，无 size 表示从当前位置截断；截断之后后面的所有字符被删除，其中 Widnows 系统下的换行代表2个字符大小。 |
| 12 | [file.write(str)](http://www.runoob.com/python3/python3-file-write.html)将字符串写入文件，返回的是写入的字符长度。 |
| 13 | [file.writelines(sequence)](http://www.runoob.com/python3/python3-file-writelines.html)向文件写入一个序列字符串列表，如果需要换行则要自己加入每行的换行符。 |

## pickle模块与数据序列化

python的pickle模块实现了基本的数据序列和反序列化。

序列化：

pickle.dump(obj, file, [,protocol])

“““

Attributes:

obj -- 想要序列化的对象

file -- 用以储存序列化对象的文件对象

protocol -- 可选的操作协议

”””

反序列化：

x = pickle.load(file)

#从file文件对象中重构出数据对象赋给x

# Python3 错误和异常

Python有两种错误很容易辨认：语法错误和异常。

语法错误

Python 的语法错误或者称之为解析错，是编译运行前检查到的错误。语法分析器指出了出错的一行，并且在最先找到的错误的位置标记了一个小小的箭头。

异常

运行期检测到的错误被称为异常。

异常处理

Python中的异常处理采用try…except块结构：

try:

…

except 错误类型名:

…

except 错误类型名 as 变量名:

#不仅拦截错误，还将错误对象以自定义的名称捕捉，可用于调用其成员

…

except: #类似default在Java语言中异常处理的作用，拦截所有“漏网之鱼”

…

* 首先，执行try子句（在关键字try和关键字except之间的语句）
* 如果没有异常发生，忽略except子句，try子句执行后结束。
* 如果在执行try子句的过程中发生了异常，那么try子句余下的部分将被忽略。如果异常的类型和 except 之后的名称相符，那么对应的except子句将被执行。最后执行 try 语句之后的代码。
* 如果一个异常没有与任何的except匹配，那么这个异常将会传递给上层的try中。
* 一个 try 语句可能包含多个except子句，分别来处理不同的特定的异常。最多只有一个分支会被执行。
* 一个except子句可以同时处理多个异常，这些异常将被放在一个括号里成为一个元组，例如: except (RuntimeError, TypeError, NameError):
* 最后一个except子句可以忽略异常的名称，它将被当作通配符使用。你可以使用这种方法打印一个错误信息，然后再次把异常抛出。
* try except 语句还有一个可选的else子句，如果使用这个子句，那么必须放在所有的except子句之后。这个子句将在try子句没有发生任何异常的时候执行。
* try 语句还有另外一个可选的子句，它定义了无论在任何情况下都会执行的清理行为，即finally字句，和Java类似。

抛出异常

Python 使用 raise 语句抛出一个指定的异常，类似Java中的throw，有两种用法：

①raise 错误类型(可选构函参数)

唯一的一个参数指定了要被抛出的异常。它必须是一个异常的实例或者是异常的类（也就是 Exception 的子类）。

②raise

单独的raise关键字用于except块中将当前处理的错误继续向上抛出。

用户自定义异常

可以通过创建一个新的异常类来拥有自己的异常。异常类继承自 Exception 类，可以直接继承，或者间接继承。

当创建一个模块有可能抛出多种不同的异常时，一种通常的做法是为这个包建立一个基础异常类，然后基于这个基础类为不同的错误情况创建不同的子类。

预定义的清理行为

关键词 with 语句就可以保证诸如文件之类的对象在使用完之后一定会正确的执行他的清理方法:

with open("myfile.txt") as f:

for line in f:

print(line, end="")

以上这段代码执行完毕后，就算在处理过程中出问题了，文件 f 总是会关闭。但不用with关键词，需手动关闭文件。

# Python3 面向对象

Python面向对象特点简介

①类的继承机制允许多个基类。

②派生类可以覆盖基类中的任何方法。

③方法中可以调用基类中的同名方法。

④对象可以包含任意数量和类型的数据。

⑤因python中变量必须赋值且多数无法更改，数据成员包括类变量和实例变量。类变量在整个实例化的对象中是公用的。类变量定义在类中且在函数体之外。类变量通常不作为实例变量使用。实例变量是定义在方法中的变量，只作用于当前实例的类。

⑥类的构造方法调用与一般编程语言类似，但定义时不以类名为构造方法名。

类定义

语法格式如下：

class ClassName:

<statement-1>

. . .

<statement-N>

举例：

class MyClass:

"""一个简单的类实例"""

i = 12345 #类变量的定义

def f(self):

return 'hello world'

类的成员

①公共类变量的定义与类变量的使用和其他编程语言的数据成员或属性相同，只是已赋值且不可更改。  
私有类变量定义时，必须以双下划线\_\_为起始命名，如\_\_weight。私有属性在类外部无法直接进行访问。在类内部的方法中使用时self.\_\_private\_attrs。

②实例变量

③公共方法成员的调用与其他编程语言相同。但定义和原型上与普通的函数有一个特别的区别——它们必须有一个额外的第一个参数名称, 按照惯例它的名称是 self。self 代表的是类的实例，代表当前对象的地址，用法类似其他语言中的this。然而self本身不是预定义的关键字，可以使用任意字符，只是习惯上用self。换句话说，成员函数必须提供一个额外的首参数发挥this的效果。调用时，忽略self参数为后续参数提供实参即可。  
私有方法以两个下划线开头，只能在类的内部调用 ，不能在类地外部调用。使用时也须以self.\_\_private\_methods调用。

类实例化

举例：

# 实例化类

x = MyClass() #直接调用构造函数，将对象赋给一个变量即可。调用时构造函数与类同名

构造方法

Python中的构造方法在定义时不以类名为方法名：

举例：

def \_\_init\_\_(self):

self.data = []

#构造方法可以有参数或没有，本例为无参数。且定义时必须名为\_\_init\_\_

类的继承

派生类的定义如下所示:

class DerivedClassName(BaseClassName1, BaseClassName2 …):

<statement-1>

. . .

<statement-N>

需要注意圆括号中基类的顺序，若是基类中有相同的方法名，而在子类使用时未指定，python从左至右搜索 即方法在子类中未找到时，从左到右查找基类中是否包含方法。

可以在子类的构函中调用基类的构函，方式为：基类名.\_\_init\_\_(参数)。或利用super函数。

直接在子类中定义同名函数即可自动覆写基类函数。

super()函数

super() 函数是用于调用父类(超类)的一个方法。有两个形式：

super(class, object) 返回class类object对象的基类的对象

super() 相当于super(class, self)，返回所在类的基类对象

运算符重载

Python同样支持运算符重载，这通过对类的专有方法进行重载来实现。专有方法具体见下节。

举例：

class Vector:

def \_\_init\_\_(self, a, b):

self.a = a

self.b = b

def \_\_str\_\_(self):

return 'Vector (%d, %d)' % (self.a, self.b)

def \_\_add\_\_(self,other):

return Vector(self.a + other.a, self.b + other.b) #重写\_\_add\_\_方法实现对+运算符重载

与类有关的预定义

①预定义变量

\_\_class\_\_：类变量，包含对象的类名（完整包路径下）

②预定义方法

* \_\_init\_\_ : 构造函数，在生成对象时调用
* \_\_del\_\_ : 析构函数，释放对象时使用
* \_\_repr\_\_ : 打印，转换
* \_\_setitem\_\_ : 按照索引赋值
* \_\_getitem\_\_: 按照索引获取值
* \_\_len\_\_: 获得长度
* \_\_cmp\_\_: 比较运算
* \_\_call\_\_: 函数调用
* \_\_add\_\_: 加运算
* \_\_sub\_\_: 减运算
* \_\_mul\_\_: 乘运算
* \_\_div\_\_: 除运算
* \_\_mod\_\_: 求余运算
* \_\_pow\_\_: 乘方

# Python3 字符串匹配与正则表达式

Python 自1.5版本起增加了re 模块，它提供 Perl 风格的正则表达式模式。

re 模块使 Python 语言拥有全部的正则表达式功能。re模块为高级字符串处理提供了正则表达式工具。对于复杂的匹配和处理，正则表达式提供了简洁、优化的解决方案。如果只需要简单的功能，应该首先考虑字符串方法，因为它们非常简单，易于阅读和调试。

正则表达式

正则表达式是一个特殊的字符序列，它能帮助你方便的检查一个字符串是否与某种模式匹配。模式字符串使用特殊的语法来表示一个正则表达式。

·字母和数字表示他们自身。一个正则表达式模式中的字母和数字匹配同样的字符串。

·多数字母和数字前加一个反斜杠时会拥有不同的含义。

·标点符号只有被转义时才匹配自身，否则它们表示特殊的含义。

·反斜杠本身需要使用反斜杠转义。

·由于正则表达式通常都包含反斜杠，所以你最好使用r或R来表示原始字符串。

正则表达式常用符号含义：（r为原始字符串标志，e代表任意正则表达式形式，所有例子省略字符串引号）

|  |  |
| --- | --- |
| **符号举例** | **描述** |
| ^ | 从字符串的开头匹配 |
| $ | 匹配至字符串的末尾。 |
| . | 匹配任意字符，除了换行符，当re.DOTALL标记被指定时，则可以匹配包括换行符的任意字符。 |
| [...] | 用来表示一组字符,单独列出：[amk] 匹配 'a'，'m'或'k' |
| [^...] | 不在[]中的字符：[^abc] 匹配除了a,b,c之外的字符。 |
| re\* | 匹配0个或多个的表达式。 |
| re+ | 匹配1个或多个的表达式。 |
| re? | 匹配0个或1个由前面的正则表达式定义的片段，非贪婪方式 |
| re{ n} | 匹配n个前面表达式。例如，"o{2}"不能匹配"Bob"中的"o"，但是能匹配"food"中的两个o。 |
| re{ n,} | 精确匹配n个前面表达式。例如，"o{2,}"不能匹配"Bob"中的"o"，但能匹配"foooood"中的所有o。"o{1,}"等价于"o+"。"o{0,}"则等价于"o\*"。 |
| re{ n, m} | 匹配 n 到 m 次由前面的正则表达式定义的片段，贪婪方式 |
| a| b | 匹配a或b |
| (re) | G匹配括号内的表达式，也表示一个组 |
| (?imx) | 正则表达式包含三种可选标志：i, m, 或 x 。只影响括号中的区域。 |
| (?-imx) | 正则表达式关闭 i, m, 或 x 可选标志。只影响括号中的区域。 |
| (?: re) | 类似 (...), 但是不表示一个组 |
| (?imx: re) | 在括号中使用i, m, 或 x 可选标志 |
| (?-imx: re) | 在括号中不使用i, m, 或 x 可选标志 |
| (?#...) | 注释. |
| (?= re) | 前向肯定界定符。如果所含正则表达式，以 ... 表示，在当前位置成功匹配时成功，否则失败。但一旦所含表达式已经尝试，匹配引擎根本没有提高；模式的剩余部分还要尝试界定符的右边。 |
| (?! re) | 前向否定界定符。与肯定界定符相反；当所含表达式不能在字符串当前位置匹配时成功。 |
| (?> re) | 匹配的独立模式，省去回溯。 |
| \w | 匹配数字字母下划线 |
| \W | 匹配非数字字母下划线 |
| \s | 匹配任意空白字符，等价于 [\t\n\r\f]。 |
| \S | 匹配任意非空字符 |
| \d | 匹配任意数字，等价于 [0-9]。 |
| \D | 匹配任意非数字 |
| \A | 匹配字符串开始 |
| \Z | 匹配字符串结束，如果是存在换行，只匹配到换行前的结束字符串。 |
| \z | 匹配字符串结束 |
| \G | 匹配最后匹配完成的位置。 |
| \b | 匹配一个单词边界，也就是指单词和空格间的位置。例如， 'er\b' 可以匹配"never" 中的 'er'，但不能匹配 "verb" 中的 'er'。 |
| \B | 匹配非单词边界。'er\B' 能匹配 "verb" 中的 'er'，但不能匹配 "never" 中的 'er'。 |
| \n, \t, 等。 | 匹配一个换行符。匹配一个制表符, 等 |
| \1...\9 | 匹配第n个分组的内容。 |
| \10 | 匹配第n个分组的内容，如果它经匹配。否则指的是八进制字符码的表达式。 |

正则表达式修饰符 - 可选标志flags

正则表达式可以包含一些可选标志修饰符来控制匹配的模式Mode。修饰符被指定为一个可选的标志。多个标志可以通过按位 OR(|) 它们来指定。这并非是指定其中任一，因为标志位和标志一一对应，通过此法可将对应标志位设置成所期望的结果，最终通过一个单独的标志二进制序列来表示全部设置。

|  |  |
| --- | --- |
| 修饰符 | 描述 |
| re.I | 使匹配对大小写不敏感 |
| re.L | 做本地化识别（locale-aware）匹配，表示特殊字符集 \w, \W, \b, \B, \s, \S 依赖于当前环境。 |
| re.M | 多行匹配，影响 ^ 和 $ |
| re.S | 使 . 匹配包括换行符在内的所有字符 |
| re.U | 表示特殊字符集 \w, \W, \b, \B, \d, \D, \s, \S 依赖于 Unicode 字符属性数据库 |
| re.X | 为了增加可读性，忽略空格和' # '后面的注释 |

re模块常用函数

①re.match

re.match 尝试从字符串的起始位置匹配一个模式，匹配成功re.match方法返回一个匹配的对象re.MatchObject，如果不是起始位置匹配成功的话，match()就返回none。

函数语法：

re.match(pattern, string, flags=0)

函数参数说明：

|  |  |
| --- | --- |
| 参数 | 描述 |
| pattern | 匹配的正则表达式 |
| string | 要匹配的字符串。 |
| flags | 标志位，用于控制正则表达式的匹配方式，如：是否区分大小写，多行匹配等等。参见：正则表达式修饰符 - 可选标志 |

②re.search

re.search 扫描整个字符串并返回第一个成功的匹配。匹配成功re.search方法返回一个匹配的对象re.MatchObject，否则返回None。

函数语法：

re.search(pattern, string, flags=0)

③re.sub

用于替换字符串中的匹配项。

语法：

re.sub(pattern, repl, string, count=0)

|  |  |
| --- | --- |
| 参数 | 描述 |
| repl | 替换的字符串，也可为一个函数。 |
| count | 模式匹配后替换的最大次数，默认 0 表示替换所有的匹配。 |

④re.compile

compile 函数用于编译正则表达式，生成一个正则表达式（re.RegexObject）对象，供 match() 和 search() 这两个函数使用。注意，这两个函数与re.match和re.search的区别，它们是RegexObject的成员。

语法格式为：

re.compile(pattern[, flags])

re.RegexObject还有一个成员函数：

findall(string[, pos[, endpos]])

在字符串中找到正则表达式所匹配的所有子串，并返回一个列表，如果没有找到匹配的，则返回空列表。

注意： match 和 search 是匹配一次 findall 匹配所有。

|  |  |
| --- | --- |
| 参数 | 描述 |
| pos | 可选参数，指定字符串的起始位置，默认为 0。 |
| endpos | 可选参数，指定字符串的结束位置，默认为字符串的长度。 |

⑤re.finditer

和 findall 类似，在字符串中找到正则表达式所匹配的所有子串，并把它们作为一个迭代器返回。语法：

re.finditer(pattern, string, flags=0)

⑥re.split

split 方法按照能够匹配的子串将字符串分割后返回列表，它的使用形式如下：

re.split(pattern, string[, maxsplit=0, flags=0])

|  |  |
| --- | --- |
| 参数 | 描述 |
| maxsplit | 分隔次数，maxsplit=1 分隔一次，默认为 0，不限制次数。 |

re.MatchObject的成员函数

①group([group1, …])

方法用于获得一个或多个分组匹配的字符串，当要获得整个匹配的子串时，可直接使用 group() 或 group(0)；

|  |  |
| --- | --- |
| 参数 | 描述 |
| group | 组号，一个从1开始的数字 |

②start([group])

方法用于获取分组匹配的子串在整个字符串中的起始位置（子串第一个字符的索引），参数默认值为 0；

③end([group])

方法用于获取分组匹配的子串在整个字符串中的结束位置（子串最后一个字符的索引+1），参数默认值为 0；

④span([group])

方法返回 (start(group), end(group))。

# Python3标准库概览

## os模块

操作系统接口

os模块

建议使用 "import os" 风格而非 "from os import \*"。这样可以保证随操作系统不同而有所变化的 os.open() 不会覆盖内置函数 open()。

shutil模块

针对日常的文件和目录管理任务，shutil 模块提供了一个易于使用的高级接口。

## glob模块

文件通配符

glob模块提供了一个函数用于从目录通配符搜索中生成文件列表。

## sys模块

命令行参数

通用工具脚本经常调用命令行参数。这些命令行参数以链表形式存储于 sys 模块的 argv 变量。

错误输出重定向

sys 还有 stdin，stdout 和 stderr 属性，即使在 stdout 被重定向时，后者也可以用于显示警告和错误信息。

程序终止

大多脚本的定向终止都使用 "sys.exit()"。

## math模块

math模块为浮点运算提供了对底层C函数库的访问。

## random模块

random提供了生成随机数的工具。