**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，单字符也在Python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：

#!/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字符串更新**

你可以对已存在的字符串进行修改，并赋值给另一个变量，如下实例：

#!/usr/bin/python # -\*- coding: UTF-8 -\*- var1 = 'Hello World!' print "更新字符串 :- ", var1[:6] + 'Runoob!'

以上实例执行结果

更新字符串 :- 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，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 | 请看下一章节 |

实例如下：

#!/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()](http://www.runoob.com/python/att-string-format.html)，它增强了字符串格式化的功能。

**Python三引号（triple quotes）**

python中三引号可以将复杂的字符串进行复制:

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，这时用字符串组合，特殊字符串转义将会非常的繁琐。

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()](http://www.runoob.com/python/att-string-capitalize.html) | 把字符串的第一个字符大写 |
| [string.center(width)](http://www.runoob.com/python/att-string-center.html) | 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))**](http://www.runoob.com/python/att-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 |
| [string.decode(encoding='UTF-8', errors='strict')](http://www.runoob.com/python/att-string-decode.html) | 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')](http://www.runoob.com/python/att-string-encode.html) | 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))**](http://www.runoob.com/python/att-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. |
| [string.expandtabs(tabsize=8)](http://www.runoob.com/python/att-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。 |
| [**string.find(str, beg=0, end=len(string))**](http://www.runoob.com/python/att-string-find.html) | 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()**](http://www.runoob.com/python/att-string-format.html) | 格式化字符串 |
| [**string.index(str, beg=0, end=len(string))**](http://www.runoob.com/python/att-string-index.html) | 跟find()方法一样，只不过如果str不在 string中会报一个异常. |
| [string.isalnum()](http://www.runoob.com/python/att-string-isalnum.html) | 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()](http://www.runoob.com/python/att-string-isalpha.html) | 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()](http://www.runoob.com/python/att-string-isdecimal.html) | 如果 string 只包含十进制数字则返回 True 否则返回 False. |
| [string.isdigit()](http://www.runoob.com/python/att-string-isdigit.html) | 如果 string 只包含数字则返回 True 否则返回 False. |
| [string.islower()](http://www.runoob.com/python/att-string-islower.html) | 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()](http://www.runoob.com/python/att-string-isnumeric.html) | 如果 string 中只包含数字字符，则返回 True，否则返回 False |
| [string.isspace()](http://www.runoob.com/python/att-string-isspace.html) | 如果 string 中只包含空格，则返回 True，否则返回 False. |
| [string.istitle()](http://www.runoob.com/python/att-string-istitle.html) | 如果 string 是标题化的(见 title())则返回 True，否则返回 False |
| [string.isupper()](http://www.runoob.com/python/att-string-isupper.html) | 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)**](http://www.runoob.com/python/att-string-join.html) | 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)](http://www.runoob.com/python/att-string-ljust.html) | 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()](http://www.runoob.com/python/att-string-lower.html) | 转换 string 中所有大写字符为小写. |
| [string.lstrip()](http://www.runoob.com/python/att-string-lstrip.html) | 截掉 string 左边的空格 |
| [string.maketrans(intab, outtab])](http://www.runoob.com/python/att-string-maketrans.html) | 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)](http://www.runoob.com/python/att-string-max.html) | 返回字符串 str 中最大的字母。 |
| [min(str)](http://www.runoob.com/python/att-string-min.html) | 返回字符串 str 中最小的字母。 |
| [**string.partition(str)**](http://www.runoob.com/python/att-string-partition.html) | 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))**](http://www.runoob.com/python/att-string-replace.html) | 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) )](http://www.runoob.com/python/att-string-rfind.html) | 类似于 find()函数，不过是从右边开始查找. |
| [string.rindex( str, beg=0,end=len(string))](http://www.runoob.com/python/att-string-rindex.html) | 类似于 index()，不过是从右边开始. |
| [string.rjust(width)](http://www.runoob.com/python/att-string-rjust.html) | 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()](http://www.runoob.com/python/att-string-rstrip.html) | 删除 string 字符串末尾的空格. |
| [**string.split(str="", num=string.count(str))**](http://www.runoob.com/python/att-string-split.html) | 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])](http://www.runoob.com/python/att-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。 |
| [string.startswith(obj, beg=0,end=len(string))](http://www.runoob.com/python/att-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. |
| [**string.strip([obj])**](http://www.runoob.com/python/att-string-strip.html) | 在 string 上执行 lstrip()和 rstrip() |
| [string.swapcase()](http://www.runoob.com/python/att-string-swapcase.html) | 翻转 string 中的大小写 |
| [string.title()](http://www.runoob.com/python/att-string-title.html) | 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="")**](http://www.runoob.com/python/att-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 del 参数中 |
| [string.upper()](http://www.runoob.com/python/att-string-upper.html) | 转换 string 中的小写字母为大写 |
| [string.zfill(width)](http://www.runoob.com/python/att-string-zfill.html) | 返回长度为 width 的字符串，原字符串 string 右对齐，前面填充0 |
| [string.isdecimal()](http://www.runoob.com/python/att-string-isdecimal.html) | isdecimal()方法检查字符串是否只包含十进制字符。这种方法只存在于unicode对象。 |

**去掉字符串的\n strip**

  一般，我们做一些简单的状态输出都会用print，但是这是最简单的情况下使用的工具。当我们的程序比较复杂的时候，我们会使用日志文件，特别是程序运行的时间特别久，中间可能存在一些问题，需要后面来看的时候。

        所以，[**Python**](http://lib.csdn.net/base/python)自带了一个很有用的库，logger，也就是日志记录。

        使用起来还是很方便的。

**[python]** [view plain](http://blog.csdn.net/qtlyx/article/details/76037738) [copy](http://blog.csdn.net/qtlyx/article/details/76037738)

1. #!/usr/bin/env python
2. # -\*- coding: utf-8 -\*-
3. **import logging**
4. # create a log file
5. logger = logging.getLogger('atp\_log')
6. logger.setLevel(logging.DEBUG)
7. # create a handler, write the log info into it
8. fh = logging.FileHandler('atp.log')
9. fh.setLevel(logging.DEBUG)
10. # create another handler output the log though console   #console 控制台 操纵台。
11. ch = logging.StreamHandler()
12. ch.setLevel(logging.DEBUG)
13. # 定义handler的输出格式
14. formatter = logging.Formatter('%(asctime)s - %(levelname)s - %(message)s')
15. fh.setFormatter(formatter)
16. ch.setFormatter(formatter)
17. # 给logger添加handler
18. logger.addHandler(fh)
19. logger.addHandler(ch)
20. # 记录一条日志
21. logger.info('foorbar')
22. logger.error('foorbar')

        之后，我们对logger输入info warning或者error都可以，而且会被记录在日志文件里面。当然，上面的代码中，我们在设置

**[python]** [view plain](http://blog.csdn.net/qtlyx/article/details/76037738) [copy](http://blog.csdn.net/qtlyx/article/details/76037738)

1. <span style="font-size:18px;">fh = logging.FileHandler('atp.log')</span>

        这个文件的logger也创建了一个从console的日志显示的地方。

**[python]** [view plain](http://blog.csdn.net/qtlyx/article/details/76037738) [copy](http://blog.csdn.net/qtlyx/article/details/76037738)

1. <pre code\_snippet\_id="2497500" snippet\_file\_name="blog\_20170724\_3\_8385705" name="code" **class**="python"><span style="font-size:18px;">ch = logging.StreamHandler()</span></pre><p></p>
2. <pre></pre>
3. <p></p>
4. <p><span style="font-size:18px">        之后，我们就可以看到在文件和console中都会有相应的信息出现。</span></p>
5. <p><img src="http://img.blog.csdn.net/20170724203828576?watermark/2/text/aHR0cDovL2Jsb2cuY3Nkbi5uZXQvcXRseXg=/font/5a6L5L2T/fontsize/400/fill/I0JBQkFCMA==/dissolve/70/gravity/Center" alt=""><br>
6. </p>