

- Lua 教程
- Lua 教程
- Lua 环境安装
- Lua 基本语法
- Lua 数据类型
- Lua 变量
- Lua 循环
- Lua 流程控制
- Lua 函数
- Lua 运算符
- Lua 字符串
- Lua 数组
- Lua 迭代器
- Lua table(表)
- Lua 模块与包
- Lua 元表(Metatable)
- Lua 协程程序(coroutine)
- Lua 文件 IO
- Lua 错误处理
- Lua 调试(Debug)
- Lua 面向对象
- Lua 数据库访问
- Lua5.3 参考手册

Lua 字符串

字符串或串(String)是由数字、字母、下划线组成的一串字符。

Lua 语言中字符串可以使用以下三种方式表示：

- 单引号间的一串字符。
- 双引号间的一串字符。
- [和]间的一串字符。

以上三种方式的字符串实例如下：

实例

```
string1 = "Lua"
print("字符串 1 是" .. string1)
string2 = 'runoob.com'
print("字符串 2 是",string2)

string3 = [[Lua 教程]]
print("字符串 3 是",string3)
```

以上代码执行输出结果为：

```
"字符串 1 是"      Lua
字符串 2 是      runoob.com
字符串 3 是      "Lua 教程"
```

转义字符用于表示不能直接显示的字符，比如退格、回车键、等。如在字符串转义双引号可以使用“\”。

所有的转义字符和所对应的意义：

转义字符	意义	ASCII码值（十进制）
\a	响铃(BEL)	007
\b	退格(BS)，将当前位置移到前一列	008
\f	换行(F)，将当前位置移到下一页开头	012
\n	换行(LF)，将当前位置移到下一行开头	010
\r	回车(CR)，将当前位置移到本行开头	013
\t	水平制表(HT)（跳到下一个TAB位置）	009
\v	垂直制表(VT)	011
\\	代表一个反斜线字符\"	092
\'	代表一个单引号（撇号）字符	039
\"	代表一个双引号字符	034
\0	空字符串(NULL)	000
\ddd	1到3位八进制数所代表的任意字符	三位八进制
\hhh	1到3位十六进制数所代表的任意字符	二位十六进制

字符串操作

Lua 提供了很多的方法来支持字符串的操作：

序号	方法 & 用途
1	string.upper(argument) 字符串全部转为大写字母。
2	string.lower(argument) 字符串全部转为小写字母。
3	string.gsub(mainString,findString,replaceString,num) 在字符串中替换,mainString为要替换的字符串， findString 为被替换的字符， replaceString 要替换的字符， num 替换次数（可以忽略，则全部替换），如： <pre>> string.gsub("aaaa","a","z",3); zzzz 3</pre>
4	string.find(str, substr, [init, [end]]) 在一个指定的目标字符串中搜索指定的内容(第三个参数为索引),返回其具体位置。不存在则返回 nil。 <pre>> string.find("Hello Lua user", "Lua", 1) 7 9</pre>
5	string.reverse(arg) 字符串反转 <pre>> string.reverse("Lua") aul</pre>
6	string.format(...) 返回一个类似printf的格式化字符串 <pre>> string.format("the value is:%d",4) the value is:4</pre>
7	string.char(arg) 和 string.byte(arg,[n]) char 将整型数字转成字符并连接， byte 转换字符为整数值(可以指定某个字符，默认第一个字符)。 <pre>> string.char(97,98,99,100) abcd > string.byte("ABCD",4) 68 > string.byte("ABCD") 65 ></pre>
8	string.len(arg) 计算字符串长度。 <pre>string.len("abc") 3</pre>
9	string.rep(string, n) 返回字符串string的n个拷贝 <pre>> string.rep("abcd",2) abcdabcd</pre>
10	.. 链接两个字符串 <pre>> print("www.runoob..."..".com") www.runoob.com</pre>
11	string.gsub(str, pattern) 同一个迭代函数，每一次调用这个函数，返回一个在字符串 str 找到的下一个符合 pattern 描述的子串。如果参数 pattern 描述的字符串中没有找到，迭代函数返回nil。 <pre>> for word in string.gmatch("Hello Lua user", "%a+") do print(word) end Hello Lua user</pre>
12	string.match(str, pattern, init) string.match()只寻找字符串str中的第一个匹配 参数init可选，指定搜索过程的起点，默认为1。 在成功匹配时，函数将返回对模式表达式中的所有捕获结果。如果没有设置捕获标记，则返回整个匹配对字符串。当没有成功的匹配时，返回nil。 <pre>> = string.match("I have 2 questions for you.", "%d+ %a+") 2 questions > = string.format("%d, %d", string.match("I have 2 questions for you.", "(%d+) (%a+)")) 2, "questions"</pre>

字符串大小写转换

以下实例演示了如何对字符串大小写进行转换：

实例

```
string1 = "Lua";
print(string.upper(string1))
print(string.lower(string1))
```

以上代码执行结果为：

```
LUA
lua
```

字符串查找与反转

以下实例演示了如何对字符串进行查找与反转操作：

实例

```
string = "Lua Tutorial"
-- 查找字符串
print(string.find(string,"Tutorial"))
reversedString = string.reverse(string)
print("新字符串为",reversedString)
```

以上代码执行结果为：

```
5      12
新字符串为      lairout uoL
```

字符串格式化

Lua 提供了 **string.format()** 函数来生成具有特定格式的字符串。函数的第一个参数是格式，之后是对应格式中每个代号的各称数据。

由于格式字符串的存在，使得产生的字符串可读性大大提高了。这个函数的格式很像 C 语言中的 printf()。

以下实例演示了如何对字符串进行格式化操作：

格式字符串可能包含以下的转义码。

- %c -接受一个数字，并把它转化为ASCII码表中对应的字符
- %d, %i -接受一个数字并將其转化为有符号的整数格式
- %o -接受一个数字并將其转化为八进制数格式
- %u -接受一个数字并將其转化为无符号整数格式
- %x -接受一个数字并將其转化为十六进制数格式。使用小写字母
- %X -接受一个数字并將其转化为十六进制数格式。使用大写字母
- %e -接受一个数字并將其转化为科学记数法格式。使用小写字母e
- %E -接受一个数字并將其转化为科学记数法格式。使用大写字母E
- %f -接受一个数字并將其转化为浮点数值
- %(%)G -接受一个数字并將其转化为%(%)E、对应G及%中较短的一种格式
- % -接受一个字符串并將其转化为安全被Lua解释器接受的格式
- %s -接受一个字符串并按照给定的参数格式化该字符串

为进一步细化格式，可以在%号后添加参数，参数所以下的规则来定义。

- (1) 符号：一个+号表示其后的数字转义符将正数显示正号。默认情况下只有负数显示符号。
- (2) 占位符：一个0，在后面指定了字符串宽度时占位用。不输入的话默认占位符是空格。
- (3) 对齐标识：在指定了字符串宽度时，默认为右对齐。增加号可以改为左对齐。
- (4) 宽度数值
- (5) 小数位数/小数位：在宽度数值后增加的小数部分n，若后接浮点数值转义符，如%6.3则设定该浮点数值的小数只保留n位。若后接n个字符串转义符，如%5.3s则设定该字符串显示n个位

实例

```
string1 = "Lua"
string2 = "Tutorial"
number1 = 10
number2 = 20
-- 基本字符串格式化
print(string.format("基本格式化 %s %s",string1,string2))
-- 日期格式化
date = 2; month = 1; year = 2014
print(string.format("日期格式化 %02d/%02d/%04d", date, month, year))
-- 字符串格式化
print(string.format("%.4f",1/3))
```

以上代码执行结果为：

```
基本格式化 Lua Tutorial
日期格式化 02/01/2014
0.3333
```

其他例子：

实例

```
string.format("%c", 83)      -- 输出S
string.format("%d", 17.0)    -- 输出17
string.format("%04d", 17)    -- 输出0017
string.format("%0", 17)      -- 输出121
string.format("%u", 3.14)     -- 输出3
string.format("%u", 13)       -- 输出13
string.format("%x", 13)       -- 输出D
string.format("%e", 1000)     -- 输出1.000000e+03
string.format("%e", 1000)     -- 输出1.000000e+03
string.format("%g", 14), 13)  -- 输出12.000
string.format("%s", "OneTwo") -- 输出One1
                                --      Two2
string.format("%s", "monkey") -- 输出monkey
string.format("%3s", "monkey") -- 输出   monkey
string.format("%5.3s", "monkey") -- 输出    mon
```

字符串与整数相互转换

以下实例演示了字符串与整数相互转换：

实例

```
-- 字符串转换
-- 转换第一个字节
print(string.byte("Lua"))
-- 转换第三个字节
print(string.byte("Lua",3))
-- 转换从第一个字节
print(string.byte("Lua",1))
-- 第二个字节
print(string.byte("Lua",2))
-- 转换从第二个字节
print(string.byte("Lua",-2))
-- 整数 ASCII 四转换成字符
print(string.char(97))
```

以上代码执行结果为：

```
76
97
97
117
117
a
```

其他常用函数

以下实例演示了其他字符串操作，如计算字符串长度、字符串连接、字符串复制等：

实例

```
string1 = "www."
string2 = "runoob"
string3 = ".com"
-- 使用 .. 进行字符串连接
print("连接字符串",string1..string2..string3)

-- 字符串长度
print("字符串长度",string.len(string2))

-- 字符串复制 2 次
repeatedString = string.rep(string2,2)
print(repeatedString)
```

以上代码执行结果为：

```
连接字符串      www.runoob.com
字符串长度      6
runoobrunoob
```

匹配模式

Lua 中的匹配模式直接用常规的字符串来描述。它用于模式匹配函数 **string.find**, **string.gmatch**, **string.gsub**, **string.match**。

你还可以在模式串中使用转义码。

字符串匹配可以匹配一个特定字符串集合内任何字符的模式。比如，字符串 %d 匹配任意数字。所以你可以使用模式 %s%d/%s%d/%s%d 搜索 dddmm/yyyy 格式的日期：

实例

```
s = "Deadline is 30/05/1999, firm"
date = "%d/%d/%d,%d%d"
print(string.sub(s, string.find(s, date)))  --> 30/05/1999
```

下面的例子列出了Lua支持的所有转义码：

单个字符转义 %s(%i?~*+?)：与该字符自身匹配：

- (点) 与任何字符匹配
- %a 与任何字母匹配
- %c 与任何控制符匹配(例如\n)
- %d 与任何数字匹配
- %l 与任何小写字母匹配
- %p 与任何标点(punctuation)匹配
- %s 与空白字符匹配
- %u 与任何大写字母匹配
- %w 与任何字母/数字匹配
- %x 与任何十六进制数匹配
- %z 与任何代表空的字符匹配
- %x%d 是非字母非数字字符串 与字符x匹配。 主要用来处理表达式中有功能的字符(%s(%i?~*+?)的匹配问题。例如%w与%w匹配
- [数字类] 与任何[]中包含的字符类匹配。例如[%w_与任何字母/数字、下划线或下划线符号匹配
- [数字类] 与任何不包含在[]中的字符类匹配。例如[%s]与任何非空白字符匹配

当上述的字符类类用大写字母时，表示与非此字符类的任何字符匹配。例如 %S表示与任何非空白字符匹配，例如 %A非字母的字符。

```
> print(string.gsub("hello, up-down!", "%A", "."))
      hello,.up,.down.      4
```

数字4不是字符串结果的一部分，它是gsub返回的第二个结果，代表匹配替换的次数。

在模式匹配中有一些特殊字符，他们有特殊的意义，Lua中的特殊字符如下：

()	%	+	*	?	[^	\$
-----	---	---	---	---	---	---	----

% 用作特殊字符的转义字符，因此 % 匹配点：% % 匹配字符 %。转义字符 % 不仅可以用来转义特殊字符，还可以用于所有的非字母的字符。

模式条目可以是：

- 单个字符类匹配类到任意单个字符；
- 单个字符类匹配一个 'v'，将匹配零个或多个该类的字符。这个条目总是匹配尽可能长的串；
- 单个字符类匹配一个 'v'，将匹配一或更多个该类的字符。这个条目总是匹配尽可能长的串；
- 单个字符类匹配一个 '^'，将匹配零或更多个该类的字符，只要有可能，它会匹配一个；
- %n，这里的 n 可以从 1 到 9；这个条目匹配一个等于 n 号捕获物（后面有描述）的子串。
- %bxy，这里的 x 和 y 是两个相邻的字符，这些子模式被称为 捕获物。当匹配成功时，由 捕获物 匹配的字符串中的子串被保存起来用于未来的模式。捕获物以它们左括号的大写来编号。例如，对于模式 "(a+)(%w(%w+))"，字符串中匹配到 "a(1)(%w(%w+))" 的部分保存在第一个捕获物中（因此是编号 1）；由 " " 匹配的字符串是 2 号捕获物，匹配到 "%w+" 的部分是 3 号。
- 作为一个例外，空的捕获 () 将匹配到当前字符串的位置（它是一个数字）。例如，如果将模式 "(0 aa() ()" 作用到字符串 "1aaap" 上，将产生两个捕获物：3 和 5。

模式：

模式 用一个模式条目的序列。在模式最前面加上符号 ~ 将指定从字符串的开始处匹配。在模式最后面加上符号 & 将使匹配过程走到字符串的结尾。如果 ~ 和 & 出现在其它位置，它们均没有特殊含义，只表示自身。

捕获：

模式可以在内部用小括号括起一个子模式，这些子模式被称为 捕获物。当匹配成功时，由 捕获物 匹配的字符串中的子串被保存起来用于未来的模式。

模式 捕获物以它们左括号的大写来编号。例如，对于模式 "(a+)(%w(%w+))"，字符串中匹配到 "a(1)(%w(%w+))" 的部分保存在第一个捕获物中（因此是编号 1）；由 " " 匹配的字符串是 2 号捕获物，匹配到 "%w+" 的部分是 3 号。

作为一个例外，空的捕获 () 将匹配到当前字符串的位置（它是一个数字）。例如，如果将模式 "(0 aa() ()" 作用到字符串 "1aaap" 上，将产生两个捕获物：3 和 5。

◀ Lua 运算符

Lua 数组 ▶

+

4 篇笔记

✎ 写笔记

Web 前端免费公开课
授课模式：在线直播+课后视频，从零基础到高级HTML5 开发工程师

打开

广告 X

在线实例

- HTML 实例
- CSS 实例
- JavaScript 实例
- Ajax 实例
- jQuery 实例
- XML 实例
- Java 实例

字符串A工具

- HTML 字符串检测
- HTML ASCII 转义码
- HTML ISO-8859-1
- HTML 实体符号
- HTML 颜色码
- JSON 格式化工具

最新更新

- PostgreSQL 教程...
- PostgreSQL 进阶...
- PostgreSQL 创建...
- PostgreSQL 数据...
- C 语言函数与字...
- Vue.js AjaxAx...
- PostgreSQL 函数...

站点信息

- 意见反馈
- 合作联系
- 免责声明
- 关于我们
- 网站地图

关注微信

