

Lua 教程
Lua 环境安装
Lua 基本语法
Lua 数组类型
Lua 变量
Lua 循环
Lua 流程控制
Lua 函数
Lua 语句
<b>Lua 字符串</b>
Lua 数组
Lua 连接器
Lua table(表)
Lua 模块与包
Lua 元数据(Metatable)
Lua 协同程序(coroutine)
Lua 文件 I/O
Lua 错误处理
Lua 调试(Debug)
Lua 读取回收
Lua 面向对象
Lua 数据访问
Lua 5.3 参考手册

&lt; Lua 运算符

Lua 教程

**Lua 字符串**

字符串类型(String)是由数字、字母、下划线组成的一串字符。

Lua语言字符串可以使用以下三种方式来表示：

- ① 单引号包围的一串字符。
- ② 双引号包围的一串字符。
- ③ 用[]包围的一串字符。

以上三种方式的字符串示例如下：

**实例**

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

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"
```

转义字符串用于表示不能直接显示的字符，比如后退键、回车键，等，如在字符串转换双引号可以使用 `\"`。

所有转义字符串和它的意义：

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	换页(FF)，将当前位置移到下页开头	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
\\\	代表一个反斜杠字符`\'	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(NULL)	000
\ddd	1到9位八进制数所代表的任意字符	三位八进制
\hhh	1到2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：

**报错** 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 为被替换的字符串, replaceString 要替换的字符串, num 替换次数 (可以超过1,则全部替换), 如:
```

```
> string.gsub("aaaa","a","z",3);
zzza
```

```
4 string.find (str, substr, [init, [end]])
在一个指定的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,
```

```
> string.find("Hello Lua user", "Lua", 1)
7 9
```

```
5 string.reverse(arg)
字符串反转
```

```
> string.reverse("Lua")
aul
```

```
6 string.format(...)
返回一个类似print的格式化字符串
```

```
> string.format("the value is:%d",4)
the value is:4
```

```
7 string.char(arg) 和 string.byte(arg,[int])
char 将整型数字或字符串连接, byte 将整字符为整数值(可以指定某个字符, 默认第一个字符)。
```

```
> string.char(97,98,99,100)
abcde
> string.byte("ABCD",4)
65
> string.byte("ABCD")
65
>
```

```
8 string.len(arg)
计算字符串长度。
```

```
> string.len("abc")
3
```

```
9 string.rep(string,n)
返回字符串string的n个拷贝
```

```
> string.rep("abcd",2)
abcdabcd
```

```
10 ..
链接两个字符串
```

```
> print("www.runoob..com")
www.runoob.com
```

```
11 string.gmatch(str, pattern)
返回一个迭代器函数, 每次调用这个函数, 返回一个在字符串 str 找到的一个符合 pattern 描述的子串. 如果参数 pattern 描述的字符串没有找到, 迭代器函数返回nil.
```

```
> for word in string.gmatch("Hello lua user", "%a+") do print(word) end
Hello
Lua
user
```

```
12 string.match(str, pattern, init)
string.match()只寻找字符串中的第一个匹配, 参数init可选, 指定搜寻过程的起始点, 默认为1.
```

```
在成功匹配时, 尚会将匹配表达式的所有捕获结果, 如果没有设置捕获标记, 则返回整个字符串. 当没有成功的配对时, 返回nil.
```

```
> = string.match("I have 2 questions for you.", "%d+ %a+")
2 questions
```

```
> = string.format("%d %a", string.match("I have 2 questions for you.", "(%d+) (%a+)"))
2, "questions"
```

```
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**

```
string = "Lua Tutorial"
-- 直接字符串
print(string:find("Lua",1))
reverseString = string.reverse(string)
print("字符串翻转为",reverseString)
```

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"
-- 直接字符串
print(string:find("Lua",1))
reverseString = string.reverse(string)
print("字符串翻转为",reverseString)
```

以上代码执行结果为：

```
lua
```

**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码的各种数据。由于格式字符串的存在, 得使得字符串的读取性大大提高了. 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：

格式字符串可能会包含以下的感叹符号:

- ① %d -一个表示十进制数的字符串, 正数是正数, 默认情况下只有负数显示符号。
- ② %d. n -一个表示十进制数将转换为带有n位的整数格式。
- ③ %. n -接受一个数字并将转换为八进制数格式。
- ④ %. n -接受一个数字并将转换为无符号整数格式。
- ⑤ %. n -接受一个数字并将转换为十六进制数格式, 使用小写字母。
- ⑥ %. n -接受一个数字并将转换为十六进制数格式, 使用大写字母。
- ⑦ %. nE -接受一个数字并将其转换为科学记数法格式, 使用小写字母。
- ⑧ %. nG -接受一个数字并将其转换为科学记数法格式, 对应%. G 和%. nG 中较短的一种格式。
- ⑨ %. nQ -接受一个字符串并将其转化为可安全被Lua理解读入的格式。
- ⑩ %. nA -接受一个字符串并将其转化为字符串格式化读入的字符串。

为进一步简化格式化, 可以在%号后面添加参数, 参数将向以下的顺序读入:

- (1) 符号 -一个表示其后的数字符号, 正数是正数, 默认情况下只有负数显示符号。
- (2) 百分符 -一个在后面对指定了字符串宽度时的位置用一个编译时的人类字符串表示空格。
- (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) (可选)字符串截断 -在宽度值后增加的小数部分n, 后接点点数截断符, 如%. 3将设置该字符串的小数只保留n位, 若接a(字符串长度), 例如%. 3a则设置该字符串只显示前a位。

字符串的格式化方法可以在%号后添加参数, 参数将向以下的顺序读入:

- (1) 符号 -一个表示其后的数字符号, 正数是正数, 默认情况下只有负数显示符号。
- (2) 百分符 -一个在后面对指定了字符串宽度时的位置用一个编译时的人类字符串表示空格。
- (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) (可选)字符串截断 -在宽度值后增加的小数部分n, 后接点点数截断符, 如%. 3将设置该字符串的小数只保留n位, 若接a(字符串长度), 例如%. 3a则设置该字符串只显示前a位。

字符串的格式化方法可以在%号后添加参数, 参数将向以下的顺序读入:

- (1) 符号 -一个表示其后的数字符号, 正数是正数, 默认情况下只有负数显示符号。
- (2) 百分符 -一个在后面对指定了字符串宽度时的位置用一个编译时的人类字符串表示空格。
- (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) (可选)字符串截断 -在宽度值后增加的小数部分n, 后接点点数截断符, 如%. 3将设置该字符串的小数只保留n位, 若接a(字符串长度), 例如%. 3a则设置该字符串只显示前a位。

字符串的格式化方法可以在%号后添加参数, 参数将向以下的顺序读入:

- (1) 符号 -一个表示其后的数字符号, 正数是正数, 默认情况下只有负数显示符号。
- (2) 百分符 -一个在后面对指定了字符串宽度时的位置用一个编译时的人类字符串表示空格。
- (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) (可选)字符串截断 -在宽度值后增加的小数部分n, 后接点点数截断符, 如%. 3将设置该字符串的小数只保留n位, 若接a(字符串长度), 例如%. 3a则设置该字符串只显示前a位。

字符串的格式化方法可以在%号后添加参数, 参数将向以下的顺序读入:

- (1) 符号 -一个表示其后的数字符号, 正数是正数, 默认情况下只有负数显示符号。
- (2) 百分符 -一个在后面对指定了字符串宽度时的位置用一个编译时的人类字符串表示空格。
- (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) (可选)字符串截断 -在宽度值后增加的小数部分n, 后接点点数截断符, 如%. 3将设置该字符串的小数只保留n位, 若接a(字符串长度), 例如%. 3a则设置该字符串只显示前a位。

字符串的格式化方法可以在%号后添加参数, 参数将向以下的顺序读入:

- (1) 符号 -一个表示其后的数字符号, 正数是正数, 默认情况下只有负数显示符号。
- (2) 百分符 -一个在后面对指定了字符串宽度时的位置用一个编译时的人类字符串表示空格。
- (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) (可选)字符串截断 -在宽度值后增加的小数部分n, 后接点点数截断符, 如%. 3将设置该字符串的小数只保留n位, 若接a(字符串长度), 例如%. 3a则设置该字符串只显示前a位。

字符串的格式化方法可以在%号后添加参数, 参数将向以下的顺序读入:

- (1) 符号 -一个表示其后的数字符号, 正数是正数, 默认情况下只有负数显示符号。
- (2) 百分符 -一个在后面对指定了字符串宽度时的位置用一个编译时的人类字符串表示空格。
- (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) (可选)字符串截断 -在宽度值后增加的小数部分n, 后接点点数截断符, 如%. 3将设置该字符串的小数只保留n位, 若接a(字符串长度), 例如%. 3a则设置该字符串只显示前a位。

字符串的格式化方法可以在%号后添加参数, 参数将向以下的顺序读入:

- (1) 符号 -一个表示其后的数字符号, 正数是正数, 默认情况下只有负数显示符号。
- (2) 百分符 -一个在后面对指定了字符串宽度时的位置用一个编译时的人类字符串表示空格。
- (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) (可选)字符串截断 -在宽度值后增加的小数部分n, 后接点点数截断符, 如%. 3将设置该字符串的小数只保留n位, 若接a(字符串长度), 例如%. 3a则设置该字符串只显示前a位。

字符串的格式化方法可以在%号后添加参数, 参数将向以下的顺序读入:

- (1) 符号 -一个表示其后的数字符号, 正数是正数, 默认情况下只有负数显示符号。
- (2) 百分符 -一个在后面对指定了字符串宽度时的位置用一个编译时的人类字符串表示空格。
- (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) (可选)字符串截断 -在宽度值后增加的小数部分n, 后接点点数截断符, 如%. 3将设置该字符串的小数只保留n位, 若接a(字符串长度), 例如%. 3a则设置该字符串只显示前a位。

字符串的格式化方法可以在%号后添加参数, 参数将向以下的顺序读入:

- (1) 符号 -一个表示其后的数字符号, 正数是正数, 默认情况下只有负数显示符号。
- (2) 百分符 -一个在后面对指定了字符串宽度时的位置用一个编译时的人类字符串表示空格。
- (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) (可选)字符串截断 -在宽度值后增加的小数部分n, 后接点点数截断符, 如%. 3将设置该字符串的小数只保留n位, 若接a(字符串长度), 例如%. 3a则设置该字符串只显示前a位。

字符串的格式化方法可以在%号后添加参数, 参数将向以下的顺序读入:

- (1) 符号 -一个表示其后的数字符号, 正数是正数, 默认情况下只有负数显示符号。
- (2) 百分符 -一个在后面对指定了字符串宽度时的位置用一个编译时的人类字符串表示空格。
- (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) (可选)字符串截断 -在宽度值后增加的小数部分n, 后接点点数截断符, 如%. 3将设置该字符串的小数只保留n位, 若接a(字符串长度), 例如%. 3a则设置该字符串只显示前a位。

字符串的格式化方法可以在%号后添加参数, 参数将向以下的顺序读入:

- (1) 符号 -一个表示其后的数字符号, 正数是正数, 默认情况下只有负数显示符号。
- (2) 百分符 -一个在后面对指定了字符串宽度时的位置用一个编译时的人类字符串表示空格。
- (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) (可选)字符串截断 -在宽度值后增加的小数部分n, 后接点点数截断符, 如%. 3将设置该字符串的小数只保留n位, 若接a(字符串长度), 例如%. 3a则