

- [数据类型](#)
 - [标识符](#)
 - [变量的命名](#)
 - [全局变量](#)
 - [变量](#)
 - [数据类型列表](#)
 - [Nil](#)
 - [Boolean](#)
 - [Number](#)
 - [String](#)
 - [转义序列](#)
 - [所见即所得](#)
 - [自动转换](#)
 - [Function](#)
 - [标准库表:](#)
 - [Userdata and Thread](#)

数据类型

标识符

变量的命名

位置	可使用的字符
首位	<code>_</code>
	<code>A——Z</code>
	<code>a——z</code>
后位	<code>_</code>
	<code>A——Z</code>
	<code>a——z</code>
	<code>0——9</code>

Tips

- 尽量不要使用大写字母加下划线的命名方式，因为Lua中的特殊关键字就是这样命名的

全局变量

变量赋值就是使用 `=` 符号

一般，lua中可以直接来一个全局变量，即使没有对其赋初值也没有关系，(这个好像Python啊，我的C/C++，(π ^ π))没有赋予初值的家伙，返回的值是nil

```
命令提示符 - lua -i
Microsoft Windows [版本 10.0.19044.2486]
(c) Microsoft Corporation。保留所有权利。

C:\Users\Les>lua -i
Lua 5.1.5 Copyright (C) 1994-2012 Lua.org, PUC-Rio
> print(b)
nil
> b=114514
> print(b)
114514
>
```

特别的，要阐述一个变量被删除，可以赋其值为*nil*

变量

特别的，*lua*中的变量我们直接取名就行了，无需对数据类型进行定义，甚至不需要赋予初值，只要加入他们就好。下面的数据类型列表展示的是*lua*是如何在内部描述你的变量的

使用：

```
type(变量名)  -- 可以查看变量的类型判定
```

数据类型列表

变量类型	描述
nil	无效值
boolean	布尔值，只有TRUE和FALSE
number	双精度类型的浮点实数
string	字符串，由一对双引号或者单引号来进行囊括内部的字符
function	由 <i>C或Lua</i> 编写的函数
userdata	

变量类型	描述
thread	
table	

Nil

nil是一个特殊的值，他这一类中只有nil这一个值，表示的意义是没有，空，他可以表示变量的删除，也是没有赋值的变量的值，给变量赋nil就表示删除他了。

Boolean

仅有两个取值：**false**和**true**，lua中的所有值均可作为判断

false 和 nil -->false

其余值 **包括0和空的字符串**-->true

Number

表示实数，Lua中没有各种数字类型之分，指数和小数都是可选项

e.g

```
4          --整数
0.4        --小数
4.57e-3    --指数
```

String

字符串，可用单引号或双引号表示嵌套时记得单双交替

在5.1版本的更新中，他们可以自我修改

```
b = "hello";
b = string.gsub(b, "hello", "你好"); --修改字符串的一部分
```

但貌似在5.0版本中你只能新给一个变量

```
b = "hello";
a = string.gsub(b, "hello", "你好"); --修改字符串中的一部分
```

转义序列

转义序列	作用
------	----

\a	使得系统响起提示音
----	-----------

转义序列	作用
<code>\b</code>	后退一格
<code>\f</code>	换页(分页)
<code>\n</code>	换行
<code>\r</code>	回车
<code>\t</code>	空格(tab 制表键)
<code>\v</code>	带方框的空格(记得后面要打一个空格，否则就会令第一个字符在方框中)
<code>\\</code>	<code>\</code>
<code>\"</code>	<code>"</code>
<code>\'</code>	<code>'</code>
<code>\[</code>	<code>[</code>
<code>\]</code>	<code>]</code>
<code>\ASCII码</code>	显示对应的字符

所见即所得

还可以使用

```
local page = [[
很长的一段内容
带有多行
]];
```

使用 `[[...]]` 来替代“”或者是“”，在这种区域内的字符串不会进行转义，所见即所得。(^ ▽ ^)

自动转换

lua会自动在字符串和数字之间转换

在未被赋予变量时他们是自由的

```
local t = "10" + 1; --转变为number, 值为11
local z = 10 .. 20; --转变为string, 为1020, ".."这个符号在lua中表示字符串连接符
local e = "hello" + 1; --ERROR
```

但尽量不要对已有类型的变量这样操作，可能会报错

e.g

```
local w = "10"; --string
w = w + 1;      --会被转化成number, 但会被示警
```

在lua中，由于各个变量在初始化时均不用讲明类型，所以一般变量会被定义为自己最初赋予的类型

尽管*Lua*中字符串和数值可以随意转换，但“10”绝不等于10

可用

```
tostring(数字)    --将数字转化为字符串
tonumber("字符串")--将字符串转化为数字(但字符串必须是数字)
--并且这种转换一直有效，在下面使用时依然保留操作
```

Function

函数可以作为一种参数存在于Lua中，这带给了这种语言极大地便利，我们的变量值可赋为一个函数。

Lua可以调用 *C或Lua实现的函数* Lua的标准库全是由C实现的。

标准库表：

String库
table库
I/O库
OS库
算数库
debug库

Userdata and Thread

userdata是将C的数据存放在Lua变量中预定义的操作仅有 *赋值和相等比较* 两种

thread线程操作