

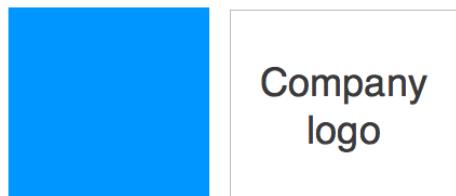


# Lua基础入门教程

version 0.1

*Last generated: October 13, 2020*

---



© 2020 编程者. [Lua基础入门教程](#)

# Table of Contents

## 序言

1.1 前言 .....	2
--------------	---

## 第1章 Lua基础入门教程

1.1 Lua基础语法 .....	3
-------------------	---

1.1.1 Lua变量 .....	7
-------------------	---

# 目录

## Summary: 目录

Lua是一种小巧灵活的脚本语言，在计算机领域很多方面都有应用，最典型的是用于游戏场景、HTTP服务器等，Lua语言学习并不难，可以快速的上手学习。

<https://lua.ren>

[LUA入门教程](#)

# Lua基本语法

**Summary:** 基本Lua语言的基本操作语法

## 基本语法

Lua 学起来非常简单。现在，让我们开始创建我们的第一个 Lua 程序吧！

### 第一个 Lua 程序

Lua 提供交互式编程模式。在这个模式下，你可以一条一条地输入命令，然后立即就可以得到结果。你可以在 shell 中使用 `lua -i` 或者 `lua` 命令启动。输入命令后，按下回车键，就启动了交互模式，显示如下：

```
$ lua -i
$ Lua 5.1.4 Copyright (C) 1994-2008 Lua.org, PUC-Rio
quit to end; cd, dir and edit also available
```

你可以使用如下命令打印输出：

```
$> print("test")
```

按下回车键后，你会得到如下输出结果：

```
'test'
```

### 默认模式编辑

使用 Lua 文件做为解释器的参数启动解释器，然后开始执行文件直到文件结束。当脚本执行结束后，解释器就不在活跃了。

让我们写一个简单的 Lua 程序。所有的 Lua 文件都扩展名都是 `.lua`。因此，将下面的源代码放到 `test.lua` 文件中。

```
print("test")
```

假如你已经设置好 Lua 程序的环境，用下面的命令运行程序：

```
$ lua test.lua
```

我们会得到如下的输出结果：

```
test
```

让我们尝试使用另外的方式运行 Lua 程序。下面是修改后的 test.lua 文件：

```
\#!/usr/local/bin/lua  
print("test")
```

这里，我们假设你的 Lua 解释器程序在 /usr/local/bin/lua 目录下。test.lua 文件中第一行由于以 # 开始而被解释器忽略，运行这个程序可以得到如下的结果：

```
$ chmod a+rwx test.lua  
$ ./test.lua
```

我们会得到如下的的输出结果：

```
test
```

接下来让我们看一下 Lua 程序的基本结构。这样，你可以更容易理解 Lua 编程语言的基本结构单元。

## Lua 中的符号

Lua 程序是由大量的符号组成的。这些符号可以分为关键字、标识符、常量、字符串常量几类。例如，下面的 Lua 语句中包含三个符号：

```
io.write("Hello world, from ",_VERSION,"!\n")
```

这三个符号分别是：

```
io.write
(
"Hello world, from ",_VERSION,"!\n"
)
```

### 注释

注释就是 Lua 程序中的帮助文档，Lua 解释器会自动忽略它们。所有注释都以 --[[ 开始，并以 --]] 结束。如下所示：

```
--[[ my first program in Lua --]]
```

### 标识符

Lua 中标识符是识别变量、函数或者其它用户自定义项的名字。标识符总是以字母或者下划线开始，其后可以是零个或多个字母、下划线或数字。

Lua 标识符中不允许出现任何标点符号，比如，@，\$ 或者 %。Lua 是大小写敏感的语言，因此 Manpower 和 manpower 是 Lua 中两个不同的标识符。下面所列的是一些合法标识符的例子。

mohd	zara	abc	move_name	a_123
myname50	_temp	j	a23b9	RetVal

### 关键字

下面列表中所示的是 Lua 中一小部分保留字。这些保留字不能用作常量、变量以及任何标识符的名字。

and	break	do	else
elseif	end	false	for
function	if	in	local

nil	not	or	repeat
return	then	true	until
while			

## Lua 中的空白符

如果 Lua 程序中某一行只包含空格或者注释，那么这样的一行被称之为空行。Lua 解释器将完全忽略这一行。

在 Lua 中，空白是用来描述空格、制表符、换行符和注释的术语。空白符用于将语句中的一部分与其它部分区分开，使得解释器可以语句中的一个元素，比如 int，何处结束，以及另一个元素从何处开始。因此，在下面的语句中：

```
local age
```

在 local 与 age 之间至少有一个空白符（通常是空格），这个空白符使得解释器可以将 local 与 age 区分开。另一方面，在下面的语句中：

```
fruit = apples + oranges --get the total fruit
```

fruit 与 = 之间以及 = 与 apples 之间的空白符都是可以没有的。但是为了程序的可读性目的，建议你在它们之间使用空白符。

# Lua变量

## Summary: Lua变量概念

## 变量

变量就是给一块内存区域赋予的一个名字。变量使得在程序中就可以修改或读取相应的内存区域中的内容。它可以代表各种不同类型的值，包括函数与表。

变量的名字由字母、数字与下划线组成。它必须是字母或下划线开头。由于 Lua 是字母大小写敏感的，所以大写字母与小写字母是不一样的。Lua 中有八种基本值类型：

在 Lua 语言中，虽然我们没有变量数据类型，但是依据变量的作用域我们可以将变量分为三类：

- 全局变量：除非显示的声明一个局部变量，否则所有的变量都被默认当作全局变量。
- 局部变量：如果我们将一个变量定义为局部变量，那么这么变量的作用域就将被限制在函数内。
- 表字段：这种特殊的变量可以是除了 nil 以外的所有类型，包括函数。

## Lua 变量定义

一个变量定义就意味着告诉解释器在什么地方创建多大的一块存储空间。一个变量定义包括一个可选的类型( type )以及该类型的一个或多个变量名的列表，如下所示：

```
type variable_list;
```

其中，type 是可以选择指定为 local 或者不指定使用默认值 global，variable\_list 是包含一个或多个由逗号分隔的标识符名字。下面是合法变量定义的示例：

```
local    i, j
local    i
local    a,c
```

local i,j 声明定义了两个变量 i 与 j；它命令解释器创建两个名称分别为 i,j 的变量，并且将其作用域限制在局部。

在声明变量的时候可以同时初始化变量（为变量赋初值）。在变量名后跟上等号和一个常量表达式就可以初始化变量。如下所示：

```
type variable_list = value_list;
```

一些例子如下：

```
local d , f = 5 ,10 --声明局部变量 d, f。  
d , f = 5, 10;      --声明全局变量 d, f。  
d, f = 10           --[[声明全局变量 d, f, 其中 f 的值是 nil--]]
```

如果只是定义没有初始化，则静态存储变量被隐式初始化为 nil。

## Lua 变量声明

正如在上面例子看到的那样，为多个变量赋值就是在变量列表后跟上值列表。例子 local d , f = 5 , 10 中,变量列表是 d , f , 值列表是 5 , 10。

Lua 赋值时会将第一个值赋给第一个变量，第二个值赋给第二个变量，依次类推。所以，d 的值是 5,f 的值是 10。

## 示例

下面的示例中，变量被声明在顶部，但是它们在主函数中定义和初始化：

```
-- 变量定义:  
local a, b  
-- 初始化  
a = 10  
b = 30  
print("value of a:", a)  
print("value of b:", b)  
-- 交换变量的值  
b, a = a, b  
print("value of a:", a)  
print("value of b:", b)  
f = 70.0/3.0  
print("value of f", f)
```

上面的代码被编译生成和执行后，会产生如下的结果：

```
value of a:      10
value of b:      30
value of a:      30
value of b:      10
value of f       23.333333333333
```

## Lua 中的左值与右值

Lua 中有两种表达式：

- 左值：引用内存位置的表达式被称之为左值表达式。左值表达式既可以出现在赋值符号的左边也可以出现在赋值符号的右边。
- 右值：术语“右值”指存在内存某个位置的数据值。我们不能为右值表达式赋值，也就是说右值表达式只可能出现在赋值符号的右边，而不可能出现在赋值符号的左边。

变量属于左值表达式，所以它可以在赋值符号的左边。数值常量属于右值表达式，它不能被赋值也不能出现在赋值符号的左边。下面是合法的语句：

```
g = 20
```

但是，下面的语句是非法的，它会产生生成时错误：

```
10 = 20
```

在 Lua 语言中，除了上面讲的这种赋值，还允许在一个赋值语句中存在多个左值表达式与多个右值表达式。如下所示：

```
g, l = 20, 30
```

在这个语句中，g 被赋值为 20，l 被赋值为 30。