Lua 基本语法 Lua 变量

Lua 数据类型

Lua是动态类型语言，变量不要类型定义,只需要为变量赋值。 值可以存储在变量中，作为参数传递或结果返回。

Lua中有8个基本类型分别为：nil、boolean、number、string、userdata、function、thread和table。

数据类型	描述
nil	这个最简单，只有值nil属于该类，表示一个无效值（在条件表达式中相当于false）。
boolean	包含两个值：false和true。
number	表示双精度类型的实浮点数
string	字符串由一对双引号或单引号来表示
function	由 C 或 Lua 编写的函数
userdata	表示任意存储在变量中的C数据结构
thread	表示执行的独立线路，用于执行协同程序
table	Lua 中的表（table）其实是一个"关联数组"（associative arrays），数组的索引可以是数字或者是字符串。在 Lua 里，table 的创建是通过"构造表达式"来完成，最简单构造表达式是{}，用来创建一个空表。

我们可以使用type函数测试给定变量或者值的类型：

```
print(type("Hello world"))      --> string
print(type(10.4*3))             --> number
print(type(print))              --> function
print(type(type))               --> function
print(type(true))               --> boolean
print(type(nil))                --> nil
print(type(type(X)))            --> string
```

nil（空）

nil 类型表示一种没有任何有效值，它只有一个值 -- nil，例如打印一个没有赋值的变量，便会输出一个 nil 值：

```
> print(type(a))
nil
>
```

对于全局变量和 table，nil 还有一个"删除"作用，给全局变量或者 table 表里的变量赋一个 nil 值，等同于把它们删掉，执行下面代码就知：

```
tab1 = { key1 = "val1", key2 = "val2", "val3" }
for k, v in pairs(tab1) do
    print(k .. " - " .. v)
end

tab1.key1 = nil
for k, v in pairs(tab1) do
    print(k .. " - " .. v)
end
```

nil 作比较时应该加上双引号 `"`：

```
> type(X)
nil
> type(X)==nil
false
> type(X)=="nil"
true
>
```

`type(X)==nil` 结果为 **false** 的原因是因为 `type(type(X))==string`。

---

boolean（布尔）

boolean 类型只有两个可选值：true（真）和 false（假），Lua 把 false 和 nil 看作是"假"，其他的都为"真":

```
print(type(true))
print(type(false))
print(type(nil))

if false or nil then
    print("至少有一个是 true")
else
    print("false 和 nil 都为 false!")
end
```

以上代码执行结果如下：

```
$ lua test.lua
boolean
boolean
```

```
nil
false 和 nil 都为 false!
```

number（数字）

Lua 默认只有一种 number 类型 -- double（双精度）类型（默认类型可以修改 luaconf.h 里的定义），以下几种写法都被看作是 number 类型：

```
print(type(2))
print(type(2.2))
print(type(0.2))
print(type(2e+1))
print(type(0.2e-1))
print(type(7.8263692594256e-06))
```

运行实例 »

以上代码执行结果：

```
number
number
number
number
number
number
```

string（字符串）

字符串由一对双引号或单引号来表示。

```
string1 = "this is string1"
string2 = 'this is string2'
```

也可以用 2 个方括号 "[[]]" 来表示"一块"字符串。

```
html = [[
<html>
<head></head>
<body>
    <a href="http://www.runoob.com/">菜鸟教程</a>
</body>
</html>
]]
print(html)
```

以下代码执行结果为：

```
<html>
<head></head>
<body>
    <a href="http://www.runoob.com/">菜鸟教程</a>
</body>
</html>
```

在对一个数字字符串上进行算术操作时，Lua 会尝试将这个数字字符串转成一个数字:

```
> print("2" + 6)
8.0
> print("2" + "6")
8.0
> print("2 + 6")
2 + 6
> print("-2e2" * "6")
-1200.0
> print("error" + 1)
stdin:1: attempt to perform arithmetic on a string value
stack traceback:
    stdin:1: in main chunk
    [C]: in ?
>
```

以上代码中"error" + 1执行报错了，字符串连接使用的是 .. ，如：

```
> print("a" .. 'b')
ab
> print(157 .. 428)
157428
>
```

使用 # 来计算字符串的长度，放在字符串前面，如下实例：

```
> len = "www.runoob.com"
> print(#len)
14
> print(#"www.runoob.com")
14
>
```

table（表）

在 Lua 里，table 的创建是通过"构造表达式"来完成，最简单构造表达式是{}，用来创建一个空表。也可以在表里添加一些数据，直接初始化表:

```
-- 创建一个空的 table
local tbl1 = {}

-- 直接初始表
local tbl2 = {"apple", "pear", "orange", "grape"}
```

Lua 中的表（table）其实是一个"关联数组"（associative arrays），数组的索引可以是数字或者是字符串。

```
-- table_test.lua 脚本文件
a = {}
a["key"] = "value"
key = 10
a[key] = 22
a[key] = a[key] + 11
for k, v in pairs(a) do
    print(k .. " : " .. v)
end
```

脚本执行结果为：

```
$ lua table_test.lua
key : value
10 : 33
```

不同于其他语言的数组把 0 作为数组的初始索引，在 Lua 里表的默认初始索引一般以 1 开始。

```
-- table_test2.lua 脚本文件
local tbl = {"apple", "pear", "orange", "grape"}
for key, val in pairs(tbl) do
    print("Key", key)
end
```

脚本执行结果为：

```
$ lua table_test2.lua
Key    1
Key    2
Key    3
Key    4
```

table 不会固定长度大小，有新数据添加时 table 长度会自动增长，没初始的 table 都是 nil。

```
-- table_test3.lua 脚本文件
a3 = {}
for i = 1, 10 do
    a3[i] = i
end
a3["key"] = "val"
print(a3["key"])
print(a3["none"])
```

脚本执行结果为：

```
$ lua table_test3.lua
val
nil
```

---

function（函数）

在 Lua 中，函数是被看作是"第一类值（First-Class Value）"，函数可以存在变量里:

```
-- function_test.lua 脚本文件
function factorial1(n)
    if n == 0 then
        return 1
    else
        return n * factorial1(n - 1)
    end
end
print(factorial1(5))
factorial2 = factorial1
print(factorial2(5))
```

脚本执行结果为：

```
$ lua function_test.lua
120
120
```

function 可以以匿名函数（anonymous function）的方式通过参数传递:

```
-- function_test2.lua 脚本文件
function testFun(tab,fun)
    for k ,v in pairs(tab) do
```

```
        print(fun(k,v));
    end
end


tab={key1="val1",key2="val2"};
testFun(tab,
function(key,val)--匿名函数
    return key.."="..val;
end
);
```

脚本执行结果为：

```
$ lua function_test2.lua
key1 = val1
key2 = val2
```

thread（线程）

在 Lua 里，最主要的线程是协同程序（coroutine）。它跟线程（thread）差不多，拥有自己独立的栈、局部变量和指令指针，可以跟其他协同程序共享全局变量和其他大部分东西。

线程跟协程的区别：线程可以同时多个运行，而协程任意时刻只能运行一个，并且处于运行状态的协程只有被挂起（suspend）时才会暂停。

userdata（自定义类型）

userdata 是一种用户自定义数据，用于表示一种由应用程序或 C/C++ 语言库所创建的类型，可以将任意 C/C++ 的任意数据类型的数据（通常是 struct 和 指针）存储到 Lua 变量中调用。

← Lua 基本语法 Lua 变量 →