【Lua】【型別】函數function

函數宣告

函數可以使用`function`來做宣告，並以`end`結束

```
function hello()
  print("Hello, World")
end

-- 等同以下
hello = function()
  print("Hello, World")
end
```

```
function genFun()
  return function()
    print("Hello, World")
  end
end


function callFun(f)
  f()
end


local _f = genFun() -- # get a function
callFun(_f) -- print("Hello, World")
```

函數傳遞

```
function plus(a, b)
    print(a,b, a + b)
    return a+b ,a ,b
end

local anser, A, B = plus(1,2)
print(A.."+"..B.."="..anser)

-- 1 2 3
-- 1+2=3
```

回傳值規則

若返回值個數大於接收變量的個數，多餘的返回值會被忽略掉； 若返回值個數小 於參數個數，從左向右，沒有被返回值初始化的變量會被初始化為 nil。

```
function init() --init 函□ 返回两个值 1 和 "lua"
return 1, "lua"
end
x = init()
print(x)
x, y, z = init()
print(x, y, z)

--output
1
1 lua nil
```

□一个函□有一个以上返回值，且函□□用不是一个列表表达式的最后一个元素，那么函□ □用只会□生一个返回值，也就是第一个返回值。

```
local function init() -- init 函□ 返回两个值 1 和 "lua"
return 1, "lua"
end
```

```
local x, y, z = init(), 2 -- init 函□的位置不在最后，此□只返回 1
print(x, y, z) -->output 1 2 nil
local a, b, c = 2, init() -- init 函□的位置在最后，此□返回 1 和 "lua"
print(a, b, c) -->output 2 1
```

```
local function init() -- init 函□ 返回两个值 1 和 "lua"
return 1, "lua"
end
local x, y, z = init(), 2 -- init 函□的位置不在最后，此□只返回 1
print(x, y, z) -->output 1 2 nil
local a, b, c = 2, init() -- init 函□的位置在最后，此□返回 1 和 "lua"
print(a, b, c) -->output 2 1
```

不定長參數

```
function hello(...)
  local members = {...}
  print("有多少成員: ", #members)
  for _,m in pairs(members) do
    print("Hello, "..m)
  end
end

-- hello("Bob", "Lua", "Luna", "Selene")
-- 有多少成員:  4
-- Hello, Bob
-- Hello, Lua
-- Hello, Luna
-- Hello, Selene
```

```
function mylog(...)
 local arr = {...}
 table.insert(arr,1,"\n=====[mylog_begin]=====")
 table.insert(arr,"=====[mylog_end]=====\n")
 local print_str = table.concat(arr,"\n")
 -- for key,value in pairs(arr) do
 --  -- ngx.say(key..":"..value)
 --  ngx.say(value)
 -- end
 ngx.log(ngx.ERR,print_str)
end


mylog("開始",1,2,3,"結束")
-- =====[mylog_begin]=====
-- 開始
-- 1
-- 2
-- 3
-- 結束
-- =====[mylog_end]=====
```

立即呼叫函式表達式(IIFE) 與 匿名函式

```
(function ()
  print("Hello, World")
end)()
```

多值返回

```
function get(dict, key, default)
  return (dict[key] or default), dict[key] ~= nil
end
```

```
_d = {}
_v, found = get(_d, "key", "value")
print(_v) -- => Output: value
print(found) -- => Output: false
```

```
function Person(args)
  local person = args
  person.name = args.name or "Bob"
  person.age = args.age or 20
  person.address = args.address or ""
  return person
end

p1 = Person {
  "位置參數1",
  age = 13,
  address = "secret",
  "位置參數2",
}

print("Name:    ", p1.name)
print("Age:     ", p1.age)
print("Address: ", p1.address)
print("-------------")
print("位置參數：", p1[1], p1[2])
```

```
Name: Bob
Age: 13
Address: secret

位置參數： 位置參數1 位置參數2
```

常用的內建函數

基本

- ipairs
- pairs
- print
- require
- tonumber
- tostring
- type

輸入/輸出

估計本系列不怎麼會提到，先放這XD。

- io.close
- io.input
- io.lines
- io.open
- io.output
- io.popen
- io.read
- io.stderr
- io.stdin
- io.stdout
- io.tmpfile
- io.type
- io.write
- file:close
- file:flush
- file:lines

- file:read
- file:seek
- file:write

數學

- math.abs
- math.ceil
- math.exp
- math.floor
- math.huge (無限大)
- math.log
- math.max
- math.maxinteger
- math.min
- math.mininteger
- math.pi (PI 常數)
- math.random
- math.randomseed
- math.tointeger
- math.type

作業系統

- os.clock
- os.date
- os.difftime
- os.execute
- os.exit
- os.getenv
- os.time

> Lua 5.1前一個和`os.getenv`很像的函數`setfenv`其實也蠻常用的，後來被移除，改用其他方式。
> 當前還有很多是使用Lua 5.1，這雖然是需要知道的事情，不過其實其與作業系統無關。

字串

- string.byte
- string.char
- string.find
- string.format
- string.len
- string.reverse
- string.sub
- utf8.char
- utf8.codes
- utf8.len

修訂版本 #7
★由 treeman 建立於 22 ﷺG﷿﷿ 2023 21:49:49
✎由 treeman 更新於 25 ﷺC﷿﷿ 2023 16:56:34