[Golang学习笔记 2](#_Toc12230)

[RUNOOB.COM----Go语言教程笔记 2](#_Toc32220)

[简介，特色，用途 2](#_Toc18405)

[Go 基本组成 3](#_Toc3867)

[Go Env 名词解释 3](#_Toc17146)

[CGO\_ENABLED 4](#_Toc24202)

[GOARCH 4](#_Toc11074)

[GOBIN 4](#_Toc22539)

[GOCHAR 4](#_Toc1360)

[GOEXE 4](#_Toc23892)

[GOHOSTARCH 4](#_Toc1470)

[GOOS 4](#_Toc19278)

[GOHOSTOS 4](#_Toc3495)

[GOPATH 4](#_Toc24370)

[GORACE 5](#_Toc27582)

[GOROOT 5](#_Toc8641)

[GOTOOLDIR 5](#_Toc3894)

[Go语言基础语法 5](#_Toc28939)

[行分隔符 5](#_Toc733)

[标识符 5](#_Toc17825)

[关键字 6](#_Toc31254)

[空格 6](#_Toc30248)

[Go语言数据类型 6](#_Toc18066)

[数字类型 8](#_Toc5281)

[其他数字类型 8](#_Toc1280)

[Go包引用--别名,省略包名 9](#_Toc6889)

[Go语言变量 9](#_Toc13)

[变量声明 10](#_Toc2106)

[多变量声明 10](#_Toc1113)

[值类型和引用类型 11](#_Toc24503)

[:= 赋值操作符 12](#_Toc30049)

[并行/同时 赋值 & 空白标识符 12](#_Toc14403)

[Go语言常量 13](#_Toc12907)

[使用内置函数返回值作为常量的赋值 13](#_Toc20537)

[Iota 14](#_Toc5096)

[Go语言运算符 15](#_Toc27585)

[Go语言条件语句 18](#_Toc19702)

[Go语言循环语句 20](#_Toc15051)

[For循环 20](#_Toc6627)

[循环嵌套 21](#_Toc11264)

[循环控制语句 21](#_Toc11239)

[Break语句 21](#_Toc31788)

[Continue语句 21](#_Toc866)

[Goto语句 22](#_Toc3555)

[无限循环 22](#_Toc13671)

[Go语言函数 22](#_Toc13174)

[函数参数 23](#_Toc15972)

[值传递 24](#_Toc9592)

[引用传递 24](#_Toc21271)

[函数用法 24](#_Toc29268)

[函数作为值 24](#_Toc27198)

[闭包 25](#_Toc22479)

[方法 25](#_Toc19610)

[Go语言变量作用域 26](#_Toc15727)

[初始化局部和全局变量 27](#_Toc9793)

[Go语言数组 27](#_Toc32525)

[多维数组 28](#_Toc11093)

[向函数传递数组 30](#_Toc28969)

[Go语言指针 30](#_Toc13207)

[空指针 31](#_Toc5308)

[Go 指针数组 32](#_Toc2930)

[Go 指向指针的指针（二级指针） 32](#_Toc24206)

[Go 向函数传递指针参数 32](#_Toc28270)

[Go语言结构体 32](#_Toc22117)

[Go语言切片 33](#_Toc32299)

[Go语言范围(Range) 35](#_Toc13447)

[Go 语言Map(集合) 35](#_Toc25177)

[Go 语言递归函数 36](#_Toc6241)

# Golang学习笔记

# RUNOOB.COM----Go语言教程笔记

## 简介，特色，用途

简介 -- Go 是从2007年末开发的一个开源编程语言.

特色 -- 简洁,快速,安全,并行,有趣,开源,内存管理,v数组安全,编译迅速

用途 -- Go 语言被设计成一门应用于搭载 Web 服务器，存储集群或类似用途的巨型中央服务器的系统编程语言.对于高性能分布式系统领域而言，Go 语言无疑比大多数其它语言有着更高的开发效率。它提供了海量并行的支持，这对于游戏服务端的开发而言是再好不过了。

## Go 基本组成

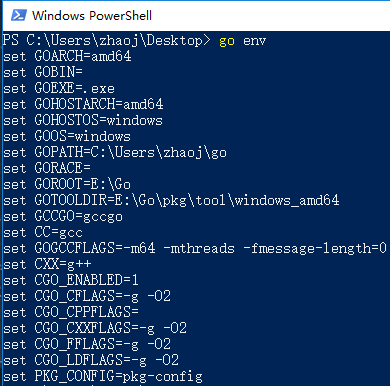
包声明,引入包,函数,变量,语句 & 表达式,注释

注意事项:

1. package main表示一个可独立执行的程序，每个 Go 应用程序都包含一个名为 main 的包。
2. main 函数是每一个可执行程序所必须包含的，一般来说都是在启动后第一个执行的函数（如果有 init() 函数则会先执行该函数）
3. 单行注释是最常见的注释形式，你可以在任何地方使用以 // 开头的单行注释。多行注释也叫块注释，均已以 /\* 开头，并以 \*/ 结尾，且不可以嵌套使用，多行注释一般用于包的文档描述或注释成块的代码片段
4. (重点)当标识符（包括常量、变量、类型、函数名、结构字段等等）以一个大写字母开头，如：Group1，那么使用这种形式的标识符的对象就可以被外部包的代码所使用（客户端程序需要先导入这个包），这被称为导出（像面向对象语言中的 public）；标识符如果以小写字母开头，则对包外是不可见的，但是他们在整个包的内部是可见并且可用的（像面向对象语言中的 protected ）

## Go Env 名词解释

使用go env 可以查看当前的go的环境变量



### CGO\_**ENABLED**

指明cgo工具是否可用的标识。

### GOARCH

程序构建环境的目标计算架构。

### GOBIN

存放可执行文件的目录的绝对路径。

### GOCHAR

程序构建环境的目标计算架构的单字符标识。

### GOEXE

可执行文件的后缀。

### GOHOSTARCH

程序运行环境的目标计算架构。

### GOOS

程序构建环境的目标操作系统。

### GOHOSTOS

程序运行环境的目标操作系统。

### GOPATH

工作区目录的绝对路径。

### GORACE

用于数据竞争检测的相关选项。

### GOROOT

Go语言的安装目录的绝对路径。

### GOTOOLDIR

Go工具目录的绝对路径。

## Go语言基础语法

### 行分隔符

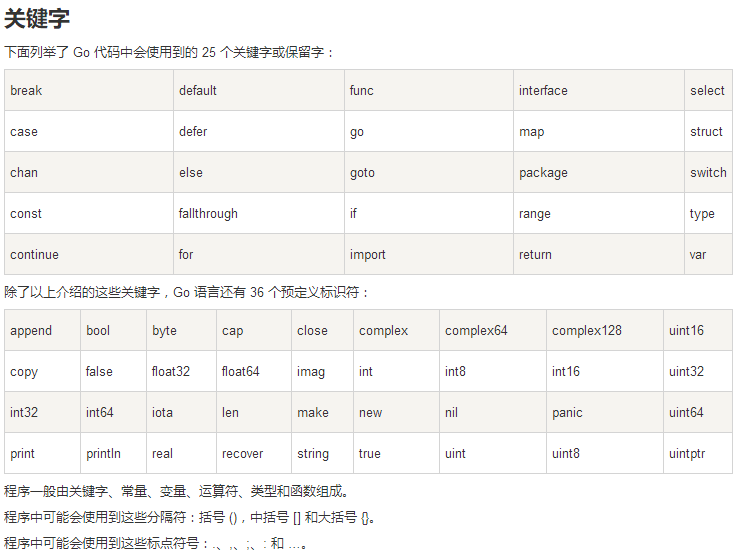
在 Go 程序中，一行代表一个语句结束。每个语句不需要像 C 家族中的其它语言一样以分号 ; 结尾，因为这些工作都将由 Go 编译器自动完成。

如果你打算将多个语句写在同一行，它们则必须使用 ; 人为区分，但在实际开发中我们并不鼓励这种做法。

### 标识符

标识符用来命名变量、类型等程序实体。一个标识符实际上就是一个或是多个字母(A~Z和a~z)数字(0~9)、下划线\_组成的序列，但是第一个字符必须是字母或下划线而不能是数字。

### 关键字



### 空格

Go 语言中变量的声明必须使用空格隔开，如：var age int;

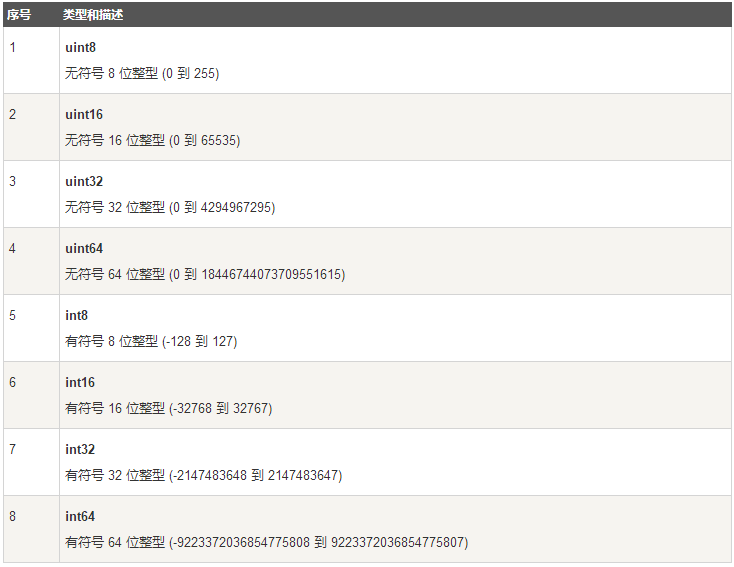
### Go语言数据类型

在 Go 编程语言中，数据类型用于声明函数和变量。

数据类型的出现是为了把数据分成所需内存大小不同的数据，编程的时候需要用大数据的时候才需要申请大内存，就可以充分利用内存。



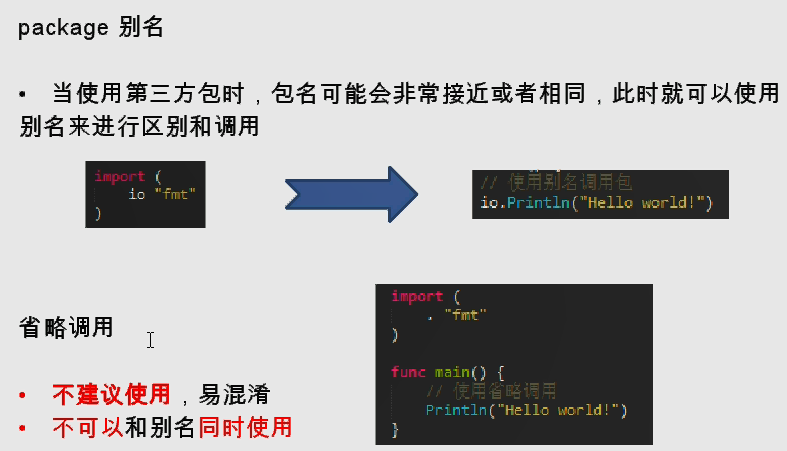
### 数字类型



### 其他数字类型



### Go包引用--别名,省略包名



### Go语言变量

变量来源于数学，是计算机语言中能储存计算结果或能表示值抽象概念。变量可以通过变量名访问。

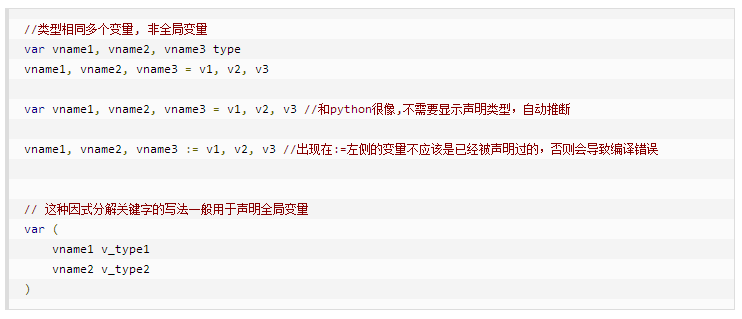
Go 语言变量名由字母、数字、下划线组成，其中首个字母不能为数字。

声明变量的一般形式是使用 var 关键字：var identifier type

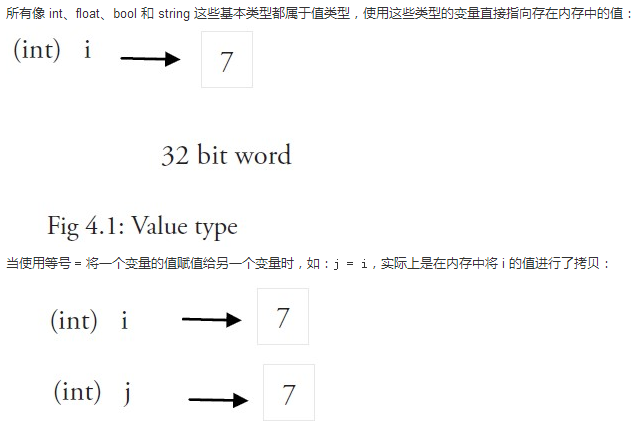
### 变量声明



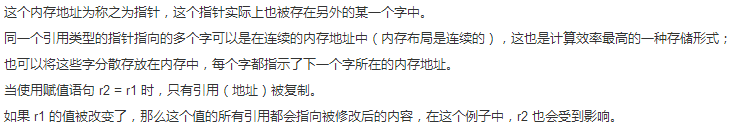
### 多变量声明



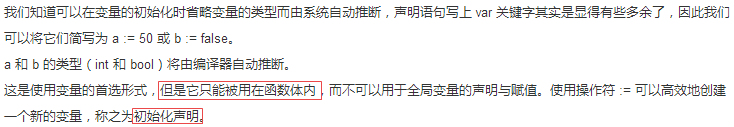
### 值类型和引用类型

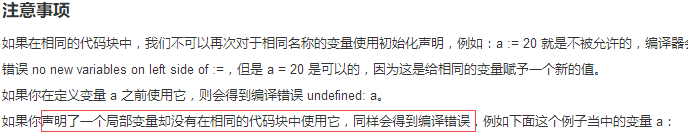






### := 赋值操作符



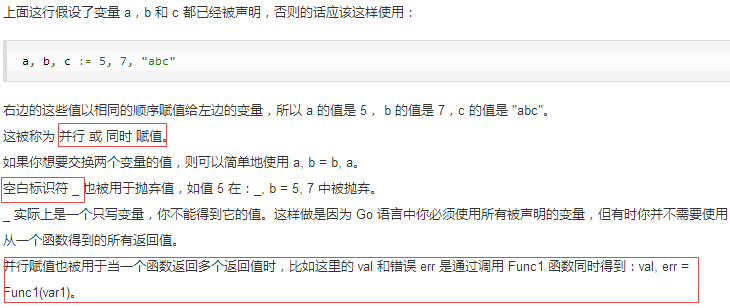




### 并行/同时 赋值 & 空白标识符

并行赋值的作用是 快速定义 和 接收返回的方法传递的多参数

空白标识符用于接收不想要的多参数返回的补位符号. 例: \_,b =hello\_world(‘hello’)



### Go语言常量

常量是一个简单值的标识符，在程序运行时，不会被修改的量。

常量中的数据类型只可以是布尔型、数字型（整数型、浮点型和复数）和字符串型。

常量的定义格式：

const identifier [type] = value

你可以省略类型说明符 [type]，因为编译器可以根据变量的值来推断其类型。

显式类型定义： const b string = "abc"

隐式类型定义： const b = "abc"

多个相同类型的声明可以简写为:

const c\_name1, c\_name2 = value1, value2

常量还可以用作枚举:

const(

a int = 1

b float = 1.20

C bool = true

)

常量还可以延续赋值:

const(

A = 1

B

C=true

D

E=iota

)

### 使用内置函数返回值作为常量的赋值



const (

a = "abc"

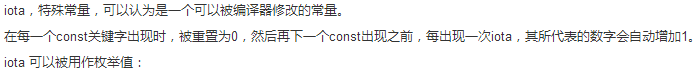
b = len(a)

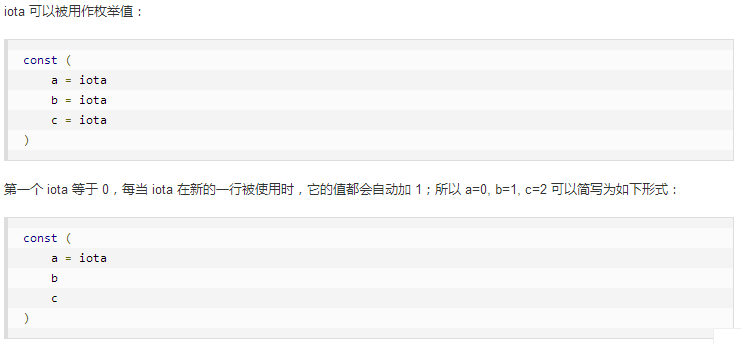
c = unsafe.Sizeof(a)

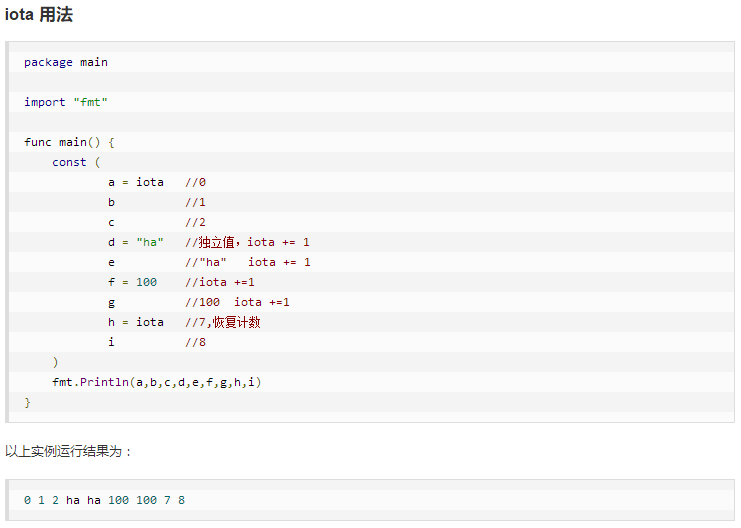
)

### Iota

仅能在方法中定义，因为iota多次被调用是动态的内容,编译器会报错







### Go语言运算符

运算符用于在程序运行时执行数学或逻辑运算,

Go 语言内置的运算符有:算术运算符,关系运算符,逻辑运算符,位运算符,赋值运算符,其他运算符





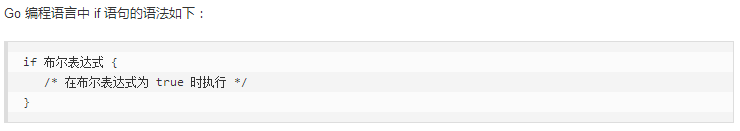




### Go语言条件语句



**IF的使用**



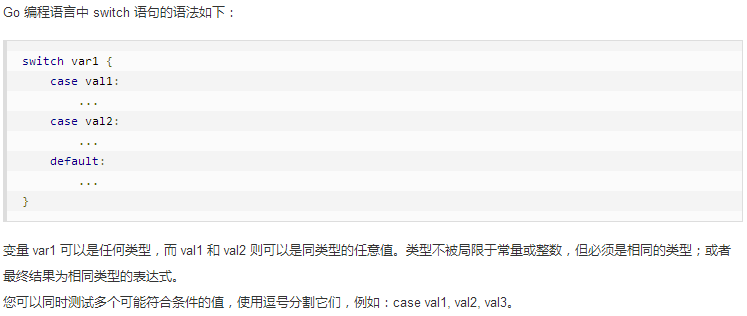
**IF...ELSE的使用**



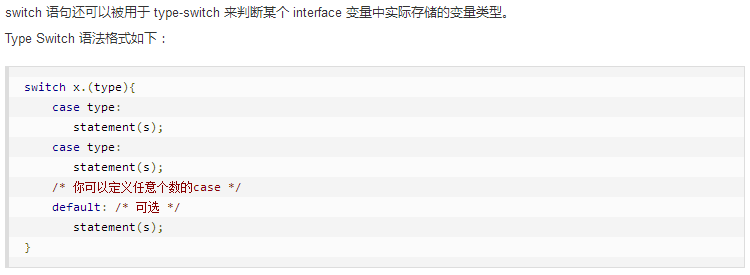
**SWITCH的使用 -- 不需要添加break**

switch 语句用于基于不同条件执行不同动作，每一个 case 分支都是唯一的，从上直下逐一测试，直到匹配为止。。

switch 语句执行的过程从上至下，直到找到匹配项，匹配项后面也不需要再加break

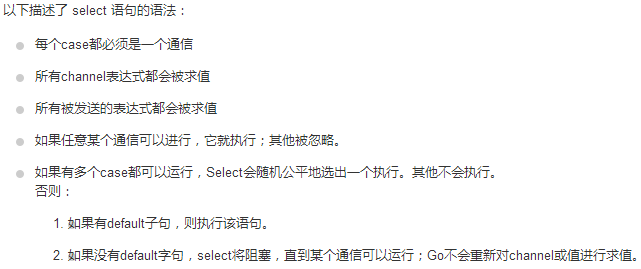


**TYPE SWITCH 使用**

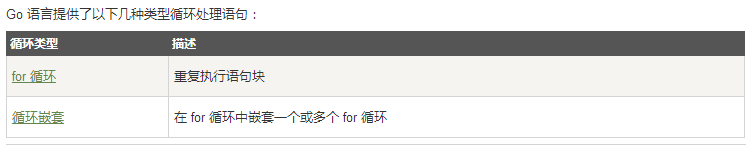


**Select 使用**

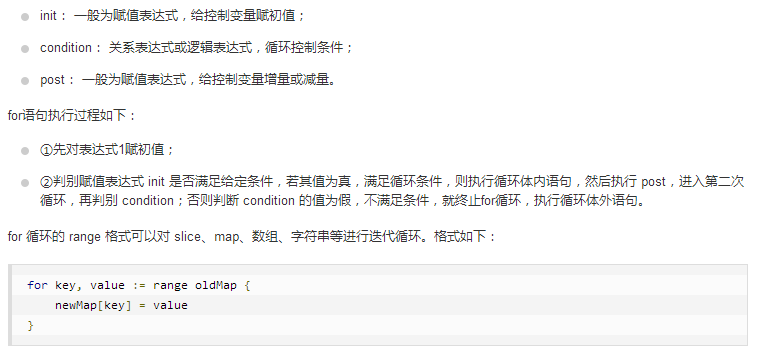




### Go语言循环语句



### For循环



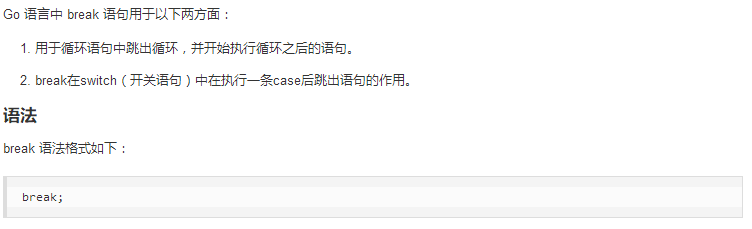
### 循环嵌套



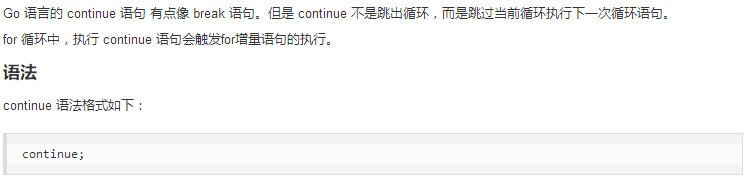
### 循环控制语句

循环控制语句可以控制循环体内语句的执行过程。

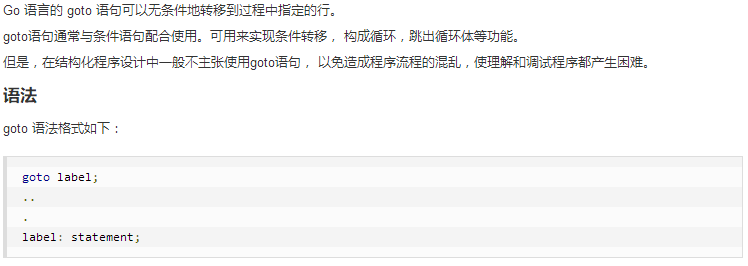
### Break语句



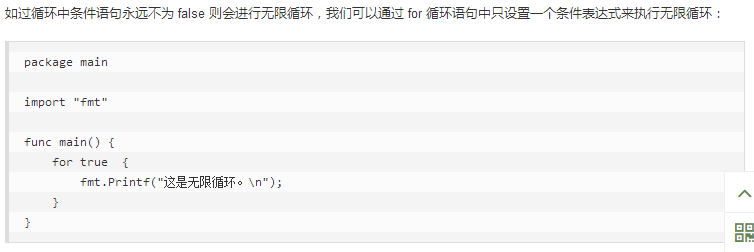
### Continue语句



### Goto语句



### 无限循环



## Go语言函数

1.函数是基本的代码块，用于执行一个任务。

2.Go 语言最少有个 main() 函数。

3.你可以通过函数来划分不同功能，逻辑上每个函数执行的是指定的任务。

4.函数声明告诉了编译器函数的名称，返回类型，和参数。

Go 语言标准库提供了多种可动用的内置的函数。例如，len() 函数可以接受不同类型参数并返回该类型的长度。如果我们传入的是字符串则返回字符串的长度，如果传入的是数组，则返回数组中包含的函数个数。



### 函数参数

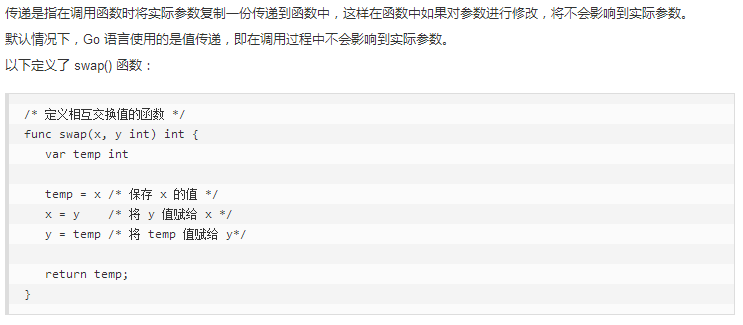
函数如果使用参数，该变量可称为函数的形参。

形参就像定义在函数体内的局部变量。

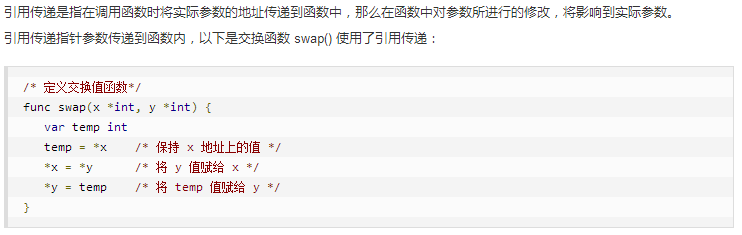
调用函数，可以通过两种方式来传递参数



### 值传递



### 引用传递



### 函数用法



### 函数作为值



### 闭包

//闭包函数返回int ,所以外层函数是需要定义是返回int

func getSequence() func() **int** {

i:=0

return func() int {

i+=1

return i

}}

//闭包函数返回string,所以外层函数是需要定义是返回string

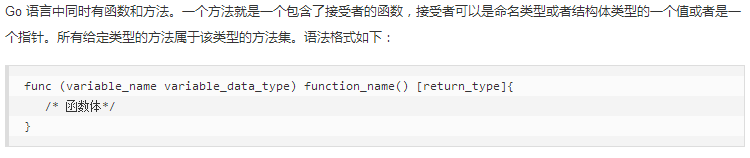
func getSequence() func() **string** {  
 i := 0  
 return func() string {  
   
 }  
}

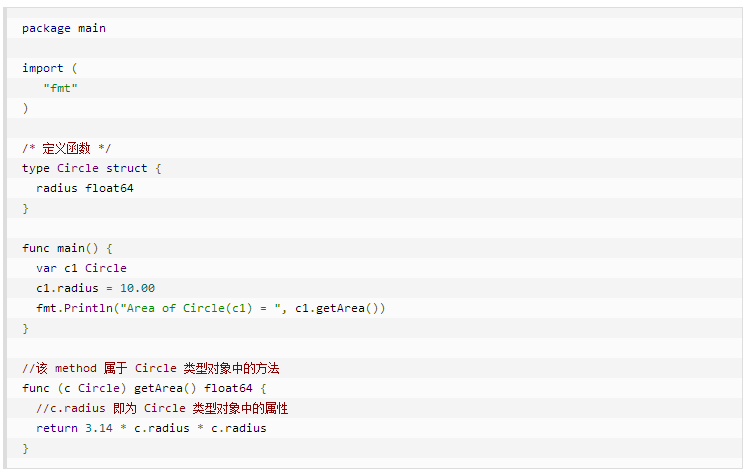
闭包返回的方法必须是带int类型的，说明闭包函数其实是一个地址段



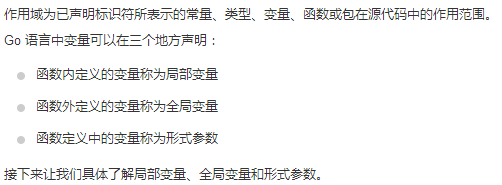
### 方法

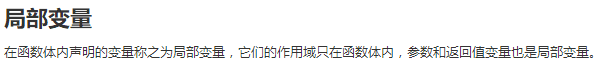
就是将一个方法附加到一个结构体中，让其具有成员方法





### Go语言变量作用域



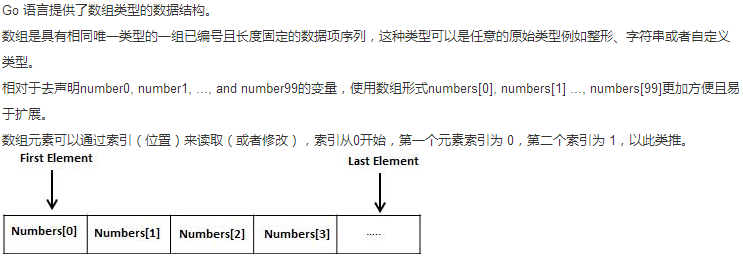


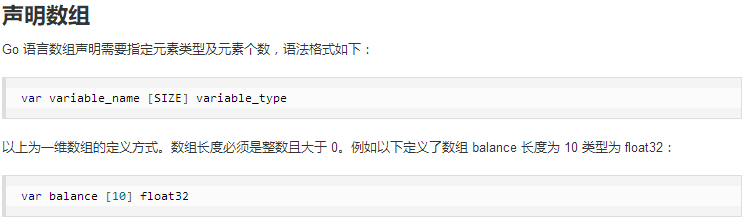


### 初始化局部和全局变量

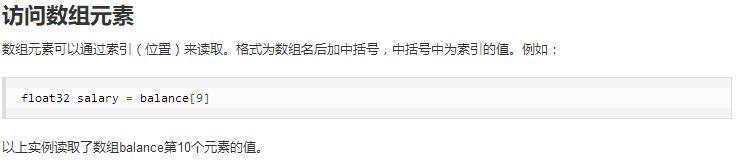


## Go语言数组

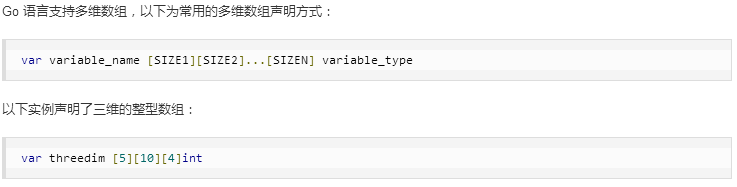


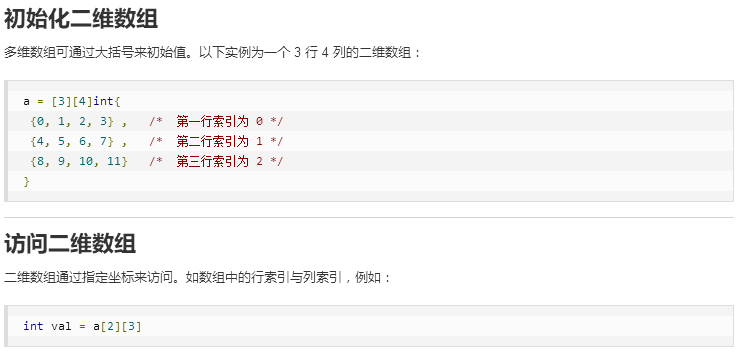
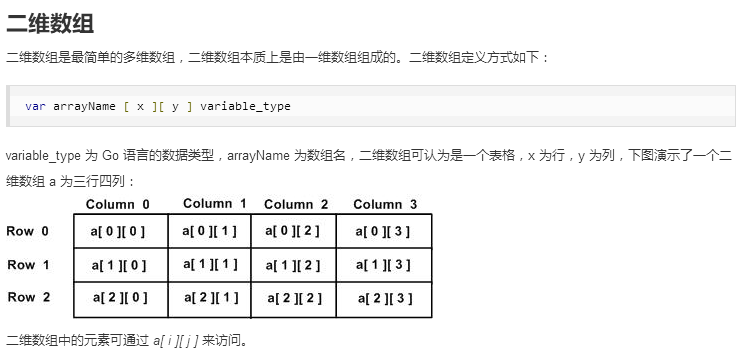






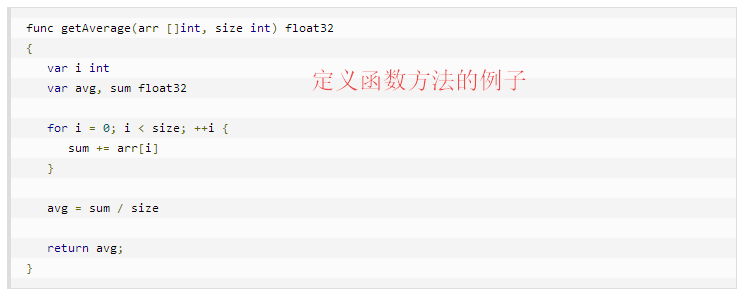
### 多维数组





### 向函数传递数组





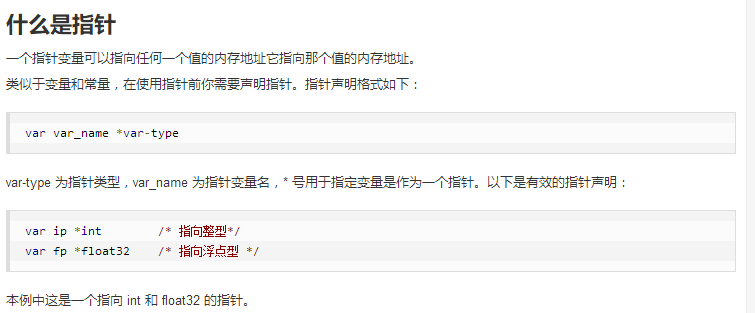
## Go语言指针

Go 语言中指针是很容易学习的，Go 语言中使用指针可以更简单的执行一些任务。

接下来让我们来一步步学习 Go 语言指针。

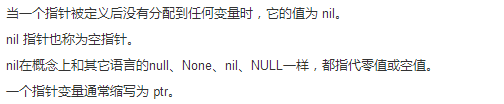
我们都知道，变量是一种使用方便的占位符，用于引用计算机内存地址。

Go 语言的取地址符是 &，放到一个变量前使用就会返回相应变量的内存地址。





### 空指针

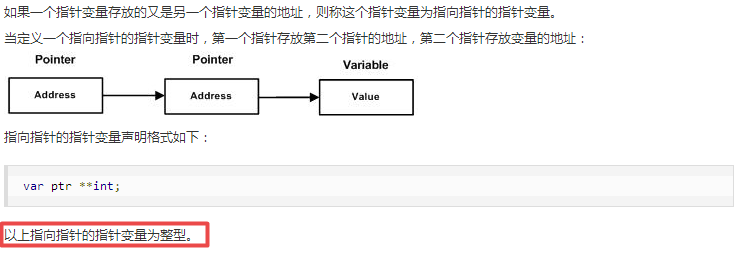




### Go 指针数组

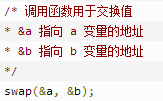
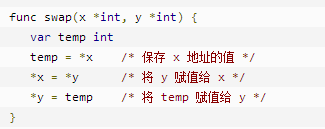
定义一个指针数组来存储地址

### Go 指向指针的指针（二级指针）



### Go 向函数传递指针参数

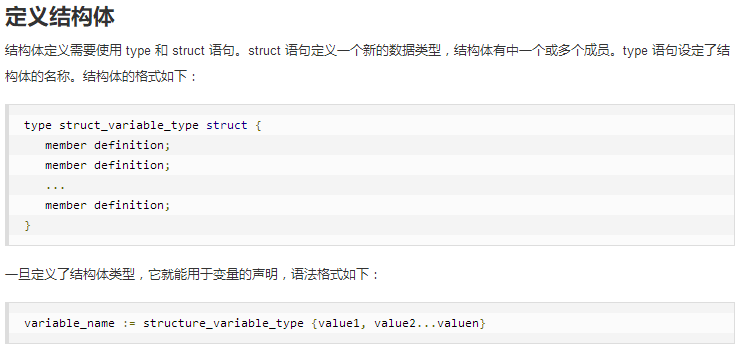
Go 语言允许向函数传递指针，只需要在函数定义的参数上设置为指针类型即可。

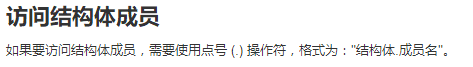


## Go语言结构体

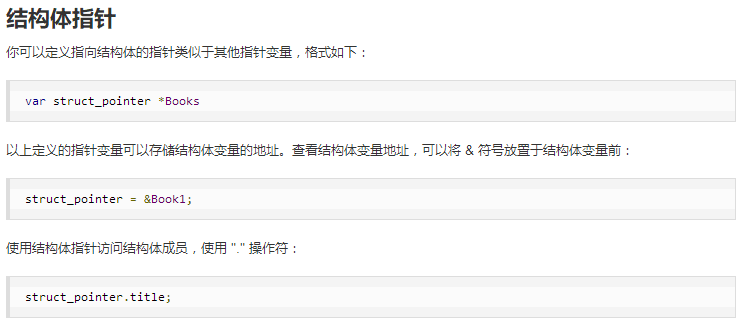
Go 语言中数组可以存储同一类型的数据，但在结构体中我们可以为不同项定义不同的数据类型。

结构体是由一系列具有相同类型或不同类型的数据构成的数据集合。





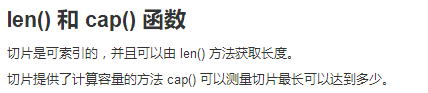




## Go语言切片

Go 语言切片是对数组的抽象。

Go 数组的长度不可改变，在特定场景中这样的集合就不太适用，Go中提供了一种灵活，功能强悍的内置类型切片("动态数组"),与数组相比切片的长度是不固定的，可以追加元素，在追加时可能使切片的容量增大。



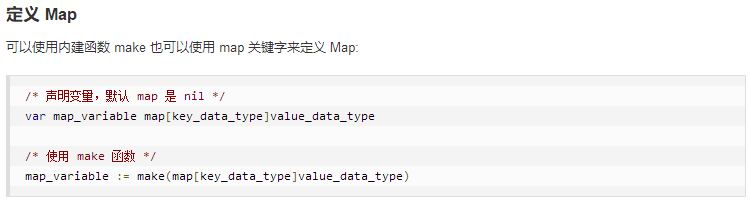
## Go语言范围(Range)

Go 语言中 range 关键字用于for循环中迭代数组(array)、切片(slice)、通道(channel)或集合(map)的元素。在数组和切片中它返回元素的索引值，在集合中返回 key-value 对的 key 值。

## Go 语言Map(集合)

Map 是一种无序的键值对的集合。Map 最重要的一点是通过 key 来快速检索数据，key 类似于索引，指向数据的值。

Map 是一种集合，所以我们可以像迭代数组和切片那样迭代它。不过，Map 是无序的，我们无法决定它的返回顺序，这是因为 Map 是使用 hash 表来实现的。



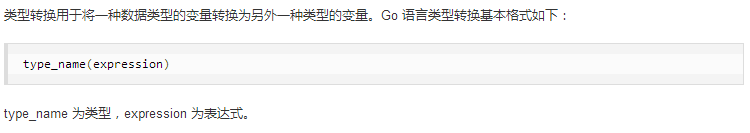


## Go 语言递归函数

递归，就是在运行的过程中调用自己。

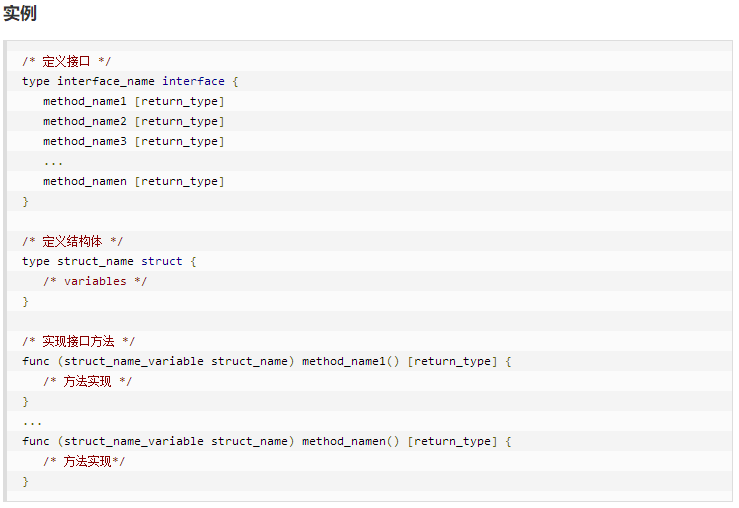


## Go语言类型转换



## Go语言接口

Go 语言提供了另外一种数据类型即接口，它把所有的具有共性的方法定义在一起，任何其他类型只要实现了这些方法就是实现了这个接口。



## Go错误处理

