# Go语言编程重点笔记

### Go语言的核心开发团队

Ken Thompson 1983年图灵奖获得者，C语言的主要发明人

Rob Pike 贝尔实验室Unix团队的成员，和Ken共创出广泛使用的UTF-8编码  
Robert Griesemer：曾协作制作Java的HotSpot编译器，和Chrome浏览器的JavaScript引擎V8

### Go语言诞生的小故事

Google为什么要创建出Go语言？

1.计算机硬件技术更新频繁，目前主流编程语言的发展明显落后于硬件，不能合理利用多核多CPU的优势提升软件系统性能。

2.软件系统复杂度越来越高，维护成本越来越高，目前缺乏一个足够简洁高效的编程语言。

现有的编程语言存在这些问题：

1.风格不统一

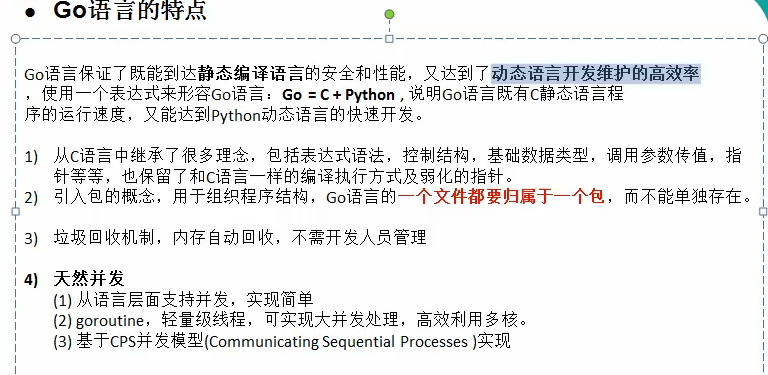
2.计算能力不够

3.处理大并发不够优秀

3.企业运行维护很多c/c++的项目，虽然运行速度很快，但是编译速度很慢，同时还存在**内存泄露**的一系列的困扰需要解决。

Go语言既兼顾编译语言的速度，又同时拥有脚本语言的开发速度。

### Go语言的特点



1.Go语言指针案例：



2.**引入包的概念**，用于组织结构，**Go语言的文件都要归属于一个包（类似于Java）**,而不能单独存在。

**Go语言同一个文件目录**里面的Go文件**不能有多个package**

**3.引入了自动垃圾回收机制**

**4.天然并发(重要特点)**

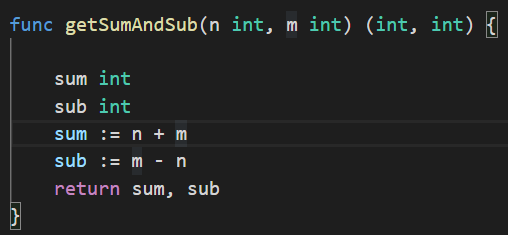
（1）从语言层面支持并发，实现简单

（2）goroutine，轻量级线程，可实现大并发处理，高效利用多核

（3）基于CPS并发模型(Communicating Sequential Process)

**5.吸收了管道通信机制，形成Go语言特有的管道channel，通过管道channel，可以实现不同的goroutine之间的相互通信。**

**6.函数返回多个值**

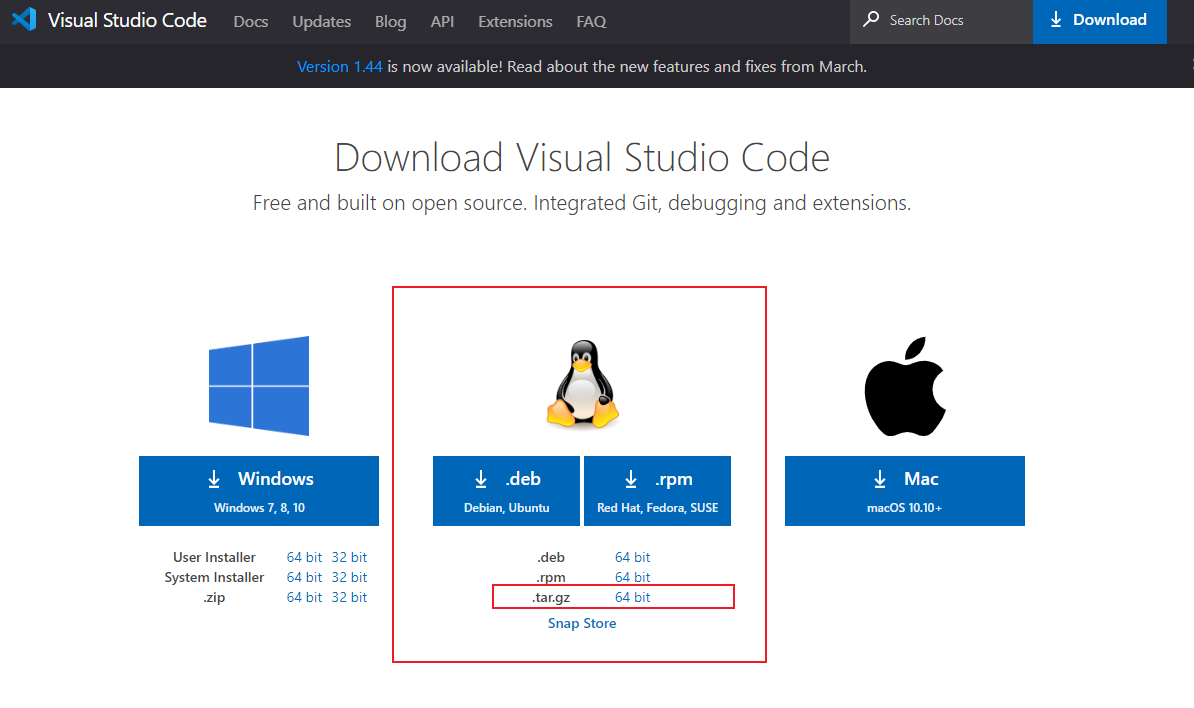


**7.新的创新：比如切片（slice，类似于Java里面的集合）、延时执行defer（关键字）等**

### 简单的Go代码示例：

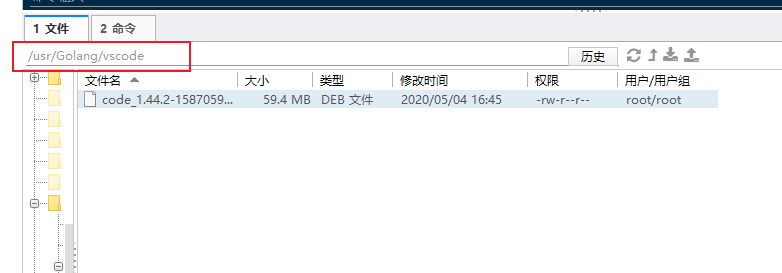
### 在Linux下安装vs code

1.先下载Linux版本的vs code 安装软件。Vscode压缩包



官方链接：[https://code.visualstudio.com/download#](https://code.visualstudio.com/download)

2.将软件传输到Linux操作系统



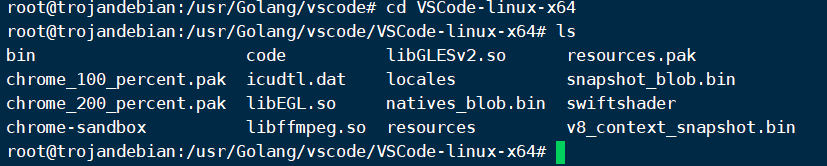
3. 用Linux命令进入Linux系统（debian）vs压缩包的目录，解压缩visual studio





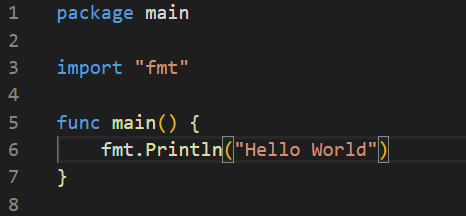
Linux命令：tar -zxvf code-stable-1587060099.tar.gz

4.cd 到解压缩后的目录（VSCode-linux-x64），查看目录内容



5.

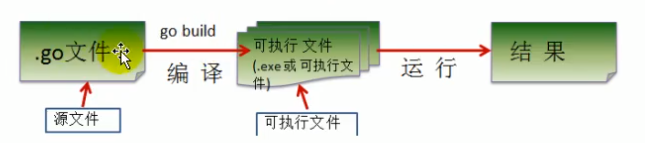
### 使用VSCode编写第一个Go程序实例：



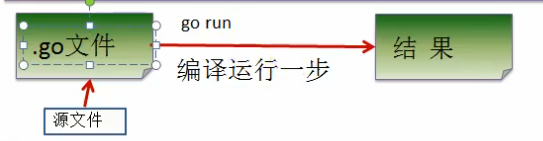
### Golang执行流程分析

如果是对源码编译后，再执行，Go的执行流程如下：

go build 指令：

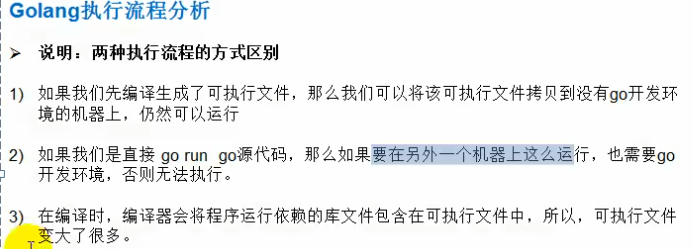


go run 指令：



还是会把源码编译成二进制文件后再执行，只是把这个过程隐藏了起来。

在实际的开发中，还是应该把Golang编译成二进制代码，然后再执行。



指定生成的.exe文件名的指令

eg:go build -o helloWorld.exe

如果编译时，程序有错误，会在错误的那行报错，可以帮助我们快速的定位代码在哪里出错

运行有两种形式：

1.go build test.go

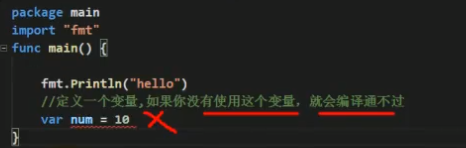
2.go run test.go

### Go程序开发的注意事项



和Java的区别：6）go语言**定义的变量**或者**import的包**如果没有使用到，代码不能编译通过

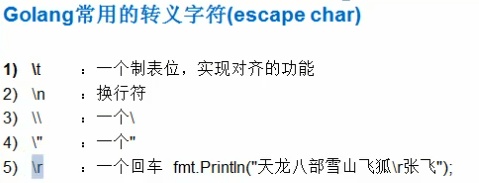
这个特性的好处是。假如B接手了A的一个项目，如果A定义了一些没用的变量或者引用了一些没用的包，B不知道这些包和变量是否有用，就不敢乱动。这样做会导致代码阅读性很差。Go语言的这个强制规定可以让代码变得更加简洁。

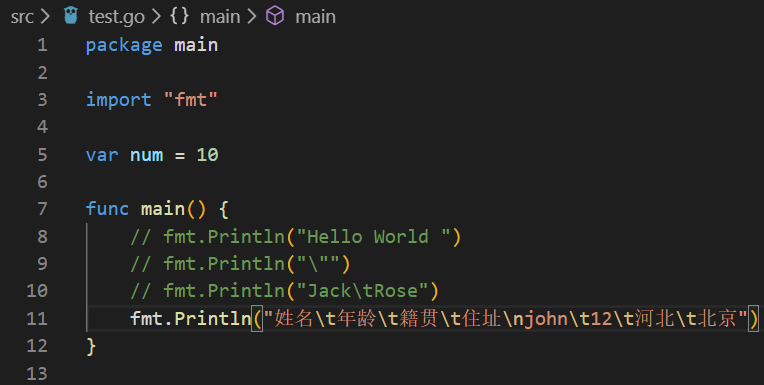
eg：

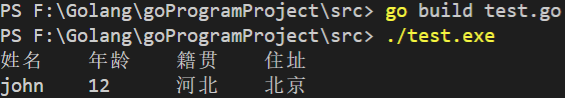
fmt：format 格式化

### Go语言的转义字符（escape char）

常用的转义字符：



eg：

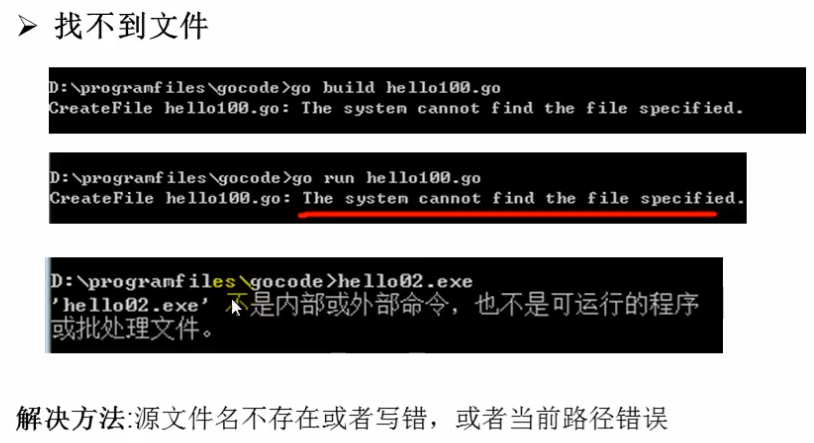


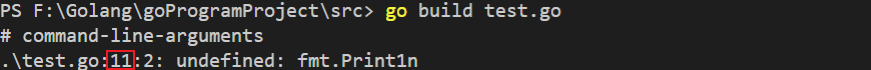
因为Go语言是以package来组织代码的，因此假如在**同一个文件下**，**同一个package**里面有两个函数，就会提示**重复定义**

因此，如果**想使用相同的package名**，则可以新建一个文件夹**。**因为在不同的文件夹里，即使package名字相同，但是也不是同一个package

Go语言里面要求，如果函数是**main函数**，则**必需在main包里面**

### Go初学者开发常见问题和解决方法



11表示代码在11行出现编译错误

### Golang注释

块注释里面不允许有块注释嵌套

### 规范的Golang代码风格

官方推荐用**行注释**来注释整个方法和语句

正确的缩进和空白

1）使用一次tab操作，实现缩进，默认整体向右移动的时候，用**shift+tab**整体向**左移**。

2）或者使用format来进行格式化

3）运算符两边习惯加一个空格。如 1 + 2 = 3

可以在终端使用命令行 gofmt -w xxx.go来进行格式化代码

eg：

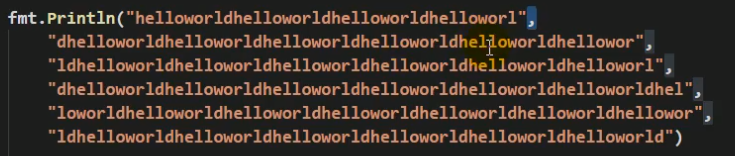
或者，在VisualStudio默认保存的时候，自动格式化代码

### Go设计者：一个问题尽量只有一个解决办法

eg：



一行最长不超过80个字符，尽量保持代码格式优雅



Go语言可以用**逗号**拼接字符串，类似于Java的 + 号

### Golang 官方文档

<https://golang.org/#>

### Golang API

<https://golang.org/pkg/>

eg：比特币中常用的加密算法 sha256



也可以直接搜索某个API(**A**pplication **P**rograming **I**nterface，应用程序编程接口)

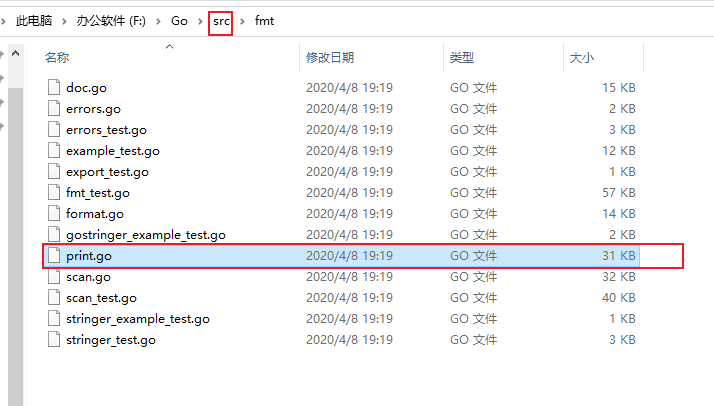
Go的各个包的各个函数



不需要翻墙的Golang API官方文档

<https://studygolang.com/pkgdoc>

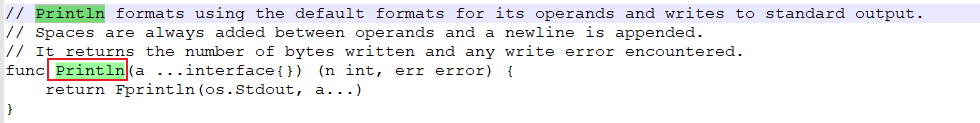
### 查看Go语言函数的源代码



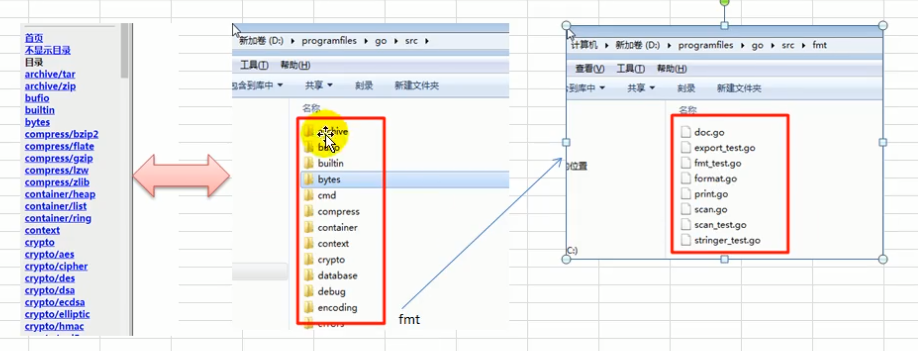
eg：如想看fmt.Println 这个方法的源代码

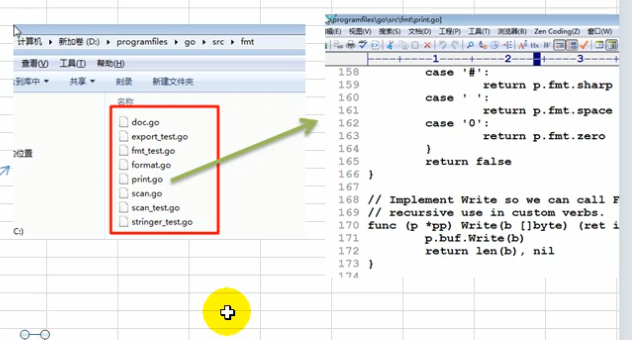
那么，就可以打开print.go

按 ctrl + F 搜索Println



### Golang的包和源文件和函数的关系简图





**包（package）**和下载的Golang语言里面的**src文件夹**相互对应，文件夹里面的**源文件，xxx.go**里面包含了这个包里面的所有**函数**

Golang中，调用一个函数的方式：import 包

eg：

import package “fmt”

使用包的函数

**包名.函数名**

eg:

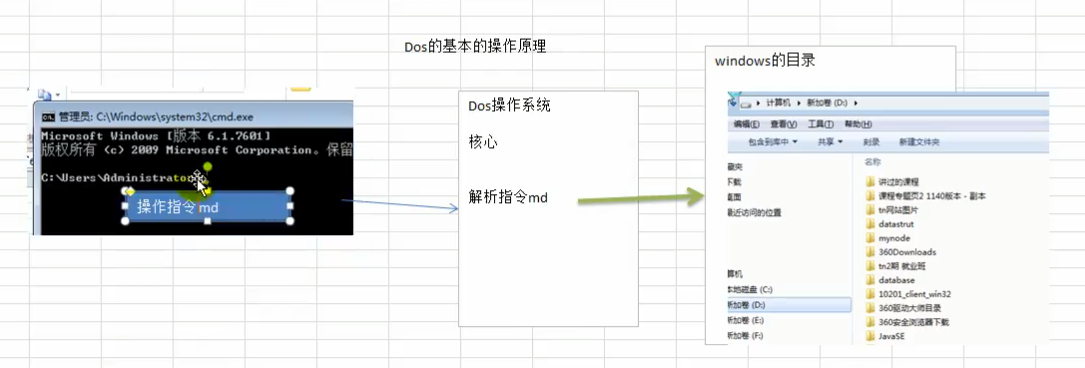
**fmt.Println**

### Dos常用的命令

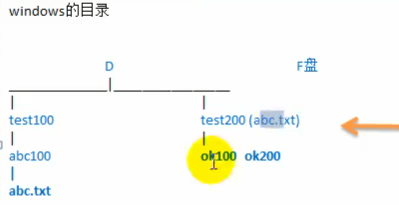
dos的基本介绍：

Disk Operating System 磁盘操作系统

cmd进入的只是Dos的基本终端



通过Dos终端，向Dos操作系统发送指令，然后Dos操作系统解析终端发送过来的指令，从而操作Windows系统。



**dir**：查看目录下有多少文件

**切换到d盘**

d:

绝对路径：从**当前盘的最上面**开始定位，比如D：，找到需要对应的目录

相对路径：从**当前位置**开始定位，去找对应的目录

**回到上级目录**

cd..

**切换到根目录**

cd \

**新建目录**

mk test100 （md make directory 新建目录）

**新建多个目录**

mk test100 test200

**删除空目录**

rd test100 (rd remove directory)

**删除非空目录（不带询问）**

rd /q /s test100 (q:query，不用询问的安静模式。s:层级式的)

**删除非空目录（带询问）**



**文件**的操作

把 hello 写入 abc.txt （绝对路径）



追加到该目录上的文件里 （相对路径）



把文件从某个地方复制到另一个地方



复制并重新命名文件



移动文件（剪切）



删除文件



删除所有文件（后缀为.txt）



清屏：



退出dos：



### Go语言的SDK是什么？

SDK就是Software Development Kit（软件开发工具包）

### Golang环境变量配置及其作用：

GOROOT:指定 go sdk 安装目录。

Path:指定 sdk\bin目录

这个目录里面包含了三个非常重要的程序：

go.exe:编译和运行Go程序

godoc.exe

gofmt.exe 格式化

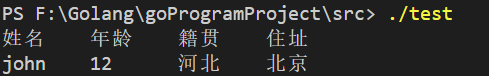
GOPATH：就是**Go项目**的工作目录，所有的项目源码都放在这个目录下面。

### Go程序的编写、编译、运行步骤是什么？能否一步执行？

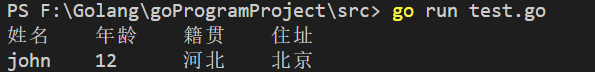
编写：写源代码 编译：go build 源码 -> 生成一个二进制的可执行文件（.exe）

运行：

1.直接对编译后的可执行文件进行运行。 xx.exe 或者 ./可执行文件



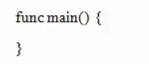
2. go run 源码



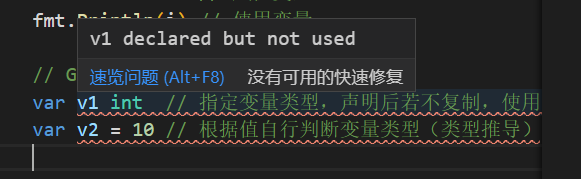
推荐用第一种方式

### Golang程序编写的规则





### 4）eg:



### 简述：在配置环境、编译、运行各个步骤中常见的错误

容易遇到拼写、路径等错误

### Go语言核心编程

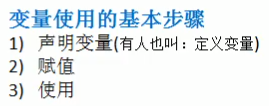
### 变量

变量是程序最基本的组成单位

### 变量介绍

变量相当于**内存**中一个**数据存储空间**的表示

可以把变量当做是一个房间的门牌号，通过门牌号我们可以找到房间，同样的道理，通过变量名可以访问到变量（值）。



### 变量快速入门案例



### Golang变量使用注意事项

1）变量表示**内存中的一个存储区域**

2）该区域有自己的名称（变量名）和类型（数据类型）

3）Golang变量使用的三种方式

1）第一种：指定变量类型，声明后若不复制，使用默认值

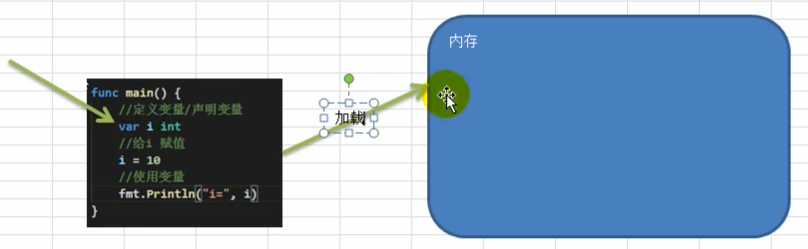
2）根据值自行判断变量类型（类型推导）

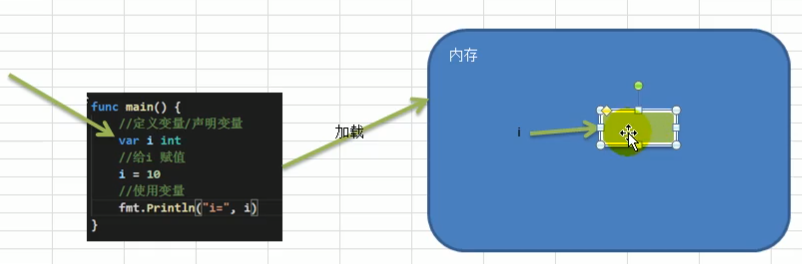
3）省略var，注意：=左侧的变量不应该是已经声明过的，否则会导致编译错误

4）多变量声明。在编程中，有时我们需要一次性声明多个变量，Golang也提供这样的语法

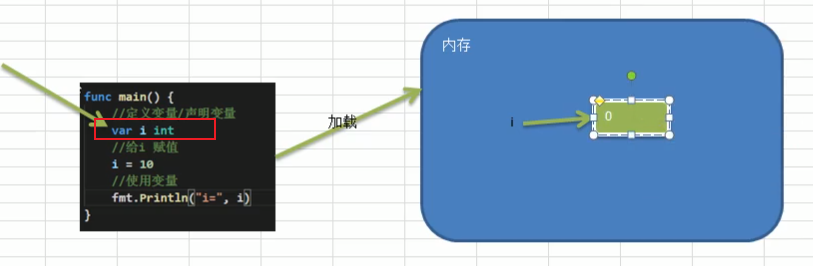
### 变量使用的基本原理

所有的代码要运行起来都必须加载到内存，才能运行。

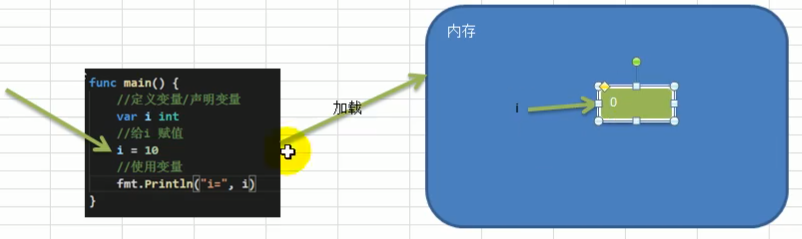




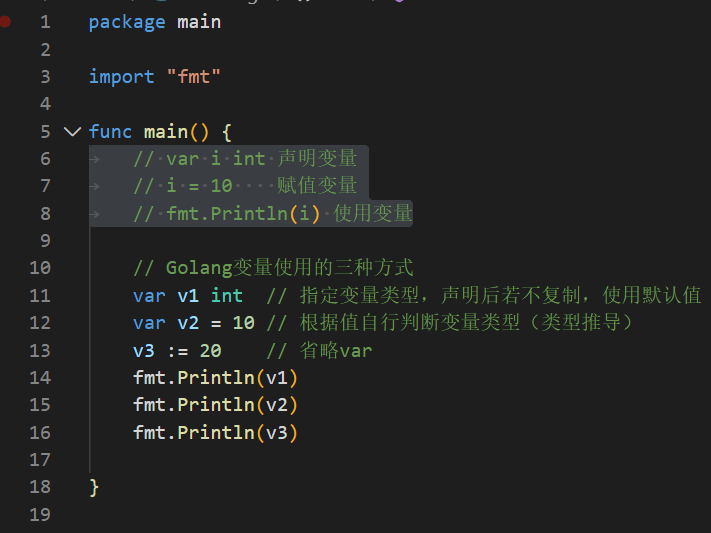
如上图所示。i与内存中的一个数据空间关联了起来



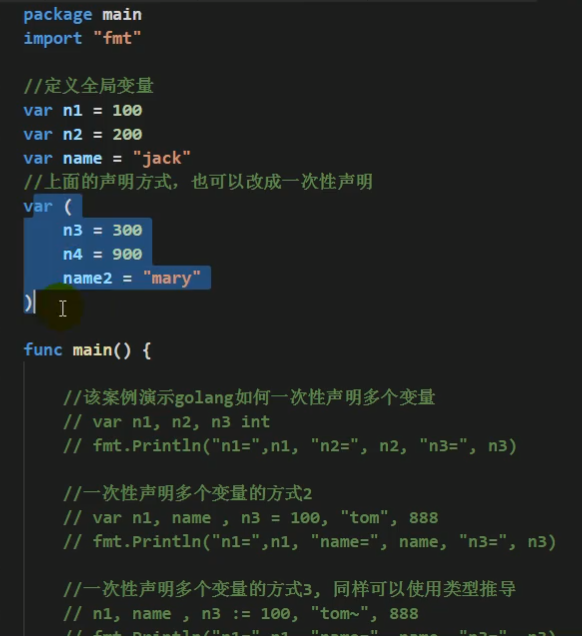
当定义变量i时（var i int），内存中有一个默认的值为0

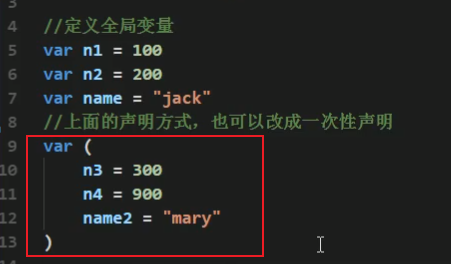


当给i赋值时（i = 10），计算机就会根据i找到内存中与i关联的那个空间，并且把空间里面的值0重新赋值为10



全局变量：在函数外定义的变量





第二种为开发中常见的定义全局变量的方式

### 变量使用的注意事项

变量 = 变量名 + 变量值 + 数据类型

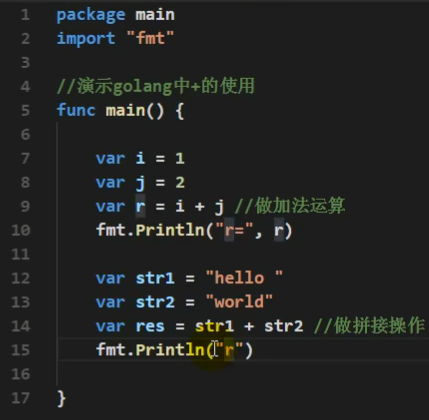
变量在该区域的数据值可以在同一类型范围内不断变化

变量在同一个作用域内不能重名

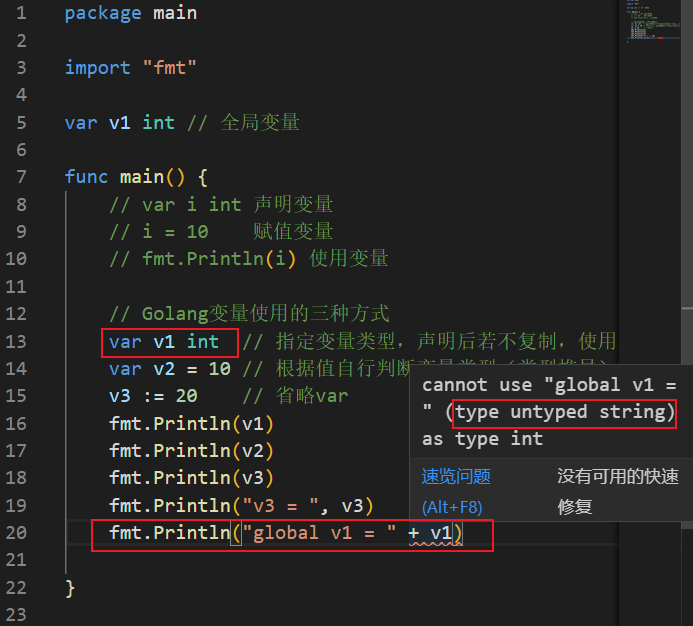
Golang的变量如果没有赋初值，编译器会使用默认值，比如int默认值0，string默认值为空字符串

变量在同一个作用域（在一个函数或者在代码块）内不能重名

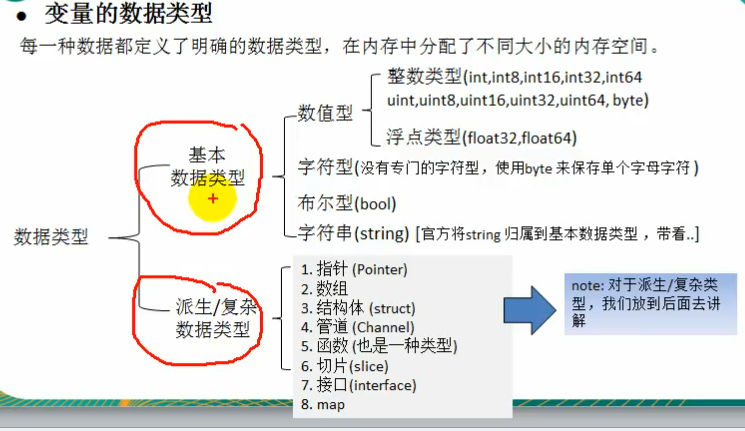
+ 号的使用，如果是字符串，则表示拼接。但是，和Java不同的是，+号不能拼接不同类型的变量 ， 需要使用，来拼接不同类型的变量。



+号不能拼接不同类型的变量，需要用 ，号



### Go的数据类型

int8 8个bit（位） = 1个byte（字节）

int16 16个bit（位） = 2个byte（字节）

int 64个bit（位） = 8个byte（字节）

uint（无符号的整数）

byte = 1个byte（字节）

float32 4个字节，单精度

float64 8个字节，双精度

rune // int32 的别名 rune在存放的时候用Unicode码值来存放，如果有需要存放中文字符的时候，用rune来表示

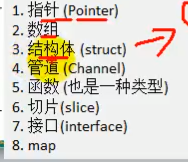
Golang没有专门的字符型，用byte来存储单个**字母（不能是汉字，因为一个汉字需要占据3个字节）**字符

Golang使用UTF-8编码，一个汉字占3个字节。

bool 只能为true或者false

string 被Go语言归结为基本数据类型（Java中把String字符串类型归结为复杂数据类型）

### Go的复杂数据类型



3.struct相当于Java中的类

6.切片，slice。有点类似于动态的数组

8.map。类似于其它编程语言里面的集合

### 整数类型的取值范围

### 有符号的整数





第一位用来表示符号位

0 代表正数

1 代表负数

后面的才表示数的大小



如果全部都取0，二进制数就会变成00000000就会出现一个正0

如果变成10000000，那么就表示负0

11111111表示 -（2的7次方-1）= -127

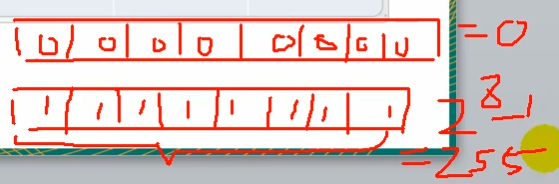
负0没有任何实际的意义，因此计算机利用了这个负0，用它来表示最小的值-（2的七次方） = -128

因此在计算机中10000000 = -128

无符号的整数



只有整数，因为第一位没有符号，代表一个具体的数

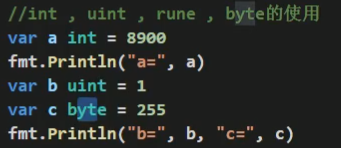


应该根据数据的范围，选取合适的数据类型

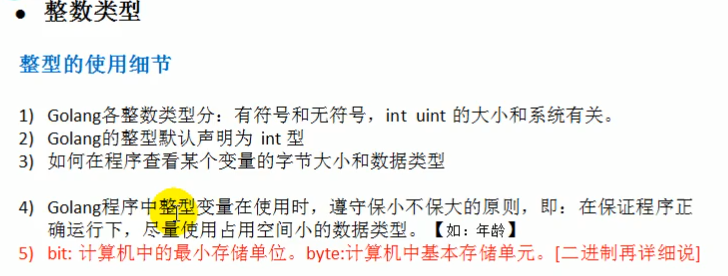
### 整数的类型

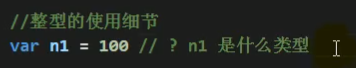


当要存储字符时，一般使用 byte 来存储



### 整型的使用细节





n1是int类型，因为整型在没有明确说明的情况下默认定义为int类型

### 如何查看某个变量的数据类型？

fmt.printf() 可以用于格式化输出

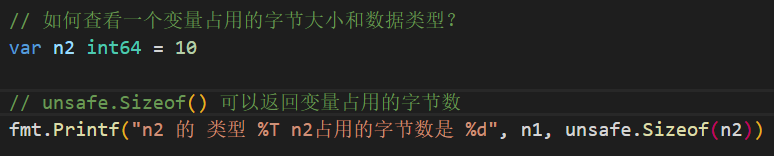


output:



### 如何查看某个变量占用的字节大小和数据类型？

Golang开发中，程序员经常使用这些方法来查看变量的数据类型和大小





Go语言一般使用这种方式引入多个包

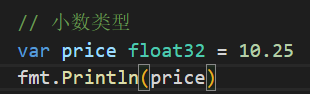
**bit**：计算机中最小的存储**单位**

**byte：**计算机当中最小的存储**单元**

**1byte = 8bit**

### Go语言的浮点类型

小数类型/浮点型 用于存放小数的类型



浮点类型的分类：

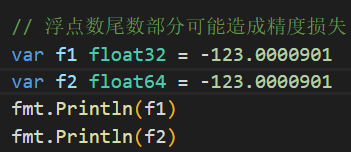


浮点数在机器中存放的简单说明：

浮点数 = 符号位 + 指数位 + 尾数位

说明：浮点数都是有符号位

浮点数的尾数部分可能丢失，造成**精度损失**





Golang的浮点类型默认为 float64

### 用科学计数法形式表示浮点类型

### 

e换成大写的E也可以

开发中，**推荐**使用**float64**

### **Golang的字符类型**

Golang中没有专门的字符类型，如果要存储单个字节的字符（字母），一般使用byte来保存。

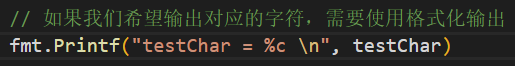
字符串就是一串固定长度的字符连接起来的字符序列。Go的字符串是由单个字节连接起来的。也就是说对于传统的字符串是由字符组成的，而**Go的字符串不同，它是由字节组成的。**

当我们直接输出**字符**时

输出的是字符对应的 **ASCII 码值（十进制）**

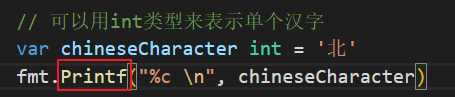
因为在计算机里，每个字符都有对应一个数字

如果我们希望输出对应的字符，需要使用**格式化输出**

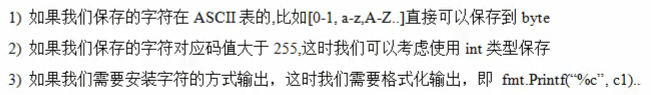




可以用int类型来表示单个汉字



注意：使用格式化输出必须要调用fmt里面的Printf() 函数

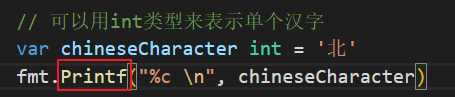


Go语言使用UTF-8对字符进行编码

UTF-8 就是Unicode 的一种具体实现

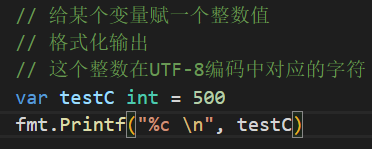
字符的本质就是**整数**，直接输出时，是该字符对应的UTF-8编码

可以直接给某个变量赋一个数字，然后按格式化输出时%c，会输出该对应的unicode字符



字符类型是可以进行运算的，相当于一个整数，因为它都对应有Unicode编码

我们也可以直接给某一个变量赋一个整数值，然后按格式化输出时%c，会输出该数字对应的unicode字符



output:



### 字符的编码

UTF-8编码是对ASCII表的一种升级

是计算机编程中用得最多的一种编码

UTF-8是兼容ASCII

在UTF-8中，英文字母占据1个字节

汉字占据3个字节

### 字符类型本质探讨

1）字符型存储到计算机中，需要将字符对应的UTF-8码值（整数）找出来

存储：字符 -> 对应的UTF-8码值 -> 二进制数 -> 存储

读取：二进制 -> 对应的UTF-8码值 -> 码值对应的字符 -> 读取

2）字符和码值的对应关系是通过字符编码表决定的（是规定好的）

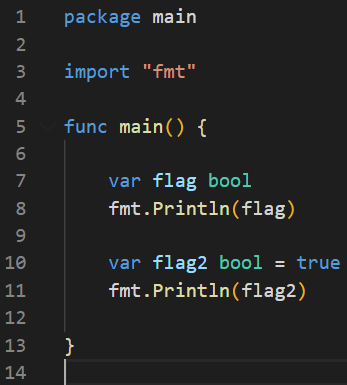
3）Go语言的编码都统一成了UTF-8。非常的方便，统一，再也没有编码的困扰了

### Golang的bool类型的使用

1）bool类型数据只允许取值false或者true

2）bool类型占据一个byte的存储空间

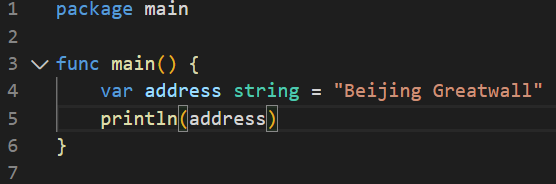
3）bool类型适合用于逻辑运算，一般用于流程控制

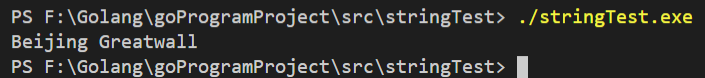


### Golang字符串类型：string

基本介绍：

字符串就是一串固定长度的**字符，**连接起来的**字符序列**。Go的字符串是由单个字节连接起来的，Go语言的字符串的字节使用**UTF-8编码**标识Unicode文本

output:



### Golang字符串使用细节

1）Go语言的字符串的字节使用UTF-8编码标识Unicode文本，这样Golang统一使用UTF-8编码，**中文乱码问题**就不会再困扰程序员了