安装Go语言

- Go源码安装：参考链接
- Go标准包安装：下载地址
- 第三方工具安装

```
C:\Users\Unknown>go env
set GOARCH=amd64
set GOBIN=
set GOCHAR=6
set GOEXE=.exe
set GOGCCFLAGS=-g -O2 -m64 -mthreads
set GOHOSTARCH=amd64
set GOHOSTOS=windows
set GOOS=windows
set GOPATH=E:\Go\Development
set GOROOT=D:\Go
set GOTOOLDIR=D:\Go\pkg\tool\windows_amd64
set CGO_ENABLED=1
```

Go环境变量与工作目录

根据约定，GOPATH下需要建立3个目录：
- bin（存放编译后生成的可执行文件）
- pkg（存放编译后生成的包文件）
- src（存放项目源码）

Go命令

在命令行或终端输入go即可查看所有支持的命令

Go常用命令简介

- go get：获取远程包（需 **提前安装** git或hg）
- go run：直接运行程序
- go build：测试编译，检查是否有编译错误
- go fmt：格式化源码（部分IDE在保存时自动调用）
- go install：编译包文件并编译整个程序
- go test：运行测试文件
- go doc：查看文档（CHM手册）

程序的整体结构

```
bin/
    mathapp
pkg/
    平台名/ 如: darwin_amd64、linux_amd64
         mymath.a
         github.com/
              astaxie/
                   beedb.a
src/
    mathapp
          main.go
      mymath/
          sqrt.go
      github.com/
           astaxie/
                beedb/
                    beedb.go
                    util.go
```

Go内置关键字（25个均为小写）

break	default	func	interface	select
case	defer	go	map	struct
chan	else	goto	package	switch
const	fallthrough	if	range	type
continue	for	import	return	var

Go注释方法

- // ：单行注释
- /* */：多行注释

Go程序的一般结构：**basic_structure.go**

- Go程序是通过 **package** 来组织的（与python类似）
- 只有 **package** 名称为 **main** 的包可以包含 **main** 函数
- 一个可执行程序 **有且仅有** 一个 main 包

- 通过 **import** 关键字来导入其它非 **main** 包
- 通过 **const** 关键字来进行常量的定义
- 通过在函数体外部使用 **var** 关键字来进行全局变量的声明与赋值
- 通过 **type** 关键字来进行结构(**struct**)或接口(**interface**)的声明
- 通过 **func** 关键字来进行函数的声明

```
// 当前程序的包名
package main

// 导入其它的包
import "fmt"
import (
    "fmt"
    "log"
    "os"
    "time"
    "strings"
)

// package别名
import print "fmt"
// 省略调用
import  .(点)  "fmt"
// 不建议使用，容易混淆，不可以和别名同时使用
// 常量的定义
const PI = 3.14

// 全局变量的声明与赋值
var name = "jiemin"

// 一般类型声明
type newType int
```

// 结构的声明
type gopher struct{}

// 接口的声明
type golang interface{}

// 由 main 函数作为程序入口点启动
func main(){
	fmt.Println("Hello world!你好，世界！")
}

```
1.
2.  // package别名
3.  import print "fmt"
4.  // 省略调用
5.  import  .(点)  "fmt"
6.  // 不建议使用，容易混淆，不可以和别名同时使用
7.  // 常量的定义
8.  const PI = 3.14
9.
10. // 全局变量的声明与赋值
11. var name = "jiemin"
12.
13. // 一般类型声明
14. type newType int
15.
16. // 结构的声明
17. type gopher struct{}
18.
19. // 接口的声明
20. type golang interface{}
21.
22. // 由 main 函数作为程序入口点启动
23. func main(){
24.     fmt.Println("Hello world!你好，世界！")
25. }
```

Go导入 package 的格式

```
import "fmt"
import "os"
import "io"
import "time"
import "strings"
```

```
import (
    "fmt"
    "io"
    "os"
    "strings"
    "time"
)
```

- 导入包之后，就可以使用格式<PackageName>.<FuncName>来对包中的函数进行调用
- 如果导入包之后 **未调用** 其中的函数或者类型将会报出编译错误：

```
imported and not used: "io"
```

package 别名

- 当使用第三方包时，包名可能会非常接近或者相同，此时就可以使用别名来进行区别和调用

```
import (
    io "fmt"
)
```

```
// 使用别名调用包
io.Println("Hello world!")
```

省略调用

- **不建议使用**，易混淆
- 不可以和别名同时使用

```
import (
    . "fmt"
)

func main() {
    // 使用省略调用
    Println("Hello world!")
}
```

可见性规则

- Go语言中，使用 **大小写** 来决定该 常量、变量、类型、接口、结构或函数 是否可以被外部包所调用：

根据约定，函数名首字母 **小写** 即为private

```
func getField(v reflect.Value, i int) refle
    val := v.Field(i)
    if val.Kind() == reflect.Interface && !
        val = val.Elem()
```

函数名首字母 **大写** 即为public

```
func Printf(format string, a ...interface{}) (n int
    return Fprintf(os.Stdout, format, a...)
}
```

课堂作业

既然导入多个包时可以进行简写，那么声明多个 常量、全局变量或一般类型（非接口、非结构）是否也可以用同样的方法呢？

请动手验证。

```
// 常量的定义
const (
    PI     = 3.14
    const1 = "1"
    const2 = 2
    const3 = 3
)
```

```
// 全局变量的声明与赋值
var (
    name  = "gopher"
    name1 = "1"
    name2 = 2
    name3 = 3
)
```

```
// 一般类型声明
type (
    newType int
    type1   float32
    type2   string
    type3   byte
)
```

作业：

```
1.  package main
2.
3.  import (
4.      "fmt"
5.  )
6.
7.  const (
8.      PI = 3.14
9.      const1 = "1"
10.     const2 = 2
11.     const3 = 3
12. )
```

```
13.
14. var (
15.     name = "gopher"
16.     name1 = "1"
17.     name2 = 2
18.     name3 = 3
19. )
20.
21. type (
22.     newType int
23.     type1 float32
24.     type2 string
25.     type3 byte
26. )
27.
```