一，基本语法

一）变量定义

- var定义

```
// 会自动初始化
var a int // default 0
var b string // default ""
var c, d int = 3, 4
// go会自动判断变量类型
var c, d, e, f = 3, 4, true, "hello"
```

- 简单写法

```
a := 1
b := "hello world"
c, d , e, f := 3, 4, true, "hello"
```

注：在方法体外则必须使用var定义

```
package main

var (
  a = 1
  b = true
  c = "hello"
)

func main() {
  // to do something
}
```

二）内建变量类型

- boole, string
- (u)int, (u)int8, (u)int16, (u)int32, (u)int64 , uintptr指针 // 加u表示无符号整数，不加u表示有符号整数
- byte（8位）, rune （字符型，相当于char, 32位）
- float32, float64, complex64（实部虚部都是float32）, complex128（实部虚部都是float64）（复数类型，实位，虚位）

```
i = √-1
//复数
3 + 4i // 3实部,  4i虚部
|3 + 4i| = √3^2 + 4^2 = 5
i^2 = -1, i^3 = -i,i^4 = 1,...
```

```
func euler() {
  fmt.Println(
    cmplx.Exp(1i*math.Pi) + 1
  )
}
// 0+1.2246467991473515e-16i
```

三）常量与枚举

- 常量

```
const a = "hello"
const b, c = 3, 4
const (
  filename = "reboot.ini" // 常量名不使用大写，go中大写具有特殊意义
  d, e = 1, 2
)
```

- 枚举

```
// ioat自动自增
const (
  cpp = iota // 0
  java // 1
  python // 2
  php // 3
  golang // 4
  javascript // 5
)

const (
  b = 1 << (10 * iota)
  kb
  mb
  gb
  tb
  pb
)
```

四）条件语句

- if else

```go
func main() {
  const filename = "reboot.txt"
  /*content, err := ioutil.ReadFile(filename)
  if err != nil {
    fmt.Println(err)
  } else {
    fmt.Printf("%s\n", content)
  }*/
  // 简洁写法
  if content, err := ioutil.ReadFile(filename); err != nil {
    fmt.Println(err)
  } else {
    fmt.Printf("%s\n", content)
  }
}
```

- switch

```go
func eval(a, b int, op string) int {
  var result int
  switch op {
  case "+":
    return a + b
  case "-":
    return a - b
  case "*":
    return a * b
  case "/":
    return a / b
  default:
    panic("unsupported operator:" + op)
  }
  return result
}

func grade(score int) string {
  g := ""
  switch {
  case score < 0 || score > 100:
    panic(fmt.Sprintf("err score input"))
  case score < 60:
    g = "F"
  case score < 80:
    g = "C"
  case score < 90:
    g = "B"
  case score <= 100:
    g = "A"
```

```
  }
  return g
}
```

五）循环语句

- for

```
func printFile(filename string) {
  file, err := os.Open(filename)
  if err != nil {
    panic(err)
  }
  scanner := bufio.NewScanner(file)
  for scanner.Scan() {
    fmt.Println(scanner.Text())
  }
}
// 死循环
for {
  // TODO
}
```

六）函数

```
// 函数返回多值时可以起名
// 仅适用于简单的函数
// func 函数名(参数) 返回值/类型 {}
func eval(a, b int, op string) int {
}
// 返回多值
func div(a, b int) (int, int) {
}
// 接收返回值
func main() {
  a := eval(1, 2)
  c, d := div(3, 4)
  // 多值只接收一个
  _, f := div(4, 5)
}

// 函数作为参数
func apply(op func(int, int) int, a, b int) {
  p := reflect.ValueOf(op).Pointer()
  opName := runtime.FuncForPC(p).Name()
  fmt.Printf("Calling function %s with args "+"(%d, %d)", opName, a, b)
}
func main() {
  apply(func(a int, b int) int {
```

```
    return a + b
  }, 3, 4)
}
```

七）*指针

```
var a int = 2
var pa *int = &a
*pa = 3
fmt.Println(a)
```

- 指针不能运算
- Go只有值传递一种方式

```
func swap(a, b *int) {
  *b, *a = *a, *b
}

func main() {
  a, b = 3, 4
  swap(&a, &b)
  fmt.Println(a, b)
}
```