变量与常量

1.变量

> 虽然Go语言与python变量定义类似，但是其变量一定是强类型的，只是在创建时自由，之后类型是不能改变的。

1.1显式指定变量类型

> 在变量名后指定类型，可以初始化。不初始化的情况下为默认零值。
>
> 零值：
>
> int类：0
>
> string类：""
>
> bool类：false

```
var v_name v_type
```

例如：

```
var b int
// bool 零值为 false
var c bool
```

1.2隐式指定变量类型

> Go语言会根据初始化的值自动判断变量类型，之后变量类型固定

```
var v_name = value
```

例如：

```
var d = true
var h = "hello"
```

2.短变量

> 属于特殊的变量

2.1声明方式

```
v_name := value
```

2.2局部作用域

> 短变量的只能作用在局部作用域，例如函数内部，if，switch等

2.3允许重复声明

> 使用var不能重复声明变量，但是使用短变量可以在如下条件下重复声明

```
a := "hello"
a,c := "hello","n"      //因为:=左边有新变量，所以可以重复声明a
fmt.Println(a)
fmt.Println(c)
```

> 这样会报错

```
a := "hello"
a := "hello"        //报错：no new variables on left side of :=
fmt.Println(a)
```

注意

- 被重复声明的变量并没有创建多次， 只是将新的值赋予原来的变量
- 只有在同一作用域才是重复声明，如果在不同作用域声明相同变量名，则是创建新的变量

3.常量

> 常量是不能被修改的量，在Go中只能是布尔型、数字型（整数型、浮点型和复数）和字符串型。

3.1创建

> 类型可以省略，Go语言会根据值自动推断

```
const identifier [type] = value
```

例如：

```
const b string = "abc"
//或者
const b = "abc"
```

多个变量可以简写为：

```
const c_name1, c_name2 = value1, value2
//例如
const a,b,c = "a","b","c"
```

3.2枚举

> 常量可以用作枚举类型

```
const(
    Blue = 0
    Yellow = 1
    Green = 2
)
```

> 枚举可以省略赋值，表示与上面的常量赋值一致

```
const(
    Blue = 0    //0
    Yellow      //0
    Green       //0
)
```

3.3使用内置函数初始化

> 常量可以用函数，但必须是内置函数进行初始化。
>
> 内置函数例如：len(), cap(), unsafe.Sizeof()

```
package main

import "unsafe"
const (
    a = "abc"
    b = len(a)
    c = unsafe.Sizeof(a)
)

func main(){
    println(a, b, c)
}
```

3.4特殊常量iota

> 特殊常量，可以认为是一个可以被编译器修改的常量。
>
> iota 在 const关键字出现时将被重置为 0(const 内部的第一行之前)，const 中每新增一行常量声明将使 iota 计数一次(iota 可理解为 const 语句块中的行索引)。

一般用于在const枚举中创建连续或者有规律的枚举值

```
const (
    a = iota        //0
    b = iota        //1
    c = iota        //2
)

const (
    d = iota        //再次被重置为0
    e = iota        //1
    f = iota        //2
)
```

与枚举的省略赋值特性结合：

```
package main

import "fmt"

func main() {
    const (
            a = iota   //0
            b          //1
```

```
            c           //2
            d = "ha"    //独立值，iota += 1
            e           //"ha"   iota += 1
            f = 100     //iota +=1
            g           //100  iota +=1
            h = iota    //7,恢复计数
            i           //8
    )
    fmt.Println(a,b,c,d,e,f,g,h,i)
}

//输出：0 1 2 ha ha 100 100 7 8
```