



# 迭代类问题求解

---



张华

64174234@qq.com

# 内容

---

1.1 什么是迭代类问题

1.2 利用迭代法计算 $1+2+\dots+100=?$

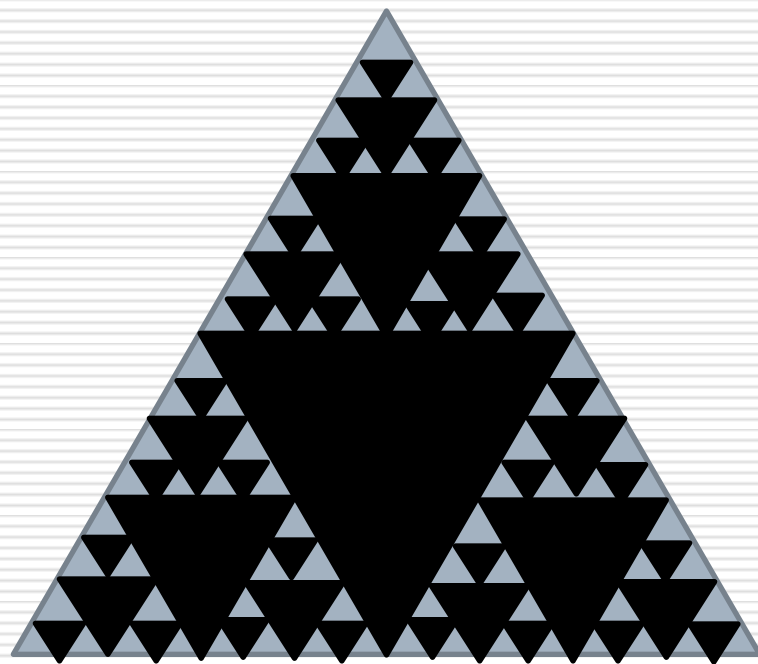
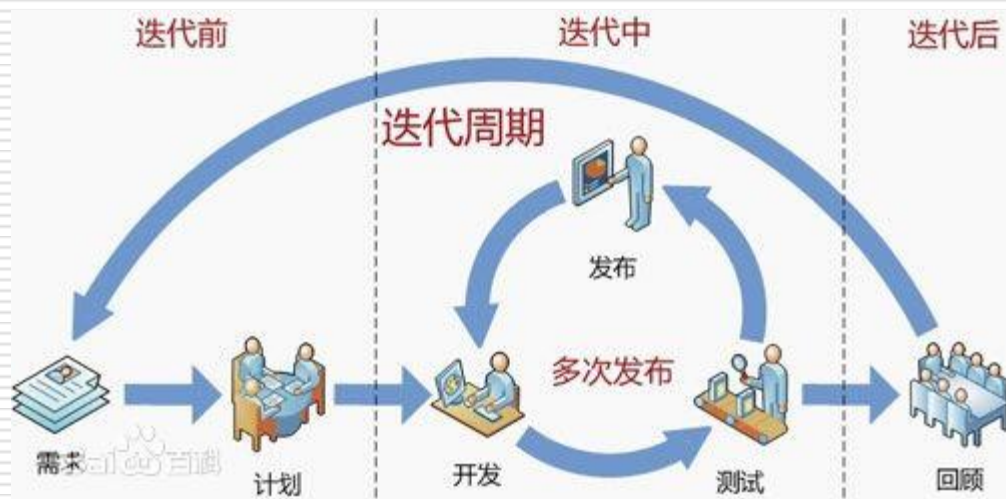
1.3 总结

1.4 思考

# 1.1什么是迭代类问题

## □ 迭代类问题的含义

- 迭代(iteration)类问题:重复反馈过程的活动,其目的通常是为了逼近所需目标或结果。每一次对过程的重复称为一次“迭代”,而每一次迭代得到的结果会作为下一次迭代的初始值。



RUP (Rational Unified Process, 统一软件开发过程) 推荐的迭代模型

分形迭代: 谢尔宾斯基三角形

# 1.1什么是迭代类问题(续)

---

## □ 求解迭代类问题需要做的工作

- 确定迭代变量：至少存在一个直接或间接地不断由旧值推出新值的变量
- 确定关系/抽象模型：指如何从变量的前一个值推出其下一个值的公式，迭代关系的建立是解决迭代类问题的关键，通常可使用递推或倒推的方式完成
- 过程控制：如果迭代次数是个确定的值，可以用循环来实现，如果迭代次数无法确定，需要进一步分析出用来结束迭代过程的条件。

## □ 计算机运算速度快、适合做重复性操作，既适合求解迭代类问题。

# 1.2利用迭代法计算 $1+2+\dots+100=?$

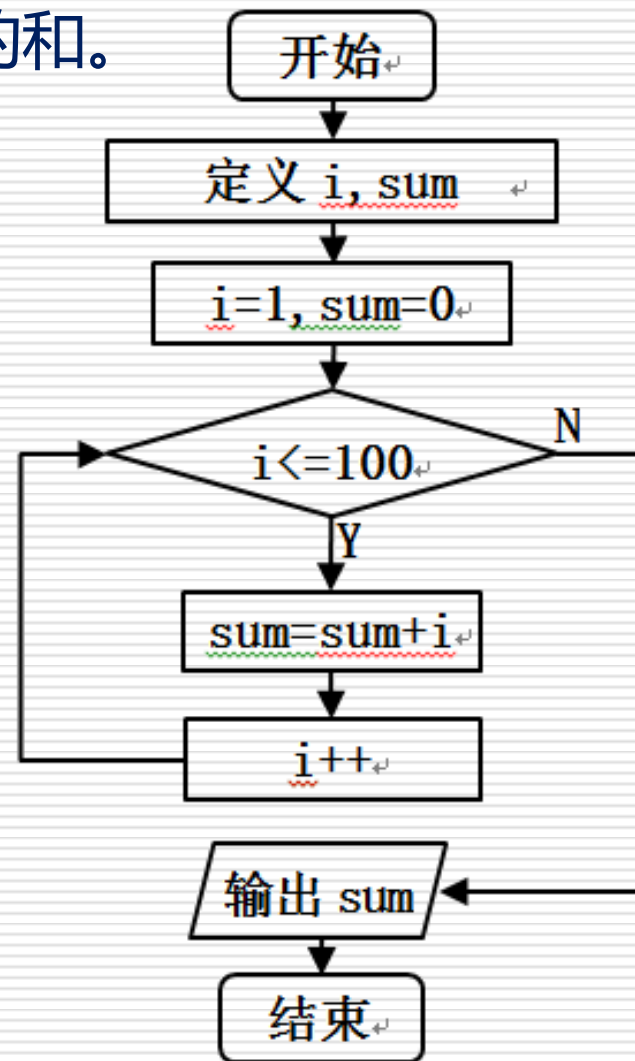
## □ 迭代变量sum

- 总共需要求和100次，每一次求和的结果sum都是前一次的结果和当前的次数i的和。
- 最开始的时候 $sum=0$

## □ 数学模型

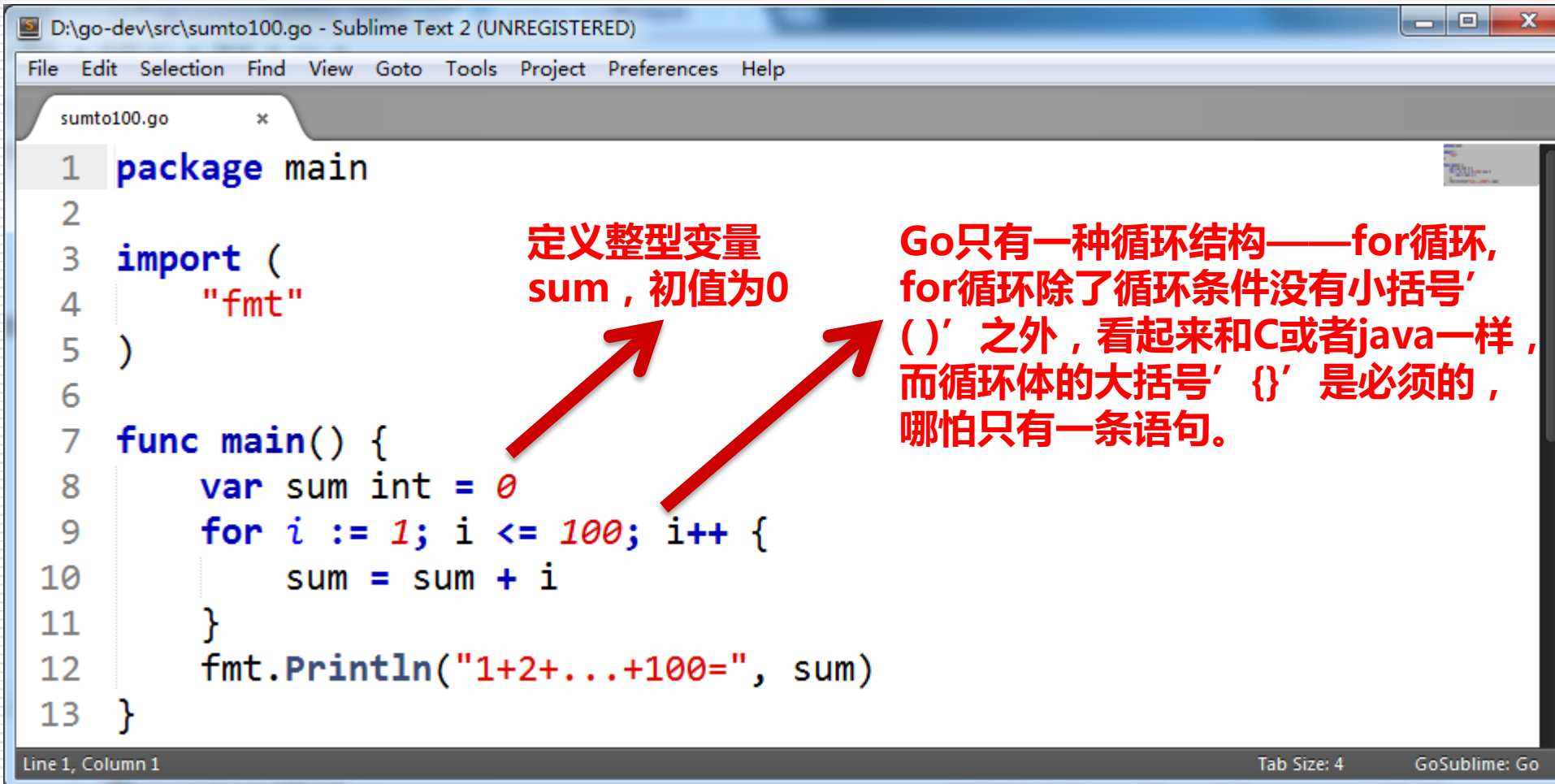
- $sum_i = sum_{i-1} + i$
- $i$  from 1 to 100,  $sum_0 = 0$

## □ 迭代流程/计算流程



# 1.2利用迭代法计算 $1+2+\dots+100=?$ (续)

## □ 实现——方式1



```
D:\go-dev\src\sumto100.go - Sublime Text 2 (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help

sumto100.go
1 package main
2
3 import (
4     "fmt"
5 )
6
7 func main() {
8     var sum int = 0
9     for i := 1; i <= 100; i++ {
10         sum = sum + i
11     }
12     fmt.Println("1+2+...+100=", sum)
13 }
```

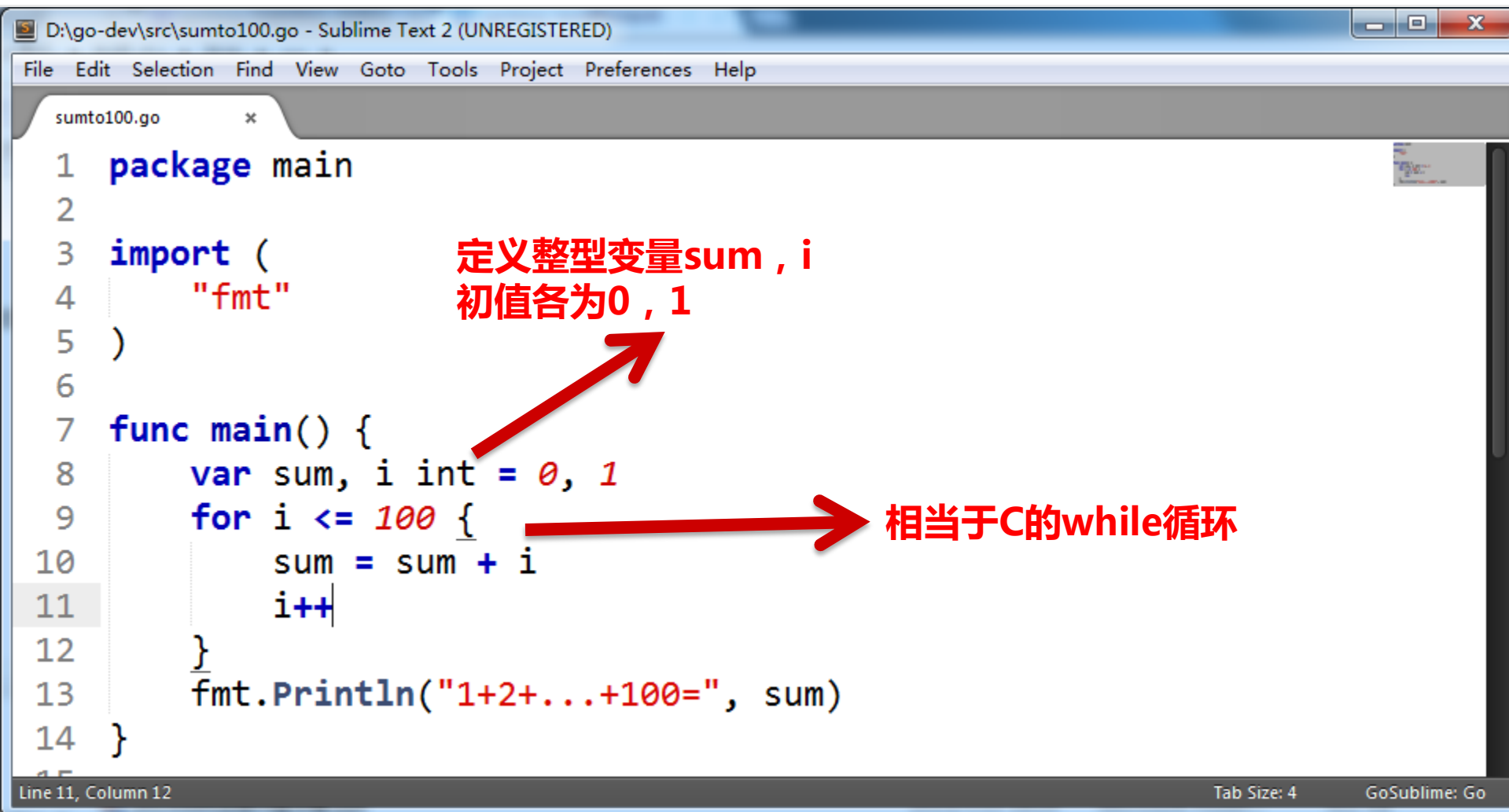
定义整型变量  
sum，初值为0

Go只有一种循环结构——for循环，  
for循环除了循环条件没有小括号'  
( )' 之外，看起来和C或者java一样，  
而循环体的大括号'{'}' 是必须的，  
哪怕只有一条语句。

Line 1, Column 1 Tab Size: 4 GoSublime: Go

# 1.2利用迭代法计算 $1+2+\dots+100=?$ (续)

## □ 实现——方式2



```
D:\go-dev\src\sumto100.go - Sublime Text 2 (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help

sumto100.go
1 package main
2
3 import (
4     "fmt"
5 )
6
7 func main() {
8     var sum, i int = 0, 1
9     for i <= 100 {
10         sum = sum + i
11         i++
12     }
13     fmt.Println("1+2+...+100=", sum)
14 }
```

定义整型变量sum, i  
初值各为0, 1

相当于C的while循环

Line 11, Column 12 Tab Size: 4 GoSublime: Go

# 1.3 总结

---

- 迭代的含义(理解)
- 迭代类问题的求解技术(掌握)
- for循环结构(掌握)
  - for-range结构
    - 一般形式for ix, val := range coll {  
...  
}
    - 可以迭代任何一个集合，包括数组和map
  - for init; condition; post { } // 计数循环
  - for condition { } // 相当于C的while 循环
  - for { } // 死循环，相当于C的for {;;}



# 1.3 总结(续)

---

## □ 变量

- `var sum, i int = 0, 1` 用var关键字可定义一个变量的列表，类型在后面，
- 还可以为各个变量赋初值。如果初始化是表达式，则可以省略类型，变量从初始值中获得类型。

## □ 短声明变量

- `i := 1` 在函数中:=用在明确类型的地方，可以用于替代var来定义变量，但在函数外不能使用:=来定义并初始化变量

# 1.3 总结(续)

---

## □ Go的基本类型有

### ■ bool

□ 布尔类型

### ■ string

□ 字符串类型

### ■ int int8 int16 int32 int64

□ 有符号整数类型,

### ■ uint uint8 uint16 uint32 uint64 uintptr

□ 无符号整数类型

### ■ byte // uint8 的别名

### ■ rune // int32 的别名 , 代表一个Unicode码

### ■ float32 float64

□ 实数类型

### ■ complex64 complex128

□ 复数类型

# 1.3 总结(续)

## □ 具有不同类型的变量演示

The image shows a Sublime Text editor window with a Go file named `basictypes.go`. The code defines several variables of different types and a `main` function that prints them. A red arrow points from the `var` block to the explanatory text on the right. Another red arrow points from the `const f` line to the terminal window.

```
1 package main
2
3 import (
4     "fmt"
5     "math/cmplx"
6 )
7
8 var (
9     ToBe    bool    = false
10    MaxInt   uint64   = 1<<64 - 1
11    z        complex128 = cmplx.Sqrt(-5 + 12i)
12 )
13
14 func main() {
15     const f = "%T(%v)\n"
16     fmt.Printf(f, ToBe, ToBe)
17     fmt.Printf(f, MaxInt, MaxInt)
18     fmt.Printf(f, z, z)
19 }
```

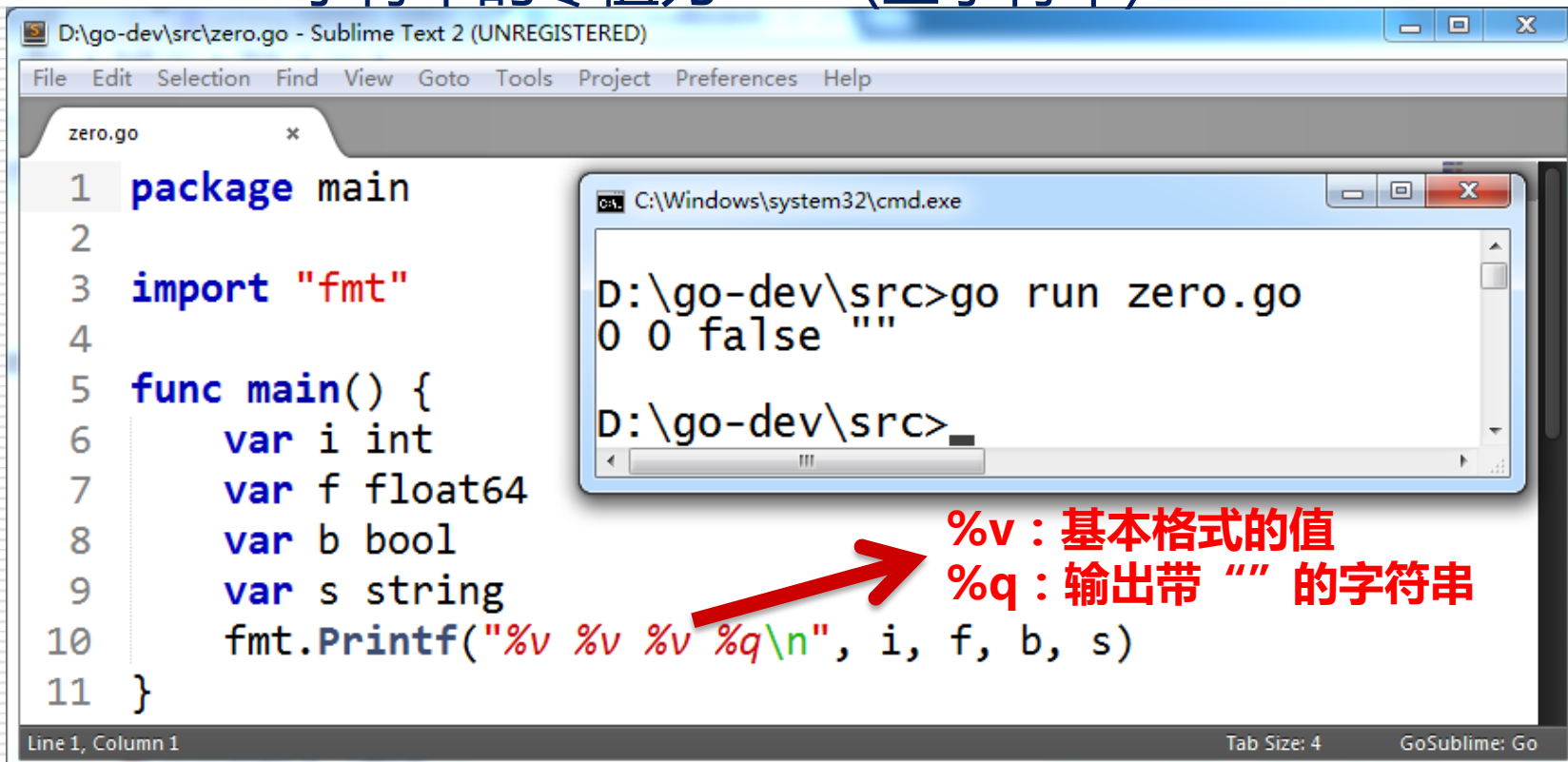
用var()的方式统一定义了一系列不同类型的变量  
布尔变量ToBe，初值为false  
无符号64位整数变量MaxInt，初值为1左移64位再减1  
即2的64次幂-1，得到类型uint64的最大值  
复数变量z，初值为-5+12i的平方根  
const是常量定义关键字  
常量f用来定义下面Printf的输出格式控制  
%T表示输出变量的类型，%v表示输出变量的值

```
C:\Windows\system32\cmd.exe
D:\go-dev\src>go run basictypes.go
bool(false)
uint64(18446744073709551615)
complex128((2+3i))
D:\go-dev\src>
```

Line 1, Column 1 Tab Size: 4 GoSublime: Go

# 1.3 总结(续)

- ❑ 零值：变量在定义时没有明确的初始化时会赋为零值
  - 数值类型的零值为 0
  - 布尔类型的零值为 false
  - 字符串的零值为 "" (空字符串)



The screenshot shows a Sublime Text 2 window with a file named `zero.go` open. The code in the editor is as follows:

```
1 package main
2
3 import "fmt"
4
5 func main() {
6     var i int
7     var f float64
8     var b bool
9     var s string
10    fmt.Printf("%v %v %v %q\n", i, f, b, s)
11 }
```

Overlaid on the editor is a Windows command prompt window titled `C:\Windows\system32\cmd.exe`. It shows the execution of the Go program:

```
D:\go-dev\src>go run zero.go
0 0 false ""
D:\go-dev\src>
```

A red arrow points from the `%q` format specifier in the Go code to the output `""` in the command prompt. To the right of the arrow, there is red text explaining the format specifiers:

- `%v` : 基本格式的值
- `%q` : 输出带 `""` 的字符串

The status bar at the bottom of the Sublime Text window indicates "Line 1, Column 1", "Tab Size: 4", and "GoSublime: Go".

# 1.3 总结(续)

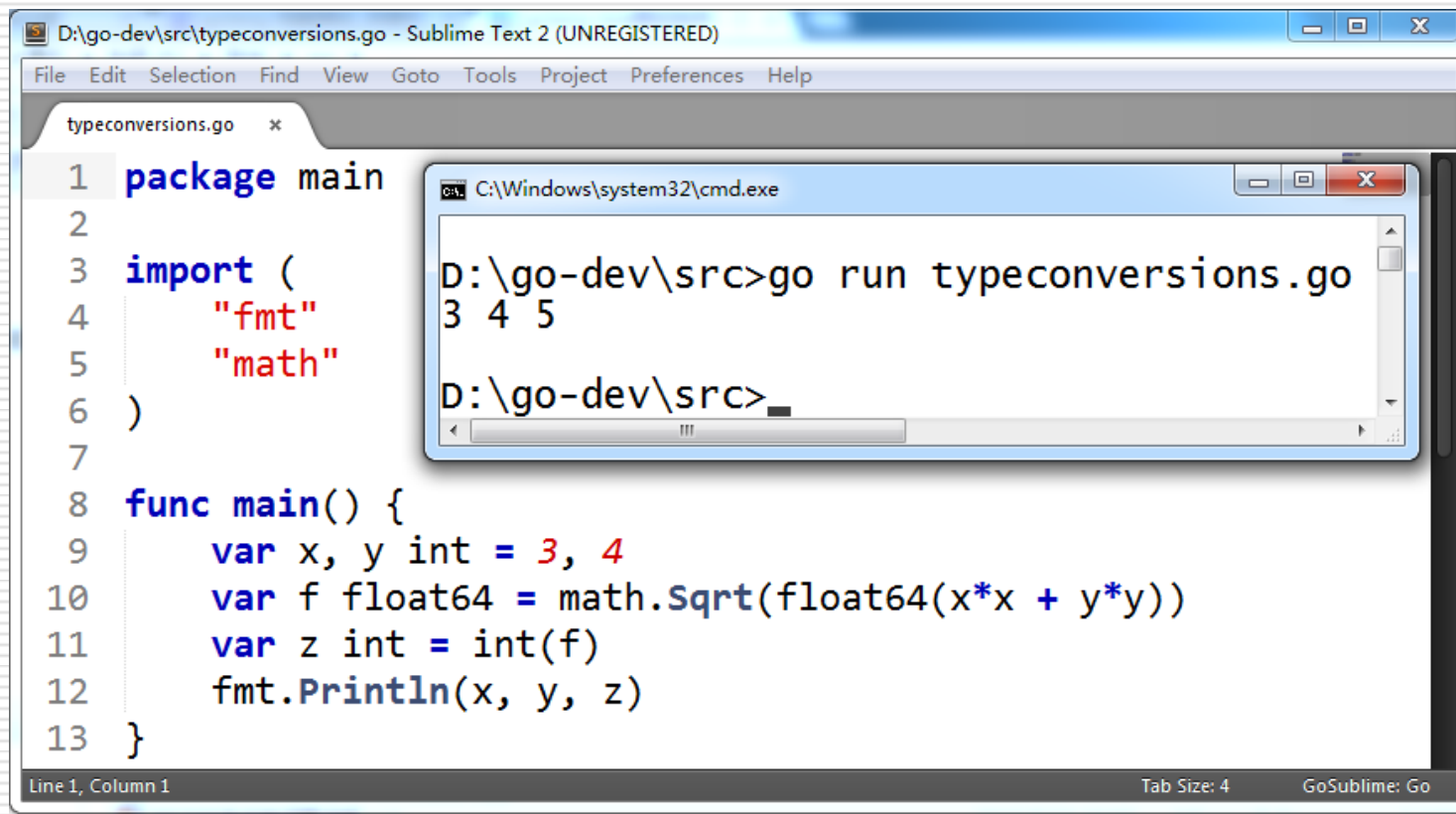
## □ 类型转换——表达式T(v)将值v转换为类型T下的值

### ■ 一些数值转换

□ `var i int = 42; var f float64 = float64(i); var u uint = uint(f)`

□ `i := 42; f := float64(i); u := uint(f)`

### ■ 与C不同，Go在不同类型之间的赋值时，需要显示转换



The screenshot shows a Sublime Text 2 window titled "D:\go-dev\src\typeconversions.go - Sublime Text 2 (UNREGISTERED)". The editor contains the following Go code:

```
1 package main
2
3 import (
4     "fmt"
5     "math"
6 )
7
8 func main() {
9     var x, y int = 3, 4
10    var f float64 = math.Sqrt(float64(x*x + y*y))
11    var z int = int(f)
12    fmt.Println(x, y, z)
13 }
```

Overlaid on the editor is a Windows command prompt window titled "C:\Windows\system32\cmd.exe". It shows the command `D:\go-dev\src>go run typeconversions.go` being executed, with the output `3 4 5` displayed on the next line. The prompt then shows `D:\go-dev\src>` waiting for further input.

At the bottom of the Sublime Text window, the status bar indicates "Line 1, Column 1", "Tab Size: 4", and "GoSublime: Go".

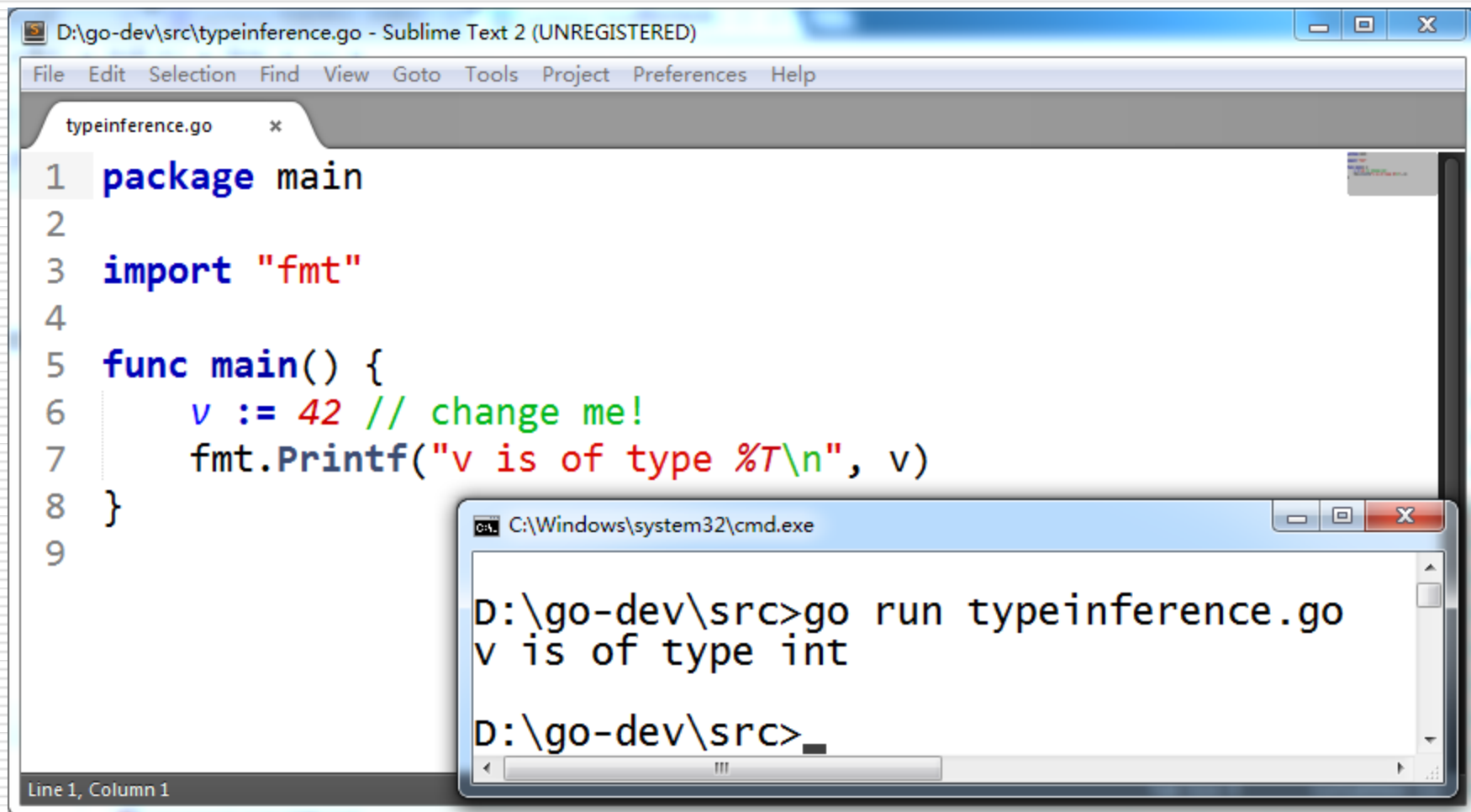
## 1.3 总结(续)

---

- 类型推导——在定义一个变量但不指定其类型时(使用没有类型的var或:=语句), 变量的类型由右值推导得出。
  - 当右值定义了类型时候, 新变量的类型与其相同
    - `var i int`
    - `j := i //j`也是一个int
  - 当右边包含了未指名类型的数字常量时, 新的变量就可能是int、float64或complex128。取决于常量的精度
    - `i := 42 //int`
    - `f := 3.142 //float64`
    - `g := 0.867+0.5i //complex128`

# 1.3 总结(续)

## □ 类型推导举例



The screenshot shows a Sublime Text 2 window titled "D:\go-dev\src\typeinference.go - Sublime Text 2 (UNREGISTERED)". The editor contains the following Go code:

```
1 package main
2
3 import "fmt"
4
5 func main() {
6     v := 42 // change me!
7     fmt.Printf("v is of type %T\n", v)
8 }
9
```

Below the editor, a Windows command prompt window titled "C:\Windows\system32\cmd.exe" is open, showing the execution of the program:

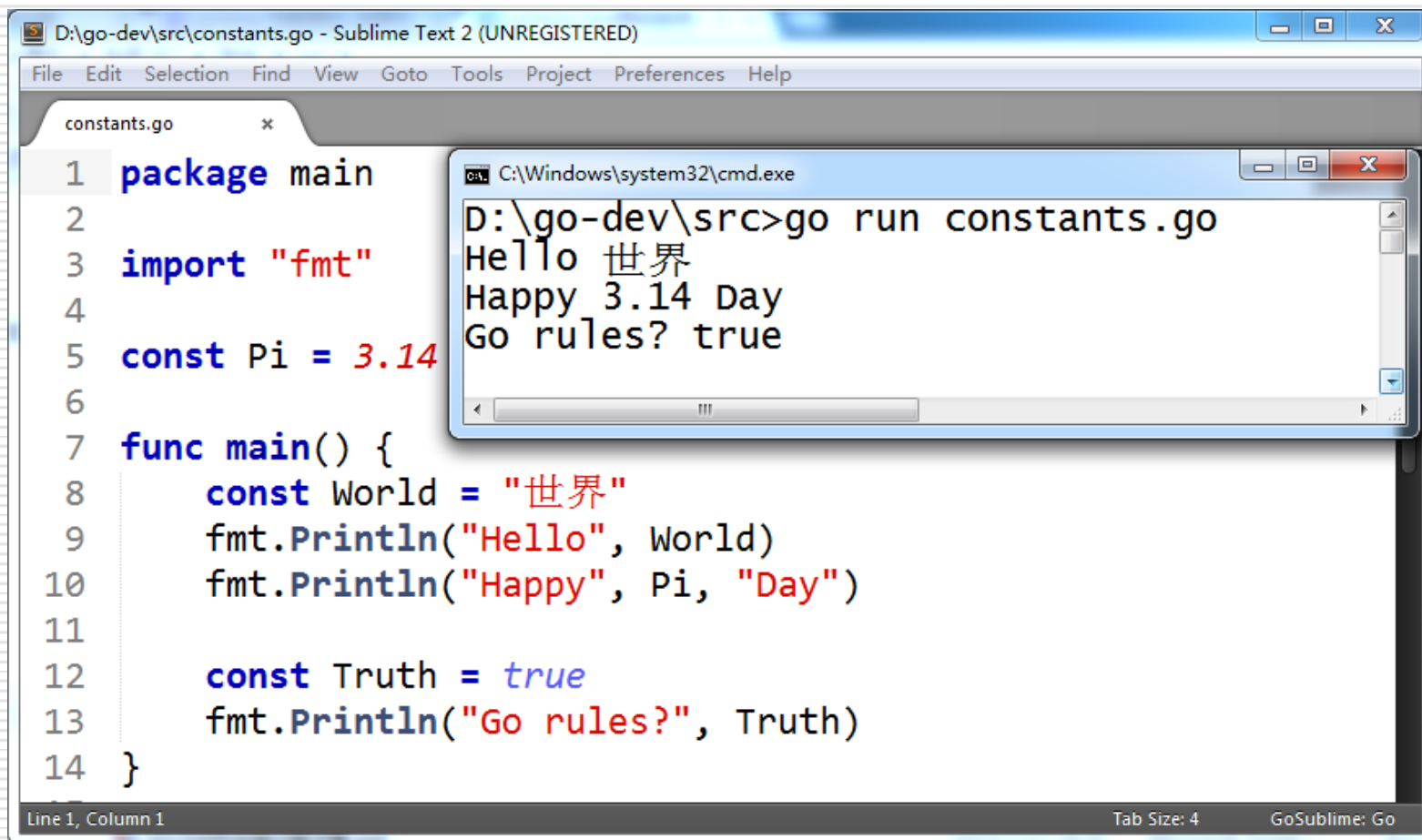
```
D:\go-dev\src>go run typeinference.go
v is of type int

D:\go-dev\src>
```

The status bar at the bottom left of the Sublime Text window indicates "Line 1, Column 1".

## 1.3 总结(续)

- 常量的定义与变量类似，只不过使用const关键字
  - 常量可以是字符、字符串、布尔或数值类型的值
  - 常量不能使用 := 语法定义



The screenshot shows a Sublime Text 2 window editing a file named `constants.go` at the path `D:\go-dev\src\constants.go`. The code defines a `main` package with a `main` function. It imports the `fmt` package and defines three constants: `Pi` (3.14), `World` ("世界"), and `Truth` (`true`). The `main` function prints these values using `fmt.Println`.

```
1 package main
2
3 import "fmt"
4
5 const Pi = 3.14
6
7 func main() {
8     const World = "世界"
9     fmt.Println("Hello", World)
10    fmt.Println("Happy", Pi, "Day")
11
12    const Truth = true
13    fmt.Println("Go rules?", Truth)
14 }
```

Overlaid on the editor is a Windows command prompt window titled `C:\Windows\system32\cmd.exe`. It shows the command `go run constants.go` being executed, with the following output:

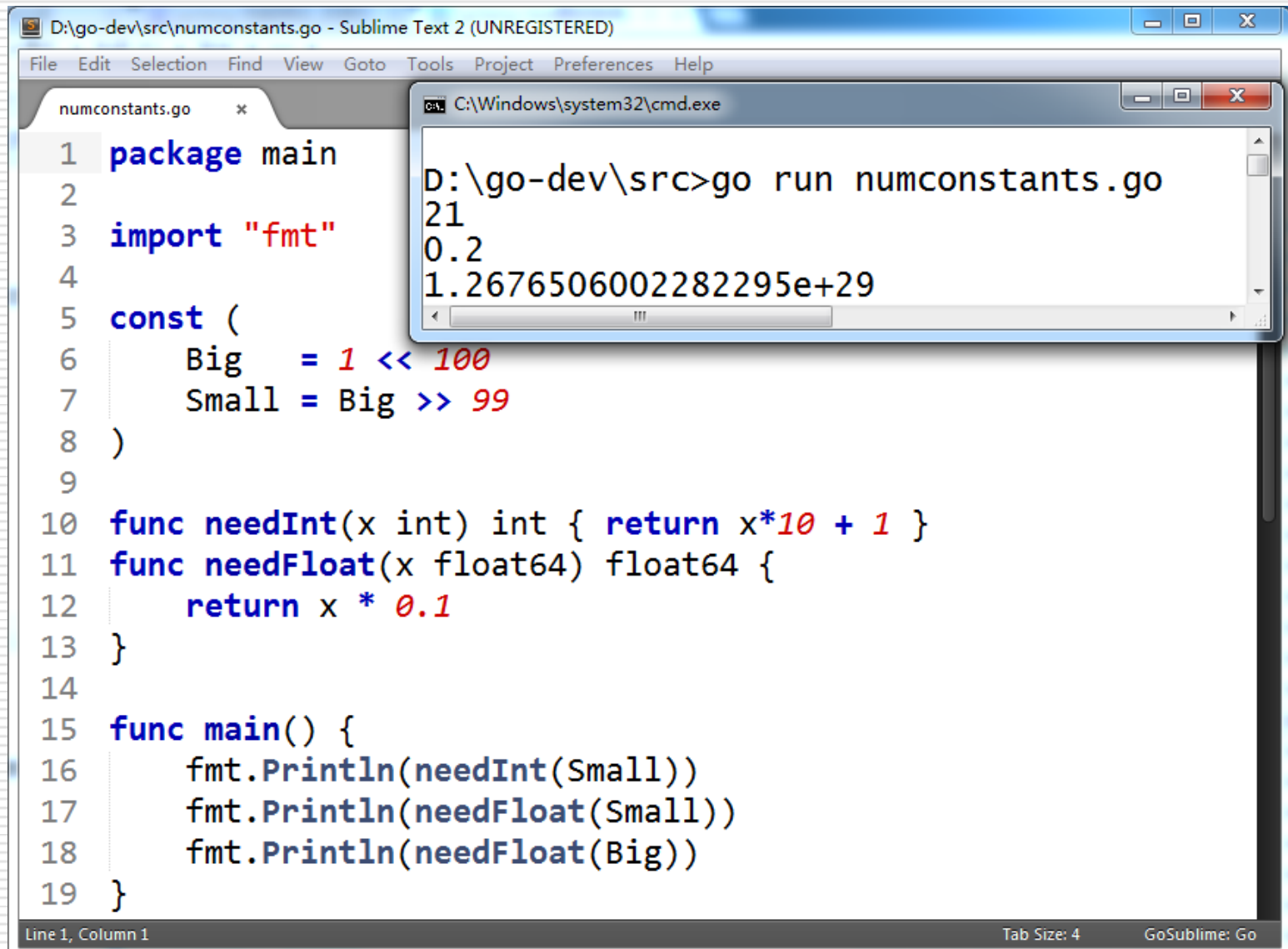
```
D:\go-dev\src>go run constants.go
Hello 世界
Happy 3.14 Day
Go rules? true
```

The status bar at the bottom of the Sublime Text window indicates "Line 1, Column 1", "Tab Size: 4", and "GoSublime: Go".



# 1.3 总结(续)

## □ 数值常量是高精度的值



The screenshot shows a Sublime Text editor window titled "D:\go-dev\src\numconstants.go - Sublime Text 2 (UNREGISTERED)". The editor contains a Go program named "numconstants.go". The program defines two constants, "Big" and "Small", and two functions, "needInt" and "needFloat". The "main" function calls these functions and prints the results. A command prompt window is overlaid on the editor, showing the command "go run numconstants.go" and its output: "21", "0.2", and "1.2676506002282295e+29".

```
D:\go-dev\src\numconstants.go - Sublime Text 2 (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help

numconstants.go x
1 package main
2
3 import "fmt"
4
5 const (
6     Big    = 1 << 100
7     Small = Big >> 99
8 )
9
10 func needInt(x int) int { return x*10 + 1 }
11 func needFloat(x float64) float64 {
12     return x * 0.1
13 }
14
15 func main() {
16     fmt.Println(needInt(Small))
17     fmt.Println(needFloat(Small))
18     fmt.Println(needFloat(Big))
19 }

C:\Windows\system32\cmd.exe
D:\go-dev\src>go run numconstants.go
21
0.2
1.2676506002282295e+29
```

Line 1, Column 1 Tab Size: 4 GoSublime: Go

## 1.4 思考

□ 以下4个for程序段的输出结果是什么？

```
for i := 0; i < 5; i++ {  
    var v int  
    fmt.Println("%d ",v)  
    v = 5  
}  
  
s := ""  
for s != "aaaaa"; {  
    fmt.Println("Value of s:", s)  
    s = s + "a"  
}  
  
for i := 0; i < 3; {  
    fmt.Println("Value of i:", i)  
}  
  
for i := 0; ; i++ {  
    fmt.Println("Value of i is now:", i)  
}
```

## 1.4 思考(续)

---

- 利用递归技术实现 $1+2+\dots+100=?$
- 利用for循环输出九九乘法表
- 利用迭代法计算一个数的平方根

---

**Thank you very much**

*Any comments and suggestions  
are beyond welcome*