# 壹、變數宣告

## 一、單變數宣告

var name string

name = “man”(不能放在全域)

變數名稱+型別

使用 var 關鍵字進行宣告

var age int = 25

變數名稱+型別+初始值  
可以直接指定變數的型別和初始值

var age = 25

也可以不用指定型別，go會根據初始值自動推斷

foo := "hello"

不用加型別，go會自己判斷

注意：只能用在局部變數，全域變數會報錯

## 二、多變數宣告

var a, b int

a = 10

b = 20

可以同時宣告多個相同型別的變數

var x, y int = 5, 8

可以直接指定多個變數的型別和初始值

var x, y = 5, 8

型別也可以省略，讓變數型別根據初始值自動推斷

x, y := 5, 8  
依樣可用簡短宣告

# 貳、算符

## 一、算術運算符

* +：加法運算
* -：減法運算
* \*：乘法運算
* /：除法運算
* %：求餘數

## 二、賦值運算符

* =：將右邊的值賦給左邊的變數。
* +=：先將右邊的值加到左邊的變數上，然後再將結果賦給左邊的變數。
* -=：先從左邊的變數中減去右邊的值，然後再將結果賦給左邊的變數。
* \*=：先將左邊的變數乘以右邊的值，然後再將結果賦給左邊的變數。
* /=：先將左邊的變數除以右邊的值，然後再將結果賦給左邊的變數。
* %=：先將左邊的變數除以右邊的值取餘數，然後再將結果賦給左邊的變數。

## 三、比較運算符

* ==：判斷兩個值是否相等。
* !=：判斷兩個值是否不相等。
* <：判斷左邊的值是否小於右邊的值。
* >：判斷左邊的值是否大於右邊的值。
* <=：判斷左邊的值是否小於等於右邊的值。
* >=：判斷左邊的值是否大於等於右邊的值。

## 四、邏輯運算符

* &&：邏輯與，用來判斷多個條件是否同時滿足。
* ||：邏輯或，用來判斷多個條件是否至少有一個滿足。
* !：邏輯非，用來取反一個條件的結果。

## 五、位運算符

* &：按位與，對每一位進行與操作。
* |：按位或，對每一位進行或操作。
* ^：按位異或，對每一位進行異或操作。
* <<：左移，將二進制位向左移動指定的位數。
* >>：右移，將二進制位向右移動指定的位數。

## 六、其他運算符

* &：取地址，用來獲取變數的內存地址。
* \*：指針，用來聲明指針變數。
* <-：通道操作符，用來發送和接收通道消息。

# 參、條件判斷與迴圈

var productName string = "Galaxy S21 Ultra"

var productPrice int = 10000

var productStock int = 100

透過上篇學到的宣告變數，print出

Product Name: Galaxy S21 Ultra

Product Price: 10000

Product Stock: 100

fmt.Scan(&quantity)

藉由輸入的值存取到quantity的指標，可輸入

Enter quantity to buy:

if quantity > productStock {

fmt.Printf("Insufficient stock. Only %d units available.\n", productStock)

} else if quantity <= 0 {

fmt.Println("Invalid quantity. Please enter a positive integer.")

} else {

productStock -= quantity

fmt.Printf("Remaining stock: %d\n", productStock)

}

透過輸入的值來判斷是否超過productStock，如果大於會print

Enter quantity to buy: 101

Insufficient stock. Only 100 units available.

如果輸入0或以下會print

Enter quantity to buy: 0

Invalid quantity. Please enter a positive integer.

如果正常輸入沒超過productStock的話會print

Enter quantity to buy: 80

Remaining stock: 20

並且更新productStock的值

for {

fmt.Printf("Product Stock: %d\n", productStock)

var quantity int

fmt.Print("Enter quantity to buy: ")

fmt.Scan(&quantity)

if quantity > productStock {

fmt.Printf("Insufficient stock. Only %d units available.\n", productStock)

} else if quantity <= 0 {

fmt.Println("Invalid quantity. Please enter a positive integer.")

} else {

productStock -= quantity

fmt.Printf("Remaining stock: %d\n", productStock)

}

if productStock == 0 {

fmt.Println("Stock is out. Thank you for shopping!")

break

}

}

加入for迴圈，這邊用的是無窮迴圈，要透過相對應條件觸發break才會跳出這個迴圈(這邊是當stock=0時會跳出)

Product Name: Galaxy S21 Ultra

Product Price: 10000

Product Stock: 100

Enter quantity to buy: 3

Remaining stock: 97

Product Stock: 97

Enter quantity to buy: 8

Remaining stock: 89

Product Stock: 89

Enter quantity to buy: 89

Remaining stock: 0

Stock is out. Thank you for shopping!

## 補充：

剛提到for迴圈，以上範例是用for{}來達到無窮迴圈的效果，以下為其他for迴圈使用效果

var sum int

for i := 0; i <= 10; i++ {

sum += i

}

fmt.Printf("Sum of first 10 natural numbers: %d\n", sum)

從1加到10，print出

Sum of first 10 natural numbers: 55

for range之後會配合map一起實做

key := []string{"a", "b", "c"}

for i := 0; i < len(key); i++ {

fmt.Println(key[i])

}

條件如果遇到map、slice，可以用len()取長度，索引從0開始

a

b

c

var(

foo string

bar int

)

var (

foo = 5

bar = "abc"

)

var (

name string = "Alice"

age int = 25

country string = "USA"

)

可同時宣告不同的型別