起手

```
package main

import (
    "fmt"
)

func main() {

}
```

宣告

```
var number int = 0
number := 0
```

兩數互換

```
a,b = b,a
```

iostream

```
fmt.Println(number)
fmt.Printf("%d\n", number)
fmt.Print(number)
```

```
PS C:\Users\USER\OneDrive - gapps.ntust.edu.tw\11001\應用程式語言\go-test> go run .\HW00-B10930233.go
1
1
1
```

Sscanf 讀取一段字串

```
var name_and_age string = "Kenneth 22"
    var number int = 0
    var str string = ""

    fmt.Sscanf(name_and_age, "%s%d", &str, &number)
```

```
PS C:\Users\USER\OneDrive - gapps.ntust.edu.tw\11001\應用程式語言\go-test> go run .\HW00-B10930233.go
Kenneth 22
```

```
    fmt.Println(str, number)
```

Scanf 讀取輸入的東西

```
fmt.Scanf("%s%d", &str, &number)
fmt.Println(str, number)
```

```
PS C:\Users\USER\OneDrive - gapps.ntust.edu.tw\11001\應用程式語言\go-test> go run .\HW00-B10930233.go
Joe 22
Joe 22
```

Scan 與 Scanf 一樣

```
fmt.Scan(&str, &number)
fmt.Println(str, number)
```

if-else

```
    a := 1
    b := 2

    if a >= b {
        fmt.Print(a)
    } else {
        fmt.Print(b)
    }
```

switch

```
    var age = 45
    switch {
    case age >= 60:
        fmt.Println("老人")
    case age >= 40:
        fmt.Println("壯年")
    default:
        fmt.Println("小孩")
    }
    switch age {
    case 60:
        fmt.Println("60")
    case 40:
        fmt.Println("40")
    default:
        fmt.Println("!60 !40")
    }
```

for 一般迴圈

```
for i := 0; i < 10; i++ {
        fmt.Println(i)
    }
```

for 無限迴圈

```
for {
```

```
    }
```

for while 迴圈

```
state := true
    for (state) {
        fmt.Println("yes")
    }
```

陣列宣告

```
var number[6]int
number := [3]int{1,2,3}
number := [...]int{1,2,3,4}
var number = make([]int,6)
```

陣列 **range** 使用

for 索引值,元素 := range array{}

```
    number := [...]int{1, 2, 3, 4}

    for i, num := range number {
        fmt.Println(i+1, ":", num)
    }
```

_代表沒有要使用，但是語法需要，所以用_代替

```
    number := [...]int{1, 2, 3, 4}

    for _, num := range number {
        fmt.Println(num)
    }
```

或是基本的迴圈輸出

```
    number := [...]int{1, 2, 3, 4}

    for i := 0; i < len(number); i++ {
        fmt.Println(number[i])
    }
```

二維陣列宣告

```
    number := [...][4]int{
        {1, 2, 3, 4},
```

```
        {5, 6, 7, 8},
        {9, 10, 11, 12},
    }

    for i := 0; i < len(number); i++ {
        for j := 0; j < 4; j++ {
            fmt.Println(number[i][j])
        }
    }
```

```
    var number = make([][]int, 6)
    for i := 0; i < len(number); i++ {
        number[i] = make([]int, 2)
    }
    for i := 0; i < len(number); i++ {
        for j := 0; j < len(number[i]); j++ {
            number[i][j] = j + i*2
        }
    }
    for i, _ := range number {
        for j, num := range number[i] {
            fmt.Println(j+i*2, num)
        }
    }
```

slice 宣告

從 number 拿 number[0]到 number[6-1]

```
sl := number[0:6]
```

從頭拿到 5

```
sl := number[:6]
```

從 1 拿到尾

```
sl := number[2:]
```

copy 與 append

s1 copy 到 s2

s3 = s2 再加上一個元素 1

```
    number := [6]int{1,2,3,4,5,6}
    s1 := number[0:3]
    s2 := make([]int,3)
    copy(s2,s1)
```

```
    s3 := append(s2,1)
```

map 宣告

```
    number_str := map[int]string{
        1 : "abc",
    }
    str_number := make(map[string]int)
```

新增 map 的值

```
    str_number := make(map[string]int)
    str_number["abc"] = 2
```

用 key 尋找 map 值

```
    number,ok := str_number["abc"]
    if (ok){
        fmt.Println(number)
    } else{
        fmt.Println("not found")
    }
```

同樣的 不需要的值也可以用_去掉

```
    number, _ := str_number["abc"]
    fmt.Println(number)
```

刪除 map 中的該 key 的 key 與元素

```
delete(str_number,"abc")
```

檔案輸入

```
func importScore() {
    importByte, err := ioutil.ReadFile("StudentList.txt")
    if err != nil {
        fmt.Println(err)
        return
    }
    importString := string(importByte)
    importArrayLine := strings.Split(importString, "\r\n")

    for i := 0; i < len(importArrayLine); i++ {
        var tmpScore Score
        tmpScore.Name = "[" + importArrayLine[i] + "]"
        score = append(score, tmpScore)

    }
```

```
}
```

檔案輸出

```
func exportScore() error {
    var exportString string
    for i := 0; i < len(score); i++ {
        exportString += fmt.Sprintf("%s\n\t%s%d\n\t%s%d\n\t%s%d\n%s%d%s%.2f\n\n",
score[i].Name, "請輸入國文成績：", score[i].Chinese_Score, "請輸入數學成績：",
score[i].Math_Score, "請輸入英文成績：", score[i].English_Score, "總分：",
score[i].Total_Score, "/平均：", score[i].Average_Score)
        fmt.Printf("%s\n\t%s%d\n\t%s%d\n\t%s%d\n%s%d%s%.2f\n\n", score[i].Name, "請輸入國文
成績：", score[i].Chinese_Score, "請輸入數學成績：", score[i].Math_Score, "請輸入英文成績：",
score[i].English_Score, "總分：", score[i].Total_Score, "/平均：", score[i].Average_Score)
    }
    data_byte := []byte(exportString)
    err := ioutil.WriteFile("ScoreBook.txt", data_byte, 0777)
    return err
}
```

輸出 err 的”錯誤訊息”(或是把 fmt.Errorf 換成 errors.New

```
func importScore(str string) (bool, error) {
    importByte, err := ioutil.ReadFile(str)
    if err != nil {
        fmt.Println(err.Error())
    }
    importString := string(importByte)
    importArrayLine := strings.Split(importString, "\r\n")

    for i := 0; i < len(importArrayLine); i++ {
        var tmpScore Score
        tmpScore.Name = "[" + importArrayLine[i] + "]"
        score = append(score, tmpScore)

    }
    return false, err
}
```

method(方法) 用法 半物件導向的概念

前提是有一個 Circle 的物件(Type)或是結構(Struct)

```
type Circle int

var circle Circle
func (circle Circle) getCircleArea() float64 {
    return float64(circle) * float64(circle) * 3.1415926

}

fmt.Printf("%s%.2f\n\n", "面積：", circle.getCircleArea()
```

```
type movie struct {
    moive_name  string
    movie_money int
}
type movie2 movie

var test2 movie2

func (_movie movie2) test(){

}
test2.test()
```

刪除指定 slice index 的 element

```
func main() {
    var s1 = []int{1, 2, 3, 4, 5}
    s1 := remove(s1, 1)
    fmt.Println(s2)

}
func remove(slice []int, s int) []int {
    return append(slice[:s], slice[s+1:]...)
}
```

輸出結果

```
[1 3 4 5]
```

偵測是不是整數

```
func main() {

    var test string
    var i int
    fmt.Scan(&test)
    i, err := strconv.Atoi(test)
    if err != nil {
        fmt.Print("error")
    } else {
        fmt.Print(i)
    }

}
```