CH4-4
邏輯控制

概要 邏輯控制

在程式語言中，邏輯控制是非常重要的一環，它可以用來執行多次任務、跳轉到其他程式碼、或是依據不同的選擇，做出分支。總而言之，可以依據程式碼的不同，依照不同的情況做出不同方式的處理。

概要 條件語句

設定條件，程式執行後判斷是否符合條件，再將其分支到不同的結果，這就是條件語句的用途。
Swift有兩種條件語句，第一種是「if」，第二種是「switch」，前者是用來判斷較簡單且情況較少的狀況，後者則是偏向較複雜且可能有多種情況分支的情境。

條件式判斷

在學條件語句之前，我們必須先學會基本的條件式判斷。
例 當我們要判斷a是否等於b時，必須使用兩個等於來比較a及b

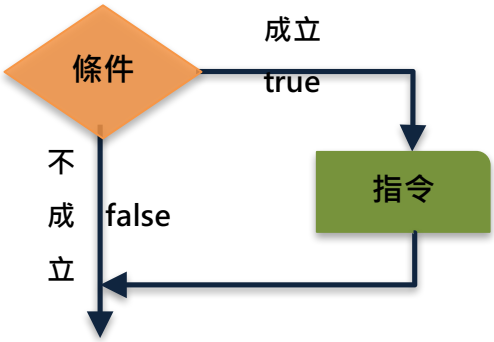
```
var a=5,b=5;

if (a == b ){
    print("a及b是相等的")
}
```

除了相等之外，還有許多不同的條件式可使用，如下表所列。

==	比較左右兩數是否相等
<	比較左邊是否小於右邊
<=	比較左邊是否不大於右邊
>	比較左邊是否大於右邊
>=	比較左邊是否不小於右邊
!=	比較左右兩數是否不相等

使用方法 if

當我們需要判斷條件成立時要執行什麼動作，就會使用if。if會檢查我們設定的條件是否有成立，如果成立的話，就執行括弧內的指令。



If語句最簡單的呈現方式就是只有一個條件式，若成立就執行指令。

例

```
var weather="rain"

if (  weather == "rain"  ){
    print("下雨，記得帶傘")
}
```

顯示結果

```
下雨，記得帶傘
```

上面的範例呈現，若符合條件的話，就會顯示訊息，不符合也不會有其他的文字顯示。

else指令

我們可以讓程式做得更縝密一點，變成「若成立，執行指令、若不成立，執行另一種指令。這時候，我們就會需要用到else指令。

例

```
var weather="sunny"

if (  weather == "rain"  ){
    print("下雨，記得帶傘")

}else{
    print("沒下雨，今天是好天氣")
}
```

顯示結果

```
沒下雨，今天是好天氣
```

if-else指令

若是我們有很多個條件，亦可以利用 if-else 語句來執行，如下範例。

例

```
var weather="sunny"

if (  weather == "rain"  ){
    print("下雨，記得帶傘")

}else if(weather=="sunny"){
    print("出太陽，注意防曬")

}else{
    print("陰天")
}
```

顯示結果

```
出太陽，注意防曬
```

在這個範例中，今天的天氣是出太陽，因此在執行條件語句時，觸發了 else if 內的分支，執行「列印“出太陽，注意防曬”」的指令。

當然，我們可以使用更多的 else if 來判斷其他的狀況。

例

```
var grade=70;

if (  grade == 100  ){
    print("評等：S")
}else if(grade>=90 && grade<100){
    print("評等：A")
}else if(grade>=80 && grade<90){
    print("評等：B")
}else if(grade>=70 && grade<80){
    print("評等：C")
}else if(grade>=60 && grade<70){
    print("評等：D")
}else {
    print("評等：E")
}
```

顯示結果

```
評等：C
```

使用方法 switch

switch語句是由許多的case組成，每一個case都是一個分支，與if不同的是，每一次執行程式遇到switch時，必定會執行其中一個分支，換句話說，每一個可能發生的事件，都一定會有對應的分支。Switch語法是利用default來表示其它的例外，與else的功用十分雷同。

上一頁的最後一個例子，若我們用switch來改寫的話，會簡單很多。

例 可以將分數除以10，取其商數，在對商數做級距判斷。

```
var grade = 72;

switch(grade/10){      //將grade除以10取商數，以便判斷分數級距
    case 10:
        print("評等：S")
    case 9:
        print("評等：A")
    case 8:
        print("評等：B")
    case 7:
        print("評等：C")
    case 6:
        print("評等：D")
    default :
        print("評等：E")

}
```

例 我們可以使用...來表示一段區間

```
var grade = 72;
switch(grade){
    case 100:
        print("評等：S")
    case 90...99:
        print("評等：A")
    case 80...89:
        print("評等：B")
    case 70...79:
        print("評等：C")
    case 60...69:
        print("評等：D")
    default :
        print("評等：E")

}
```

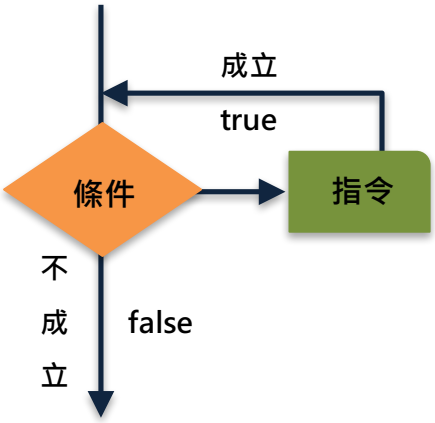
顯示結果

```
評等：C
```

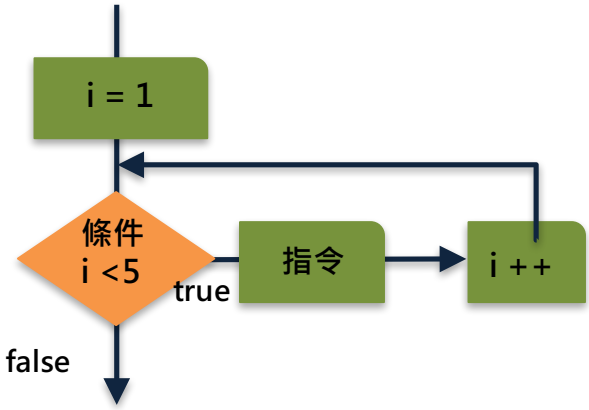
概要 迴圈

迴圈是程式中不可或缺的重要元素，它可用來執行多次任務，因此需要**重複執行多次相同事件**時，就可以使用迴圈。
迴圈有分為 for 及 while 兩種，差別在於，for 迴圈為指定要重複執行一定的次數，執行完畢即跳出。而 while 則是滿足條件時重複執行，不滿足條件時跳出。我們可以運用兩者的特性做不一樣的運用。

while 迴圈



for 迴圈



使用方法 for

上面介紹過，for 是用來指定**重複執行一定次數**，而 for 迴圈又分為普通的 for 迴圈及 for-in 迴圈，接下來我們就分別介紹這兩種用法。

For 條件遞增

Swift中for條件遞增的語法與一般常見的語言的語法沒有太大的差別，就是利用（for-condition-increment）語句來執行。

```
for initialization ; condition ; increment {
statements
}
```

第一次剛進入迴圈時，初始的計數值（initialization）用來代表初始值，初始化迴圈一切的值。重複次數（condition）則為條件式，當迴圈跑到超出條件式的範圍時，則迴圈結束，反之則執行括弧內的內容。而累加計算（increment）是遞增的語句，每跑完一次迴圈則會依照它來對值做累加的計算。

例 利用for迴圈計算1至100的總和

```
var sum=0

for var i = 1; i<=100 ; i++ {
    sum = i + sum
}

print(sum)
```

顯示結果

```
5050
```

for-in

我們可以使用 for-in 迴圈來檢視一個集合裡面所包含的數，像是陣列裡的元素、或是某一個數字區間。

例 利用for-in迴圈一一列出陣列中的值

```
var ageArray=[27,8,19,43,15]

for index in ageArray{
    print("I'm \(index) years old.")
}
```

顯示結果

```
I'm 27 years old.
I'm 8 years old.
I'm 19 years old.
I'm 43 years old.
I'm 15 years old.
```

例 利用for-in迴圈來列出區間內的值（2...6 為 2至6之間的區間）

```
for index in 2...6{
    print("I'm \(index) years old.")
}
```

顯示結果

```
I'm 2 years old.
I'm 3 years old.
I'm 4 years old.
I'm 5 years old.
I'm 6 years old.
```

使用方法 while

while 迴圈的特性就是會一直不斷的重複執行，直到條件式變為 false 時才會停止。while 迴圈又分為 while 及 do-while 兩種，這兩種的差別只在於 while 會先判斷條件式之後再決定是否執行，而 do-while 則是一定會先執行一次，執行完一次之後才判斷條件式。因差異不大，此處只介紹 while 迴圈的使用。

While

while 迴圈從判斷條件式開始，如果為 true，則會重複執行括號內的一整段程式碼，反之，若為 false 就會跳出重複的迴圈。

```
while  條件式{
statements
}
```

例 利用while迴圈來找出小於50的數中，5的倍數。

```
var num=0;
var numArray=[Int]()

while num < 50 {
    num=num+5
    numArray.append(num)
}

for index in numArray{
    print(index)
}
```

顯示結果

```
5
10
15
20
25
30
35
40
45
50
```