

決策主題（二）：條件選擇與迴圈

by 田弘華 Hung-Hua Tien

0. 基本觀念

1. 我們會根據一件事的**True**或**False**來做決定，進而執行不同的動作。

以早餐為例，如果冰箱有雞蛋，而且有時間，則會煎雞蛋當作早餐，接著坐下來吃。否則，就會帶一根巧克力棒路上吃。也就是說，我們會根據條件的真假做出不同的決定。

2. 同樣的道理，電腦會根據「布林條件式」來執程式碼。

- 如果條件判斷式的結果為**True**(真)，則執行**if**程式碼區塊；
- 如果條件判斷式的結果為**False**(假)，則執行**else**程式碼區塊。
- 這種敘述式稱為「條件敘述式」(**if conditional statement**)。
- 常見的有單向選擇、雙向選擇、多向選擇與巢狀選擇四種。

3. 只要條件式維持**True**，電腦就可以執行迴圈（重複執行一組程式碼）。這叫做條件迴圈(**conditional loop**)。

「**while**條件迴圈」是用在程式碼要執行的切確次數未知的時候。只要條件式維持**True**，它就會一直執行。例如猜數字遊戲，只要使用者沒有猜對就繼續玩，直到猜對為止。再不知道要重複多少次時，就使用這種迴圈。

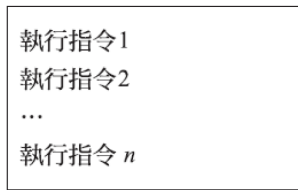
1. 單向選擇

1-1 選擇結構

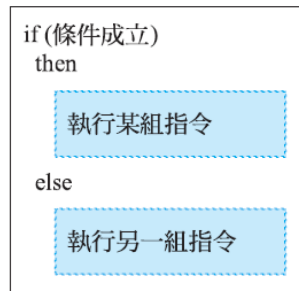
由於**Python** 是一種程序性的語言（**procedure language**），表示程式是一行接一行的循序執行，上一行敘述執行後接下一行敘述。但有時有些敘述不必執行想跳過去的話，此時就要靠選擇敘述來完成。好比程式會轉彎，避開不要執行的敘述，執行某一條件為真時的對應處理敘述。

演算法通常是由下列三種結構所組成：

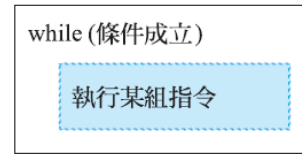
- 序列 (sequence)
- 決策 (decision)
- 重覆 (repetition)



(a) 序列結構



(b) 決策結構



(c) 重覆結構



選擇敘述表示在某一條件下，做某些事情。

「如果判斷條件成立，就執行條件成立的動作；若判斷條件不成立，則執行條件不成立的動作」。

1-2 基本觀念

單向選擇結構是最簡單的選擇結構，只有一個方向的選擇，如果條件判斷成立，就執行條件為真的動作。

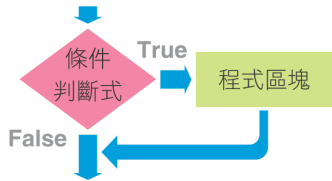
例如：「若週末天氣好的話，我們就去打球」。



● 圖5-1 單向選擇敘述示意圖

1-3 語法指令

單向判斷 (if)

Python 語言	流程圖
<pre>if 條件判斷: [程式區塊]</pre>	
說明： 如果條件判斷成立或為 True，就執行「程式區塊」，若不成立則不會執行程式區塊。	

if <布林條件式>:

<當條件式為True時，所要執行的程式碼。>

(1)if是條件指令。

(2)布林條件式是指任何敘述結果為True或False的布林值。

(3)冒號:很重要。它把條件式為True時要執行的程式碼區塊分隔。

(4)縮排程式碼：表示要執行的程式碼區塊需要縮排(indentation)。

～Python對縮排很挑剔；屬於區塊的每一行程式碼都要有相同的縮排距離，習慣上是四個空白鍵。**

例如：下雨時會如何

```
# raining為True，會執行if程式區碼。  
raining = True  
if raining:  
    print("外面濕濕的")  
    print("穿雨鞋")  
    print("帶雨傘")  
  
# 把raining改為False，什麼都不會顯示。  
raining = False  
if raining:  
    print("外面濕濕的")  
    print("穿雨鞋")  
    print("帶雨傘")
```

例：比較數字（一）

```

# 數字資料 (一)
num = int(input("請輸入一個數字: "))
if num > 0:
    print("數字為正數")
print("比較完畢")

# 數字資料 (二)
num = int(input("請輸入一個數字: "))
if num < 10:
    print("比10小的數字")
print("完成")

# 輸入5, 10, -3看結果

```

～多個單一if條件式

如果你的程式比較複雜，可能需要多個布林條件判斷式才行。

(1)串接的條件式

我們直接將多個if條件式，一個接一個地依序安排在程式中。執行程式時，每個if條件句都會被依序執行到；若條件成立就執行區塊內的敘述，若不成立就接著執行下一個if條件式。

例：比較數字（二）

```

num = int(input("輸入一個數字: "))

if num > 0:
    print("是正數")
if num < 0:
    print("是負數")
if num == 0:
    print("數字是0")
print("比較完畢!")

```

(2)巢狀條件式

有些事情事在某個條件成立下，才需要再做下一步的條件判斷。也就是第1層條件成立時，才會執行第2層的程式寫法。

例如：麥片吃完了，才考慮要買哪一種麥片。

例：比較數字（三）

```

# 巢狀條件式

num_a = int(input("輸入第一個數字? "))
num_b = int(input("輸入第二個數字? "))

if num_a < 0:
    print("第一個是負數")
    if num_b < 0:

```

```

        print("第二個也是負數")
    print("比較完成")

# 非巢狀結構
num_a = int(input("輸入第一個數字？ "))
num_b = int(input("輸入第二個數字？ "))

if num_a < 0:
    print("第一個是負數")
if num_b < 0:
    print("第二個是負數")
print("比較完成")

```

範例：假設你今天要煮義大利麵：請先弄清楚你有(1)麵和(2)醬料嗎？
然後就知道是否晚餐是義大利麵了！

方法（一）：直接把兩個問題用and合併：你是否有麵和醬料呢？

方法（二）：先問有沒有麵，在問有沒有醬料呢？

```

noodle = input("有麵嗎？ YES or NO")
sauce = input("有醬料嗎？ YES or NO")

# 方法（一）

if noodle=="YES" and sauce=="YES":
    print("今天晚上吃義大利麵")

# 方法（二）

if noodle == "YES":
    if sauce == "YES":
        print("今天晚上吃義大利麵")

```

隨堂練習：今天去超市購物，在入口處販售巧克力，你心理想要買嗎？要買多少條呢？餓了才買！如果要買，便宜就多買點！低於10元買10條，高於50元買1條，其他買3條吧！店員看到你買多於5條時，就會和你說：你今天好餓吧！

```

# Input
hungry = input("現在餓嗎？ YES or NO")
price = float(input("一條巧克力多少錢？ "))
bars = 0

# if Condition => Process + Output
if hungry == "YES":
    if price < 10:
        bars = 10
        print(f"便宜，多買點， 買{bars}條")
    if 10<= price <= 50:
        bars = 3
        print(f"還好，買一些， 買{bars}條")

```

```
if price > 50:
    bars = 1
    print(f"好貴，買一點，買{bars}條")
if hungry == "NO":
    bars = 0
    print(f"不餓就不買了，所以買{bars}條")
if bars > 5:
    print(f"店員說：你今天好餓吧！，買了{bars}條！")
```

2. 雙向選擇

雙向選擇結構有兩個方向的選擇二選一，「如果...就...，否則就...」。

- 如果條件判斷成立，就執行條件為真的動作；
- 如果條件判斷不成立，就執行條件為假的動作。



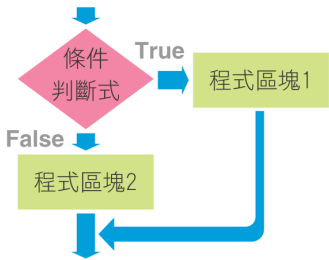
● 圖5-5 雙向選擇敘述示意圖

例如：「如果週末天氣好的話，我們就出去打球，否則就去看電影」。

語法指令

雙向選擇：**if/else**指令，布林條件式，有兩個冒號，兩個程式區塊，分屬**if**區塊和**else**區塊。

雙向判斷 (if ~ else)

Python 語言	流程圖
<pre>if 條件判斷: [程式區塊1] else: [程式區塊2]</pre>	
說明： 如果條件判斷成立或為 True，就執行底下「程式區塊 1」，若條件不成立則執行「程式區塊 2」。	

要點1: Python條件判斷指令：雙向選擇用 **if...else**。

要點2: 布林運算式：程式敘述經條件判斷運算後，得到布林值 **True** 或 **False**。

- 我們做決策或進行條件判斷時，回答Yes或No，在電腦對應的是True或False。
- 電腦無法直接依據yes/no決策結果進行運算，而是依據對應的布林值True/False進行運算。

要點3: Python語法：**if**與**else**兩處要加冒號：。

要點4: Python 程式區塊：以縮排表示區塊範圍。

- Python程式區塊，用縮排來表示，而非以一對大括號「{}」表示執行的範圍。
- 同一個程式區塊的程式，每行都要空相同長度，通常是4個空白鍵(space)。
- Tab 也可用於表示縮排，但是空白鍵與Tab 鍵不要混用。

例如：時間表

```
day = input("輸入星期幾")  
if day == "星期六" or day == "星期天":  
    alarm = "OFF"  
    print("今天週末，睡晚一點!")  
else:  
    alarm = "ON"  
    print("起床，準備上學")
```

範例: 判斷及格與不及格

寫一個程式判斷並顯示所輸入的成績是及格還是不及格。

解題想法

可以使用雙向選擇結構撰寫程式，判斷成績是否及格，及格就顯示「有及格啦～」，不及格就顯示「不及格ㄟ！」。

```
score = int(input('請輸入一個成績？'))
if score >= 60:
    print('有及格ㄟ~')
else:
    print('不及格ㄟ！')
```

程式解說

- 第1行：於螢幕顯示「請輸入一個成績？」，允許使用者輸入成績，並經由int 函式轉換成整數，指定給變數score。
- 第2到3行：用(if) 對score 做判斷，若大於等於60 分，就輸出「及格」。
- 第4到5行：用(else) 對score 做判斷，若小於60 分，就輸出「不及格」。

隨堂練習：滿2000 打九折

請寫一個程式幫助店家計算顧客所需付出的金額。

採買物品時，有時會遇到店家為了刺激消費，會使用滿額折扣。例如，滿2000 打九折，未滿2000 則不打折。

解題想法

可以使用雙向選擇結構撰寫程式，判斷購買金額是否在2000 元以上，若購買金額在2000 元以上，輸出購買金額乘以0.9；否則依照原價輸出。

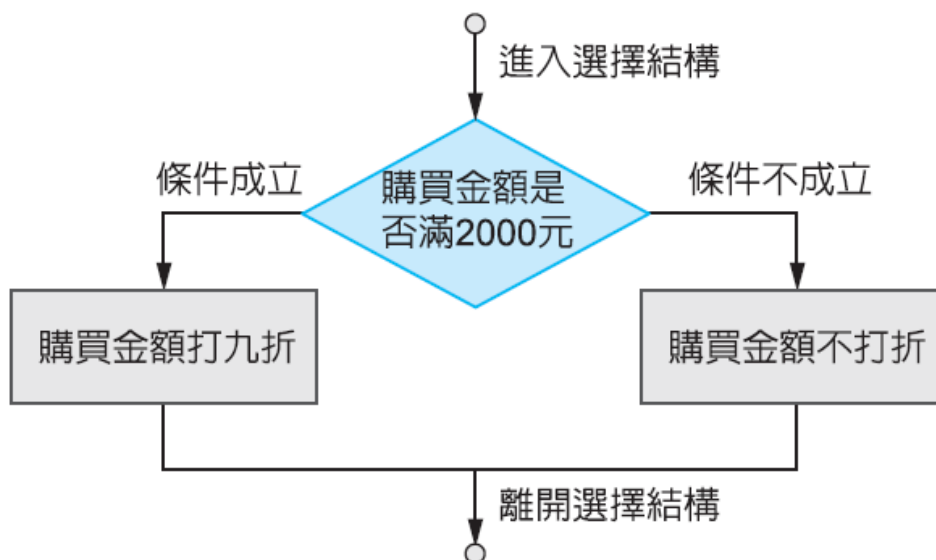


圖 A-4 流程圖

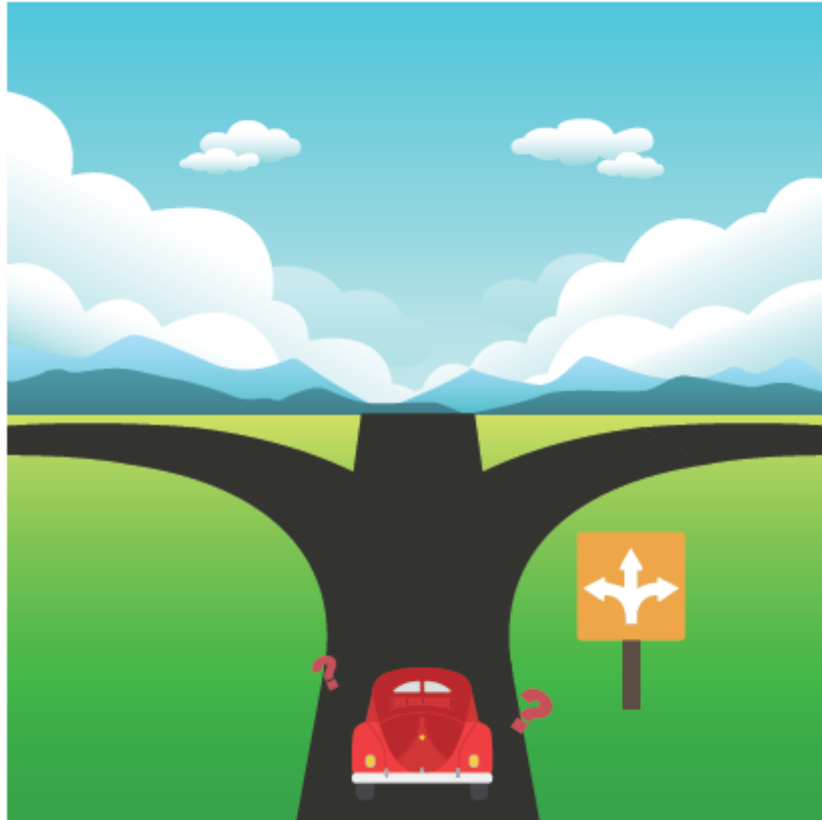
```
cost = int(input('請輸入購買金額？'))
if cost >= 2000:
    print(cost * 0.9)
else:
    print(cost)
```

程式解說

- 第1 行：於螢幕顯示「請輸入購買金額？」，允許使用者輸入購買金額，並經由 `int` 函式轉換成整數，指定給變數 `cost`。
- 第2 到5 行：條件判斷(`if`) 對 `cost` 做判斷，大於等於2000 就將該數值打九折(第2 到3 行)，否則該數值不打折(第4 到5 行)。

3. 多向選擇

一個問題需要在多個條件式使用不同的程式碼區塊。多向選擇結構有多個條件判斷的選擇結果。由上而下依序進行條件判斷，如果條件判斷成立，就執行該條件為真的動作；如果條件判斷不成立，就繼續執行下一個條件判斷，直到結束。



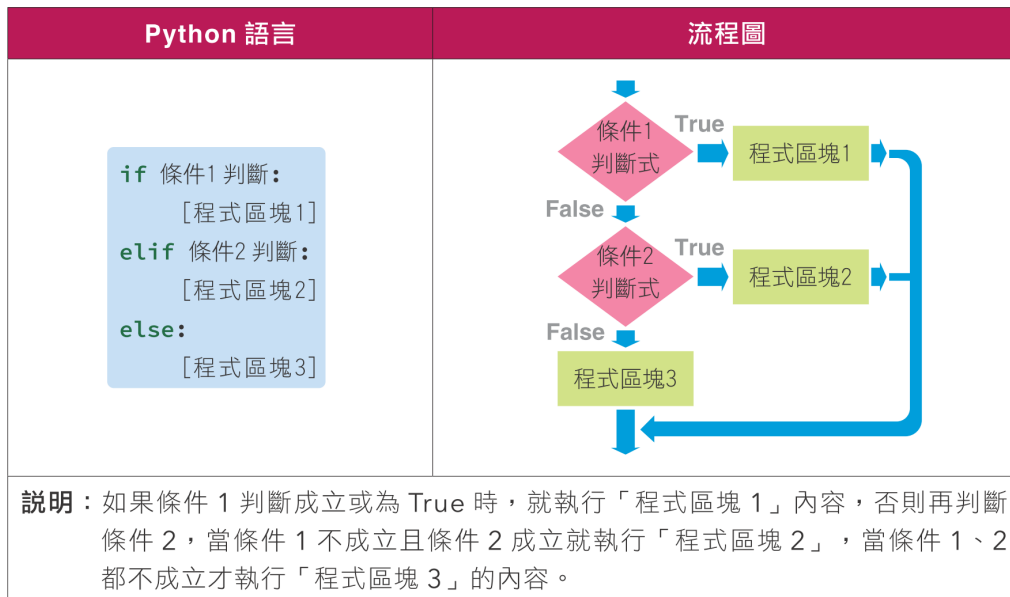
● 圖5-7 多向選擇敘述示意圖

例如：「如果週末天氣好的話，我們就出去打球；如果週末陰天的話，我們就在附近公園逛逛；否則就去看電影」。

語法指令

條件判斷指令用 `if+elif+else`；有多個布林條件式；有多個冒號；多個程式區塊，分屬於 `if` 區塊、`elif` 區塊和 `else` 區塊。

多向判斷 (if ~ elif ~ else)



- 如果多向選擇有包含關係的話，條件比較嚴苛的放上面。
- if-elif-else 條件式中的elif可以視需要加入更多個，讓if-elif-else進行更多的條件判斷。
- if-elif-else 條件式中最後的else可以可無，但只要有加的話，就一定要放在最後。
- 程式會依序確認每個決策點的條件判斷，一遇到結果為True時，就會執行該條件式的程式區塊，然後跳離整個if-elif-else條件式，不會再確認其他條件式。
- 程式會依序確認每個決策點的條件判斷，如果所有條件判斷結果都是False，則執行else敘述的程式區；如果沒有else敘述，程式不做任何事、繼續執行後續的敘述。

範例：星期程式時間表



```

# 鬧鐘、經濟學、運算思維、新聞傳播課程
day = input("輸入星期幾: ")
if day == "星期一" or day == "星期三":

```

```

alarm = "7:30am"
economics = True
coding_class = True
news = False
elif day == "星期二" or day == "星期四":
    alarm = "6:30am"
    economics = False
    coding_class = False
    news = True
elif day == "星期五":
    alarm = "7:30am"
    economics = True
    coding_class = False
    news = False
else:
    alarm = "OFF"
    economics = False
    coding_class = False
    news = False
print(alarm, economics, coding_class, news)

```

～請多試試不同的星期幾答案，並說明程式執行的流程。

隨堂練習：BMI 計算

請寫一個程式讓使用者輸入身高與體重，顯示BMI 值與肥胖程度。

BMI 等於體重（KG）除以身高（M）的平方，而BMI 與肥胖分類標準如下：

表 4-6 BMI 計算

BMI 值	肥胖分級
$BMI < 18$	體重過輕
$18 \leq BMI < 24$	體重正常
$24 \leq BMI < 27$	體重過重
$27 \leq BMI$	體重肥胖

```

#輸入身高體重資訊，計算BMI。
w = float(input('請輸入體重(KG)? ')) #Input
h = float(input('請輸入身高(M)? '))
bmi = w/(h*h)                          #Process
print('BMI為', bmi)                     #Output

# BMI分類
if bmi < 18:
    print('體重過輕')
elif bmi < 24:
    print('體重正常')
elif bmi < 27:
    print('體重過重')

```

```
else:
    print('體重肥胖')
```

課後作業：閏年判斷

設計程式允許輸入西元幾年，請求出該年是否是閏年。

閏年表示該年多一天。年份若為4的倍數稱做閏年，但若為100的倍數就不為閏年，且若為400倍數又是閏年。

```
year = int(input('請輸入年份? '))

if ((year % 400) == 0):
    print(year, "是閏年")
elif ((year % 100) == 0):
    print(year, "不是閏年")
elif ((year % 4) == 0):
    print(year, "是閏年")
else:
    print(year, "不是閏年")
```

4. 巢狀選擇

有些事情是在某個條件下，才需要再做下一步的條件判斷。也就是說，只有在第一層條件成立時（外層），才會執行第二層條件式（內層），內層條件式屬於外層條件式的一部份。

使用觀念

巢狀選擇結構為選擇結構內包含選擇結構，是一種有層次的選擇，將會依照程式的需要組合出不同的選擇結構。

- 巢狀選擇結構有好幾層，每一層的選擇都可以使用單向、雙向與多向選擇結構。
- 巢狀結構沒有標準程式語法，每個範例的程式語法可能都不相同。
- 巢狀選擇結構有無限可能執行的路徑或狀態。

範例：猜數字遊戲

使用者輸入的數不等於秘密數，則它會看這個數太低或太高，給使用者適當訊息。



```
# Guess the number.
# 兩個雙向選擇
secret_number = 83
n = input("Guess a number between 1 and 100: ")
n = int(n) # 文字轉數字

if n == secret_number:
    print("Guess Right!")
else:
    # 猜錯則檢查太高或太低
    if n > secret_number:
        print("Guess lower!") # 太高要往低猜
    else:
        print("Guess higher") # 太低要往高猜
print("Thank you for joining the game!")
```

課後作業: 分數與評語

寫一個程式若成績大於等於80分，評語為「非常好」；否則若成績大於等於60分，評語為「不錯喔」；否則評語為「要加油」。

```
# 方法（一）：單向選擇
score = int(input('請輸入一個成績？'))
if score >= 80:
    print('非常好')
if 80 > score >= 60: #前面要有80>才行
    print('不錯喔')
if score < 60:
    print('要加油')

# 方法（二）：多向選擇
score = int(input('請輸入一個成績？'))
if score >= 80:
    print('非常好')
elif score >= 60:
    print('不錯喔')
else:
    print('要加油')
```

```

# 方法（三）：雙向選擇+巢狀選擇
score = int(input('請輸入一個成績？'))
if score >= 80:
    print('非常好')
else:
    if score >= 60:
        print('不錯喔')
    else:
        print('要加油')

```

5. 條件迴圈

在Python，條件迴圈可以用**while**敘述式來建立。

```

# if 條件式到 while迴圈
## if 條件式
n = 5
if n < 10:
    print("n小於10!")

## while迴圈（一）
n = 5                #初始值
while n < 10:         #當n小於10印出
    print("n小於10!")

## while迴圈（二）
n = 5                #初始值
while n < 10:         #當n小於10印出
    print("n小於10!")
    print("我還在迴圈中!!") # ctrl+c 停掉

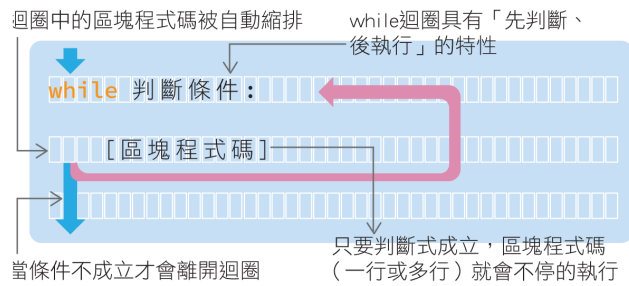
## while迴圈（三）
n = 5                #初始值
while n < 10:         #當n小於10印出
    print("n小於10!")
    print("我要出迴圈!!")
    n = 100          #變更條件
print("我出來了")

## while迴圈（四）
n = 5                #初始值start
while n < 10:         #判斷條件：當x小於10印出 stop
    print("n小於10!")
    print("我要出迴圈!!")
    n = n + 1         #變更條件step
print("我出來了")

```

基礎語法

while 指令後面所接測試條件，若為真時會不斷做迴圈內動作，直到測試條件的結果為假時跳出**while** 迴圈。



while 迴圈

while是指「當」判斷條件成立或為真值時，才會執行底下區塊的內容，執行完區塊後，再回while判斷是否條件繼續成立，若不成立，則離開迴圈，若成立就一直反覆執行區塊內容。

圖 4-3.5 while 迴圈語法說明

～所有條件迴圈的結構形式：虛擬程式碼

```
設定初始條件      # 初始條件
條件為True時：    # 判斷條件
    程式碼每次在迴圈內執行
    變更條件      # 變更條件
```

例如：從1到5逐一印出

```
# 從1到5逐一印出
n = 1 #初始條件start
while n < 5 : # 判斷條件stop
    print("變數n是", n)
    n = n + 1 # 更新條件step
print("Finished")
```

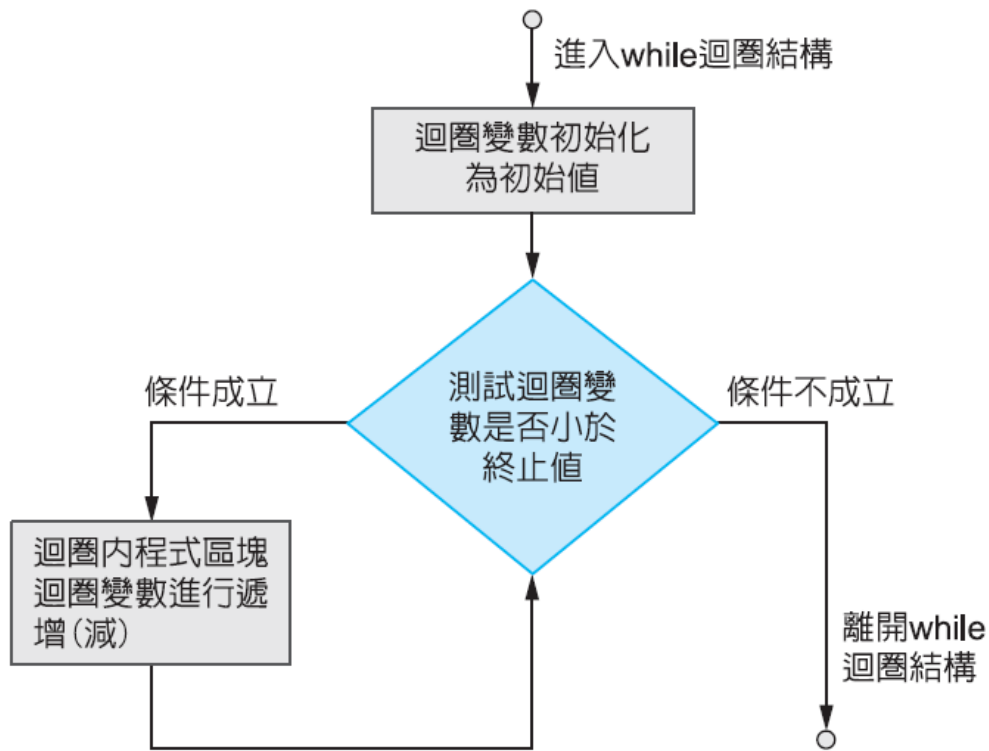


圖 A-14 流程圖

隨堂練習：印10次**hello**

```

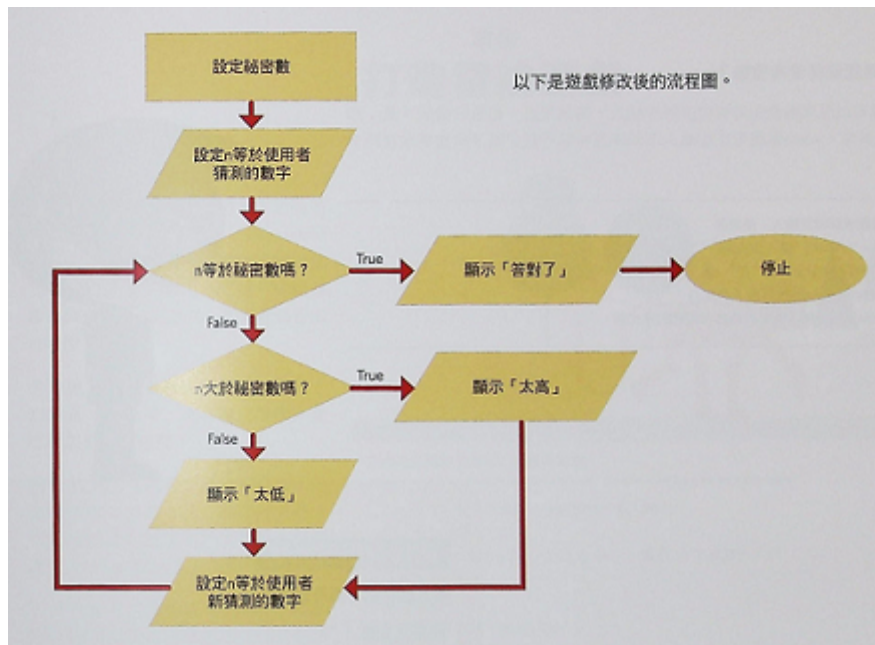
# 一般型迴圈
## 方法（一）：
n = 0 #初始條件
while n < 10: #判斷條件
    print("hello")
    n = n + 1 #更新條件

## 方法（二）：
n = 0 #初始條件
while n < 10: #判斷條件
    n = n + 1 #更新條件
    print("這是第", n, "次的hello")

## 方法（三）：
n = 0 #初始條件
left = 0 #初始條件
while n < 10: #判斷條件
    n = n + 1 #更新條件
    left = 10 - n
    print("這是第", n, "次的hello", "還有", left, "次機會")
  
```

課後作業：猜數字遊戲

在猜數字遊戲的例子，我們把初始條件設為使用者的第一個猜測。我們檢查的條件式猜測是否等於秘密數字。每次執行的程式碼會顯示太低或太高，最後變更條件式讓使用者輸入新的數，這樣要檢查的條件式也會變更。如果不變更條件式，不給使用者再猜一次的機會會怎麼樣呢？



```
secret_number = 83
n = input("猜1和100之間的秘密數")
n = int(n) # 將使用者輸入轉換為整數

while (n != secret_number):
    # 不等於secret_number，猜錯因此檢查太低或太高。
    if n > secret_number:
        print("你猜的太高，猜低點")
    else:
        print("你猜的太低，猜高點")
    # 請使用者再猜一次
    n = int(input("再猜一次"))

print("你猜對了！")
```

Summary

if 運用十分廣泛，常見的有：

- (1)單向選擇結構
- (2)雙向選擇結構
- (3)多向選擇結構
- (4)巢狀選擇結構

條件重複迴圈則用while

補充說明：**while**迴圈三大功能 - 列印、計數、加總

```
# 功能一：print列印1
i = 1
```

```

while i <= 10:
    print(i)
    i += 1
print("while loop is done.")

# 功能一：print列印2
x = int(input("請輸入起始的正整數:"))
y = int(input("請輸入終止的正整數"))
i = x
while i <= y:
    print(i)
    i += 1

# 功能二：計數一
i = 1
counter = 0
while i <= 10:
    counter += 1
    i += 1
print(counter)

# 功能二：計數二
i = 1
counter = 0
while i <= 10:
    if i % 2 == 0:
        counter += 1
    i += 1
print(counter)

# 功能三：加總
i = 1
summation = 0
while i <= 100:
    summation += i
    i += 1
print(summation)

```

練習題：電影評價

```

# 單向選擇 if條件句

movie_title = input('請輸入電影的名稱:')
movie_rating = input('請輸入電影的評等:')
movie_rating = float(movie_rating)

if movie_rating > 7:
    print("{}的評分為 {} 分值得去看!".format(movie_title,
movie_rating))

```

```

## 若輸入電影的評分高，就會印出；若輸入電影的評分低，就不會印出。

## 二合一的單向選擇

movie_title = input('請輸入電影的名稱:')
movie_rating = input('請輸入電影的評等:')
movie_rating = float(movie_rating)

if movie_rating > 7:
    print("{}的評分為 {} 分值得去看!".format(movie_title,
movie_rating))
if movie_rating <= 7:
    print("{}的評分為 {} 分不值得去看，浪費時間!".format(movie_title,
movie_rating))

# 雙向選擇

movie_title = input('請輸入電影的名稱:')
movie_rating = input('請輸入電影的評等:')
movie_rating = float(movie_rating)

if movie_rating > 7:
    print("{}的評分為 {} 分值得去看!".format(movie_title,
movie_rating))
else:
    print("{}的評分為 {} 分不值得去看，浪費時間!".format(movie_title,
movie_rating))

## 無論輸入電影評分高或低，都會印出對應訊息。

# 多向選擇
movie_title = input("請輸入電影名稱：")
movie_rating = input("請輸入電影評分：")
movie_rating = float(movie_rating)

if movie_rating > 7:
    print("去電影院看{}".format(movie_title))
elif movie_rating > 6:
    print("在家裡看{}".format(movie_title))
else:
    print("不要看{}".format(movie_title))

```

練習題：依照身份證字號，排隊領口罩

```

# 單向選擇
user_int = input("請輸入一個正整數：")
user_int = int(user_int)
if user_int % 2 == 0:
    ans = "偶數"
if user_int % 2 == 1:
    ans = "奇數"

```

```

print(ans)

## 雙向選擇
user_int = input("請輸入一個正整數：")
user_int = int(user_int)
if user_int % 2 == 0:
    ans = "偶數"
else:
    ans = "奇數"
print(ans)

## 雙向選擇
id_last_digit = input("請輸入身分證字號的尾數：")
id_last_digit = int(id_last_digit)
if id_last_digit % 2 == 0:
    ans = "星期二四六日領"
else:
    ans = "星期一至五日領"
print(ans)

## 雙向選擇
id_number = input("請輸入身分證字號：")
id_last_digit = id_number[-1]
id_last_digit = int(id_last_digit)
if id_last_digit % 2 == 0:
    ans = "星期二四六日領"
else:
    ans = "星期一至五日領"
print(ans)

## 雙向選擇
id_number = input("請輸入身分證字號：")
id_second_digit = id_number[1]
if id_second_digit == '1':
    gender = "Male"
else:
    gender = "Female"
print(gender)

```

練習題：冒險遊戲

我們可以用if敘述決定程式碼，創立一個冒險遊戲。這個遊戲讓你透過故事，挑選自己要做什麼，不同的決定將導致不同的結局。為了幫助你開始，可按照以下說明進行：

～創立一個名為「選擇你的冒險」的Python檔案並儲存它。

～使用下面的程式碼開始定義遊戲：

```

name = " Your name here"

```

```

print(f"Welcome to {name}'s Choose Your Own Adventure game! As you
follow the story, you will be presented with choices that decide
your fate. Take care and choose wisely! Let's begin.")

print(f"You find yourself in a dark room with 2 doors. The first
door is red, the second is white!")

door_choice = input("which door do you want to choose? red = red
door or white = white door")

if door_choice == "red":
    print("Great, you walk through the red door and
are now in the future!
You meet a scientist who gives you a mission of
helping him save the world!")

    choose_one = input("What do you want to do?
1 = Accept or 2 = Decline")
    if choice_one == "1":
        print("""______SUCCESS______
You will helped the scientist save the world!
In gratitude, the scientist builds a time machine
and sends you home! """)
    else:
        print("""______GAME OVER______
Too bad! You declined the sientist's offer and
now you are stuck in the future!""")

else:
    print("Great, you walked through the white door and
now you are in the past!
You meet a princess who asks you to go on a request.")

    quest_choice = input("Do you want to accept her offer      and
go on the quest, or do you want to stay where you      are? 1 =
Accept or 2 = Decline")

    if quest_choice == "1":
        print("The princess thanks you for accepting her
offer. You begin the quest.")
    else:
        print("""______GAME OVER ____
well, I guess your story ends here!""")

```