2.3 判斷式

在日常生活中，我們經常會遇到一些需要做決策的情況，然後再依決策結果從事不同的事件，例如：暑假到了，如果所有學科都及格的話，媽媽就提供經費讓自己與朋友出國旅遊；如果有某些科目當掉，暑假就要到校重修了！程式設計也一樣，常會依不同情況進行不同處理方式，這就是「判斷式」。

2.3.1 單向判斷式（if…）

「if…」為單向判斷式，是 if 指令中最簡單的型態，語法為：

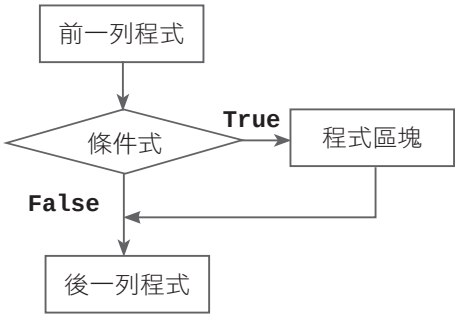
```
if 條件式：
    程式區塊
```

當條件式為 True 時，就會執行程式區塊的敘述；當條件式為 False 時，則不會執行程式區塊的敘述。

條件式可以是關係運算式，例如：「x > 2」；也可以是邏輯運算式，例如：「x > 2 or x < 5」，如果程式區塊只有一列程式碼，則可以合併為一列，直接寫成：

```
if 條件式：程式碼
```

以下是單向判斷式的流程圖：



Python 程式碼縮排格式

大部分語言如 C、Java 等，多是以一對大括號「{}」來表示程式區塊，例如：

```
if(score>=60) {
   grade = "及格";
}
sum = sum + score
```

程式區塊

下一列程式

Python 語言以冒號「:」及縮排來表示程式區塊，縮排為 1 個 Tab 鍵或 4 個空白鍵，例如：

```
if(score>=60):
   grade = "及格"
sum = sum + score
```

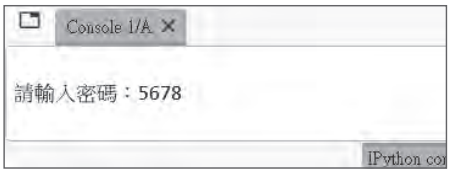
程式區塊

下一列程式

Tab 鍵或 4 個空白鍵

範例：密碼輸入判斷

讓使用者輸入密碼，如果輸入的密碼正確 (1234)，會顯示「歡迎光臨！」；如果輸入的密碼錯誤，則不會顯示歡迎訊息。



程式碼：password1.py

```
1 pw = input("請輸入密碼：")
2 if pw=="1234":
3     print("歡迎光臨！")
```

程式說明

- 2-3　預設密碼為「1234」，若輸入的密碼正確就執行第 3 列程式列印訊息，若輸入的密碼錯誤就結束程式。

因為此處 if 程式區塊的程式碼只有一列，所以第 2-3 列可改寫為：

```
if pw=="1234" : print("歡迎光臨！")
```

2.3.2 雙向判斷式（if…else）

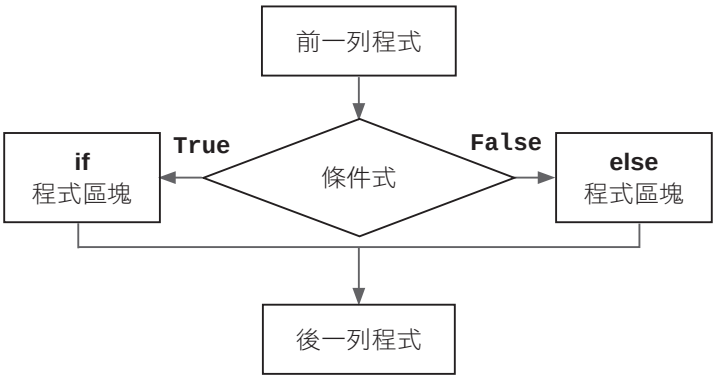
感覺上「if」語法並不完整，因為如果條件式成立就執行程式區塊內的內容，如果條件式不成立也應該做某些事來告知使用者。例如密碼驗證時，若密碼錯誤應顯示訊息告知使用者，此時就可使用「if…else…」雙向判斷式。

「if…else…」為雙向判斷式，語法為：

```
if 條件式:
    程式區塊一
else:
    程式區塊二
```

當條件式為 True 時，會執行 if 後的程式區塊一；當條件式為 False 時，會執行 else 後的程式區塊二，程式區塊中可以是一列或多列程式碼，如果程式區塊中的程式碼只有一列，可以合併為一列。

以下是雙向選擇流程控制的流程圖：



範例：進階密碼判斷

讓使用者輸入密碼，如果輸入的密碼正確 (1234)，會顯示「歡迎光臨！」；如果輸入的密碼錯誤，則會顯示密碼錯誤訊息。

程式碼：password2.py

```
1  pw = input("請輸入密碼：")
2  if pw=="1234":
3      print("歡迎光臨！")
4  else:
5      print("密碼錯誤！")
```

程式說明

- 2-3　若輸入的密碼正確，就執行第 3 列程式，顯示歡迎訊息。
- 4-5　若輸入的密碼錯誤，就執行第 5 列程式，顯示密碼錯誤訊息。注意第 4 列要由開頭處輸入「`else:`」。

2.3.3 多向判斷式（if…elif…else）

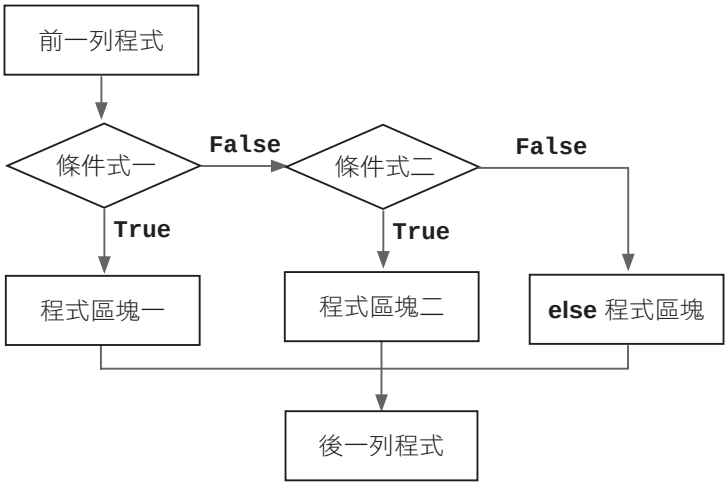
事實上，大部分人們所遇到複雜的情況，並不是一個條件就能解決，例如處理學生的成績，不是單純的及格與否，及格者還需依其分數高低給予許多等第（優、甲、乙等），這時就是多向判斷式「if…elif…else」的使用時機。

「if…elif…else」可在多項條件式中，擇一選取，如果條件式為 True 時，就執行相對應的程式區塊，如果所有條件式都是 False，則執行 else 後的程式區塊；若省略 else 敘述，則條件式都是 False 時，將不執行任何程式區塊。「if…elif…else」的語法為：

```
if 條件式一：
    程式區塊一
elif 條件式二：
    程式區塊二
elif 條件式三：
    ……
else:
    程式區塊 else
```

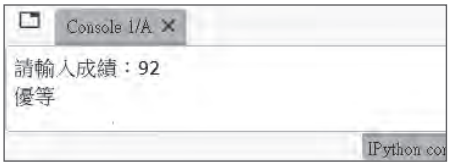
如果「條件式一」為 True 時，執行程式區塊一，然後跳離 if 多項條件式；「條件式一」為 False 時，則繼續檢查「條件式二」，若「條件式二」為 True 時，執行程式區塊二，其餘依此類推。如果所有的條件式都是 False，則執行 else 後的程式區塊。

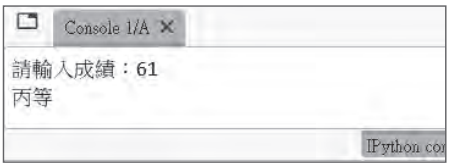
以下是多向判斷式流程控制的流程圖 (以設定兩個條件式為例) :



範例：判斷成績等第

讓使用者輸入成績，若成績在 90 分以上就顯示「優等」， 80-89 分顯示「甲等」，70-79 分顯示「乙等」，60-69 分顯示「丙等」，60 分以下顯示「丁等」。





程式碼：**grade.py**

```
1 score = input(" 請輸入成績：")
2 if(int(score) >= 90):
3     print(" 優等 ")
4 elif(int(score) >= 80):
5     print(" 甲等 ")
6 elif(int(score) >= 70):
7     print(" 乙等 ")
```

```
 8 elif(int(score) >= 60):
 9     print("丙等")
10 else:
11     print("丁等")
```

程式說明

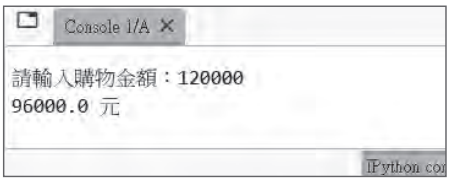
- 2-3　若輸入的成績在 90 分以上就列印「優等」。
- 4-5　若輸入的成績在 80 分以上就列印「甲等」。
- 6-7　若輸入的成績在 70 分以上就列印「乙等」。
- 8-9　若輸入的成績在 60 分以上就列印「丙等」。
- 10-11　若前面條件都不成立表示分數在 60 分以下，列印「丁等」。

2.3.4 巢狀判斷式

在判斷式 (if…elif…else) 之內可以包含判斷式，稱為巢狀判斷式。系統並未規定巢狀判斷式的層數，要加多少層判斷式都可以，但層數太多會降低程式可讀性，而且維護較困難。

範例：百貨公司折扣戰

讓顧客輸入購買金額，若金額在 100000 元以上就打八折，金額在 50000 元以上就打八五折，金額在 30000 元以上就打九折，金額在 10000 元以上就打九五折。



14.2 OpenCV 圖形處理：移動偵測

OpenCV 除了具有強大的繪圖能力及臉部辨識功能外，也可以進行許多圖形處理，例如彩色圖形灰階化、對圖形模糊處理、對兩張圖片進行運算、尋找圖片輪廓等，我們可以利用這些圖形處理來進行移動偵測。

14.2.1 移動偵測的原理

「移動偵測」是指在影片中偵測是否有移動的物體。為什麼要做移動偵測呢？使用攝影機對環境進行監控時，絕大部分時間的攝影畫面都是靜止畫面，如果每一個畫面都儲存，不但需耗費大量讀寫資源，更需要龐大儲存空間。較理想的方式為對攝影畫面進行移動偵測，當偵測到畫面有移動物體時才進行錄影，這樣既不會錯失重要畫面，也盡可能節省儲存資源。

我們看到的攝影機畫面其實是一張張靜態影像組成的，利用眼睛的視覺暫留作用而形成動態影片。移動偵測的原理很簡單，只要比對影片中連續兩張靜態影像是否完全相同，若完全相同就表示影片中沒有物體移動；若前後兩張靜態影像有部分差異，就表示有物體移動了！

cv2.absdiff：矩陣相減

OpenCV 的 absdiff 方法其功能是取得兩個 Numpy 陣列差的絕對值，就是將陣列中每個對應的元素相減再取絕對值。cv2.absdiff 的語法為：

```
cv2.absdiff( 陣列 1, 陣列 2)
```

例如：

```
A = np.array([1, 2, 3]
B = np.array([3, 2, 1]
C = cv2.absdiff(A, B)  # [2, 0, 2]
```

如果是兩個完全相同的陣列，absdiff 的結果陣列其元素值全部為 0，例如：

```
A = np.array([1, 2, 3]
B = np.array([1, 2, 3]
C = cv2.absdiff(A, B)  # [0, 0, 0]
```

OpenCV 讀入的圖片就是 Numpy 陣列，陣列元素「0」表示黑色，也就是兩張完全相同的圖片在進行 absdiff 運算後，會變成全部黑色的圖片：

cv2.cvtColor：圖片顏色轉換

使用 OpenCV 的 absdiff 方法對圖片進行減法運算時，圖片必須是灰階圖片，而一般攝影機拍攝的畫面為彩色圖片，需先將其轉換為灰階圖片才能利用 absdiff 運算。OpenCV 的 cvtColor 方法可對圖片的顏色進行轉換，而將彩色圖片轉換為灰階的語法為：

```
cv2.cvtColor( 圖片檔路徑 , cv2.COLOR_BGR2GRAY)
```

例如將 media 資料夾中 <test.jpg> 彩色圖片轉為灰階圖片：

```
cv2.cvtColor('media/test.jpg', cv2.COLOR_BGR2GRAY)
```

cv2.findContours：尋找圖片輪廓

需要解決的另一個問題是，如何判斷經過減法運算後的圖片是全部黑色的圖片呢？一張圖片的陣列元素數量非常龐大，以一張 640X480 灰階圖片為例，就有 640X480=307200 個元素，不太可能逐一檢視其元素值是否皆為 0。OpenCV 的 findContours 方法可取得圖片的輪廓，如果是全黑圖片，表示圖片中沒有任何物體，則輪廓數量為 0。

OpenCV 的 findContours 方法其語法為：

```
輪廓坐標 , 輪廓階層 = cv2.findContours( 圖片 , 偵測模式 , 輪廓算法 )
```

「偵測模式」有四種：

- **cv2.RETR_EXTERNAL**：只偵測輪廓外緣，這是最常用的模式。
- **cv2.RETR_LIST**：偵測輪廓時不建立等級關係。
- **cv2.RETR_CCOMP**：偵測輪廓時建立兩個等級關係。
- **cv2.RETR_TREE**：偵測輪廓時建立樹狀等級關係。

常用的「輪廓算法」有兩種：

- **cv2.CHAIN_APPROX_NONE**：儲存所有輪廓點。
- **cv2.CHAIN_APPROX_SIMPLE**：壓縮水平、垂直及對角線方向元素，只儲存該方向的終點，例如矩形只儲存四個點。此算法速度較快，是較常用的算法。

例如以 contours 及 hierarchy 做為接收變數偵測 <test.jpg> 圖片的輪廓：(<test.jpg> 必須為灰階圖片)

```
contours, hierarchy = cv2.findContours("test.jpg", cv2.RETR_EXTERNAL,
  cv2.CHAIN_APPROX_SIMPLE)
```

圖片輪廓資訊存於 findContours 方法的第一個傳回值中。圖片輪廓資訊是一個串列，每一個輪廓即為一個元素，如果傳回值是空串列就表示為全黑圖片，例如：

```
if len(contours) > 0:
    print('這不是全黑圖片！')
else:
    print('這是全黑圖片！'))
```

如此就可以進行移動偵測了！但事實並不是如此完美。下面範例中，<testCopy.jpg> 是由 <test.jpg> 複製得到，故兩張圖片完全相同；<testNext.jpg> 則是影片中 <test.jpg> 畫面的下一個畫片，兩張圖片看起來完全相同。

程式碼：**absdiff.py**

```
import cv2

def samePicture(pic1, pic2):
    img1 = cv2.imread(pic1)
    img2 = cv2.imread(pic2)
    gray1 = cv2.cvtColor(img1, cv2.COLOR_BGR2GRAY)
    gray2 = cv2.cvtColor(img2, cv2.COLOR_BGR2GRAY)
```

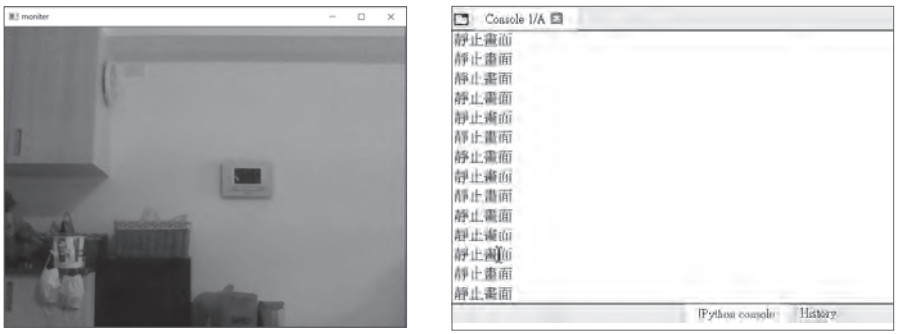
14.3 實戰：智慧監控系統

現在科技發達，許多科技產品的價格變得十分親民，新台幣幾百元即可買到影像品質還不錯的攝影機了！攝影機搭配本專題的智慧監控系統，就可用極少的預算實現功能強大的監控功能。

本專題撰寫的智慧監控系統會對攝影畫面進行移動偵測，只有在有物體進入攝影畫面時才會錄製影片，可大幅減少錄製及儲存資源，同時會以 LINE 傳送訊息及畫面給監控者。錄製的影片右上角會加上錄製時間戳記，讓使用者知道精確時間。

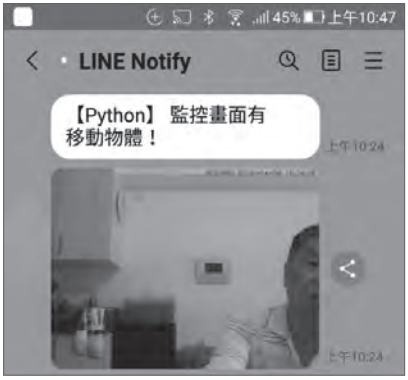
14.3.1 應用程式總覽

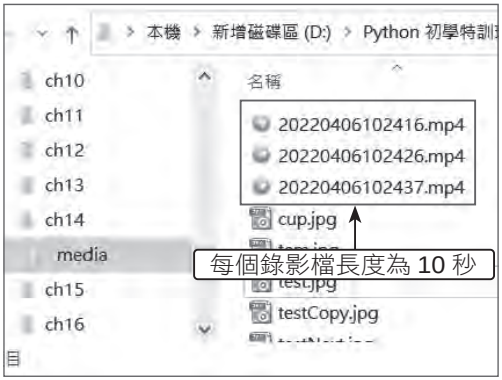
開始執行程式時將攝影鏡頭對著靜態環境，Console 會不斷顯示「靜止畫面」。(<smartMonitor.py>)



當有物體進入攝影鏡頭後會立刻啟動錄影，Console 會不斷顯示「錄影中！」。影片右上角會打印錄影時間戳記。

2 秒鐘後會以 LINE Notify 發送文字及圖片訊息告知監控者。錄影時則會以開始錄影的時間做為檔案名稱，方便監控者查詢，每段錄影時間長度為 10 秒鐘。





設定 2 秒鐘後發送 LINE 訊息是因為偵測到物體進入攝影鏡頭時，畫面僅有進入物體極少部分影像，所以等 2 秒鐘物體完全進入鏡頭時才傳送文字與圖片訊息。此處為了節省測試時間，將每段錄影時間長度設定為 10 秒鐘，實際應用時，可視需要設定錄影時間長度，例如 60 秒。

14.3.2 智慧監控系統程式碼

智慧監控系統結合移動偵測、攝影機錄影及 LINE Notify 發送訊息而成。

程式碼：**smartMonitor.py**

```
import cv2
import time
import requests

tinterval = 10  # 正式使用改為 60
timer = False
fourcc = cv2.VideoWriter_fourcc(*'MP4V')
token = ' 你的 LINE Notify 權杖 '  # 權杖
headers = {"Authorization": "Bearer " + token}
msg = ' 監控畫面有移動物體！ '
fcount = 0
prev = None
capture = cv2.VideoCapture(0)
while capture.isOpened():
    success, image = capture.read()
```