## 第 0 講　準備 Excel 2013 的 VBA 開發環境

- 在「檔案」/「選項」/「自訂功能區」內的「主要索引標籤」頁面中，勾選「開發人員」，Excel 功能區即新增「開發人員」索引標籤。

- 「開發人員」索引標籤內有「程式碼」、「增益集」、「控制項」、「XML」及「修改」等功能頁面，提供編寫 Excel VBA 程式及執行「增益集」的各項功能按鈕。

- 在 Excel 內，「巨集」或「VBA 程式」這兩個名稱涵意相同，可交互使用。

- 內建「巨集」或「VBA 程式」功能的活頁簿檔案，存檔類型必須選擇「Excel 啟用巨集的活頁簿」(附檔名為 .xlsm)或「Excel 啟用巨集的範本」(附檔名為 .xltm)。

➢ 在 Excel 2013 內錄製巨集或以 VBA 建立巨集的方法與之前版本大致相同。本講義第一、二、三、四、五、六講乃按照 Excel 較早版本撰寫，其中大部分內容在 Excel 2013 下仍屬有效。

# 第一講　VBA 入門

## 錄製巨集

(略)

## 用 VBE 建立巨集

建立巨集的方式：

1.選擇「工具」、「巨集」、「巨集」，在「巨集名稱」文字方塊中輸入巨集名稱(例如：ChgMode)，並選定儲存位置(從「所有開啓的活頁簿」、「現用活頁簿」、「其他已開啓之個別活頁簿(可能有多個)」三種選項中選一種)，按「建立」，excel 會自動切換至 VBE 工作環境，並新增一個以 ChgMode 爲名稱的程序(procedure)。模組視窗中已有下列內容存在：

```
Sub ChgMode()

End Sub
```

2.選擇「工具」、「巨集」、「Visual Basic 編輯器」，進入 VBE 編輯環境中。在「專案視窗」中指定巨集所要儲存的位置(例如：VBAProject(Book1))。接著，選擇「插入」、「模組」，即可爲 VBAProject(Book1)新增一個模組(Module1)，並出現一個空白的編輯視窗(即：模組視窗)。需自行鍵入上述「Sub……End Sub」之內容。

## 巨集對話方塊

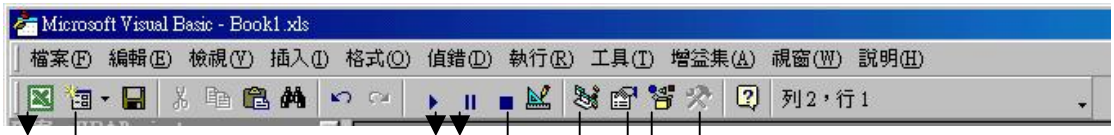
有「執行」、「取消」、「逐步執行、「編輯」、「建立」、「刪除」及「選項」等按鈕。其中，「逐步執行」通常用在巨集程序內容的除錯；「編輯」可從複雜的程式碼中找到特定巨集的程式碼位置以進行修改；「選項」按鈕可用以定義巨集之快速鍵。

## 指定巨集

任何快取圖案均可用以指定巨集：按滑鼠右鍵顯示下拉式選單，「新增文字」可在圖案上編輯巨集名稱，「指定巨集」則用以指定特定巨集到圖案上。

# 第二講　VBA 環境---VBE 編輯器

## 功能表及工具列




切換回 Excel　　執行及中斷執行

可插入：自訂表單、模組、物件類別模組、程序

重設：當程序執行發生錯誤時，此案鈕可以解除偵錯狀態

瀏覽物件：在撰寫巨集時可利用「瀏覽物件視窗」查詢特定物件的屬性、方法

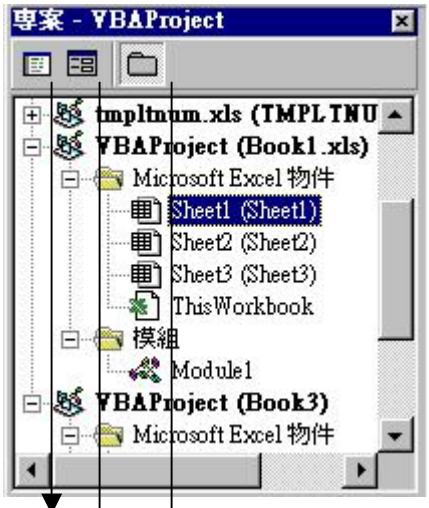
工具箱：開啓工具箱工具列，其中包含許多有用的表單控制項

切換至「專案總管」視窗

切換至「屬性」視窗

### 專案總管視窗

包含應用系統功能引用的檔案及已開啓的活頁簿檔案，每一個檔案都被視爲一個專案。活頁簿檔案是以「VBAProject(活頁簿名稱)」顯示，打開+號後可以看到活頁簿(this workbook)、工作表(sheet)、模組(module)、表單(form)等物件。




檢視程式碼：可切換至相關物件之程式碼視窗

檢視物件：可切換至所選取之物件，例如，上述畫面下，可切換至 Excel Book1 之 sheet1 工作表。

切換資料夾：沒有明顯功能

### 模組視窗

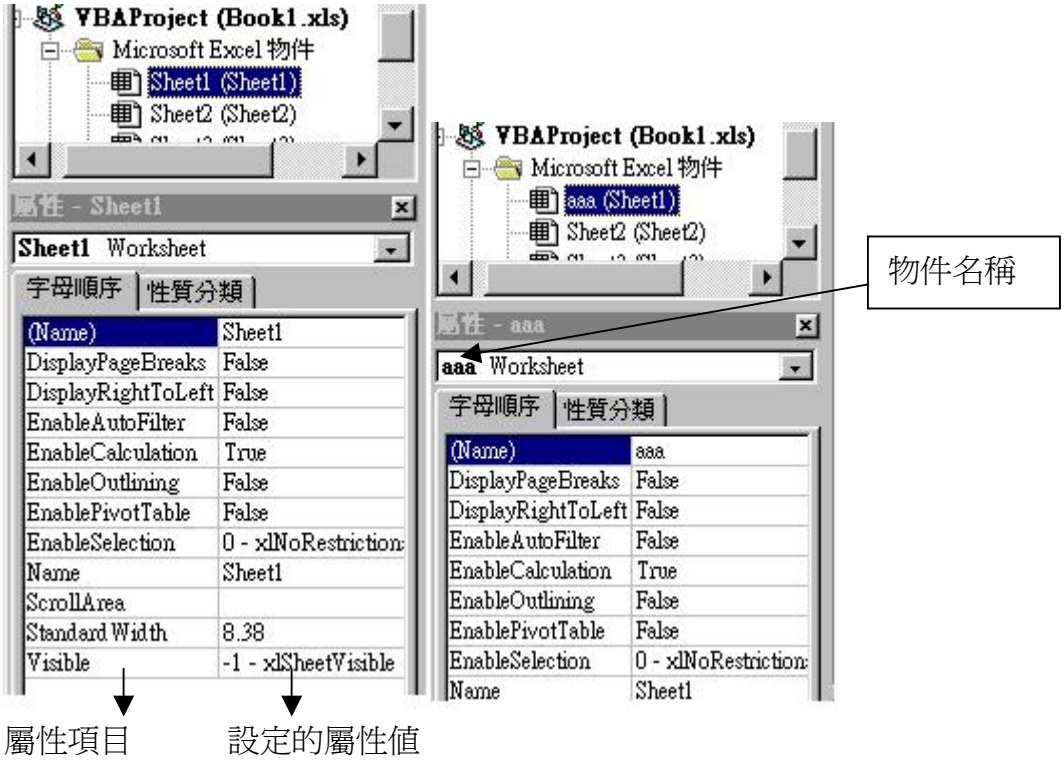
即通稱之「程式碼視窗」。在特定的活頁簿專案中，選擇「插入」、「模組」，即可爲該活頁簿新增模組視窗。


物件標示區：如開啓的是單一模組，則只有(一般)選項；若開啓的是某一特別物件，則會列出(一般)及所屬的物件名稱兩個選項，選擇(一般)代表撰寫一般程序，選擇物件名稱代表撰寫物件的事件處理程序。

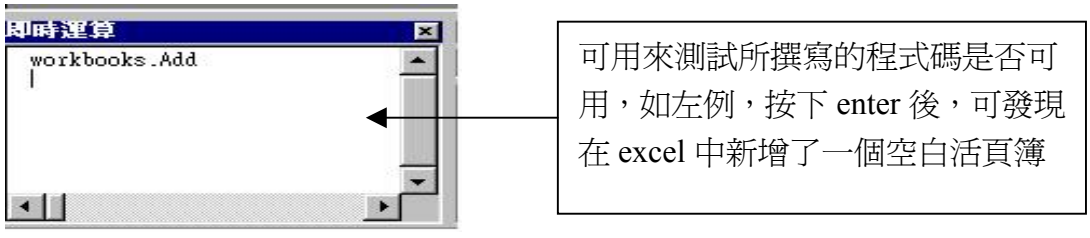
巨集標示區：若物件標示區上設定爲(一般)，則本區列示模組內所有的巨集名稱；若物件標示區設定爲特定物件名稱，則本區列示該物件所提供的事件程序。

**屬性視窗**

用來列示及設定所指定的物件之相關屬性項目

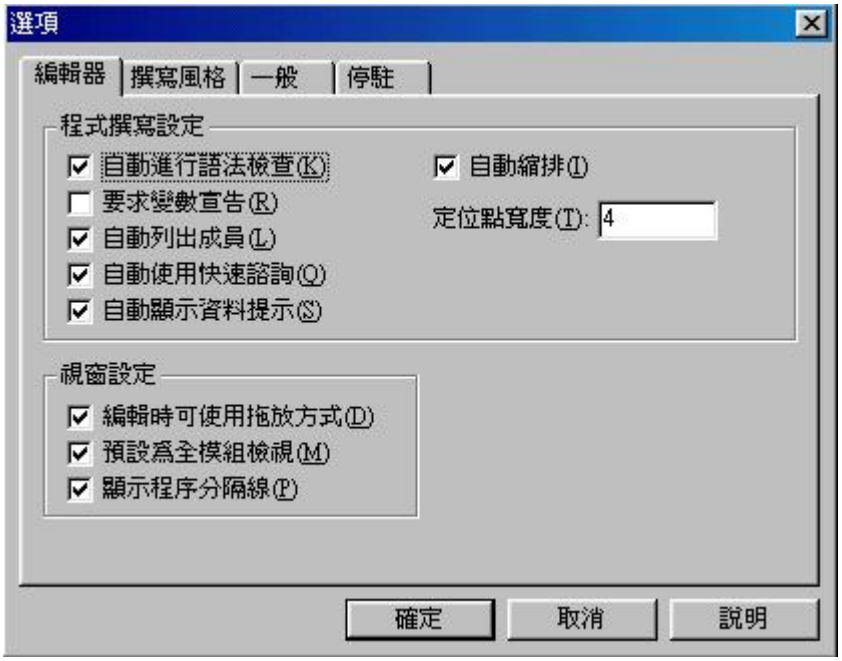



**即時運算視窗**



## 自定工作環境

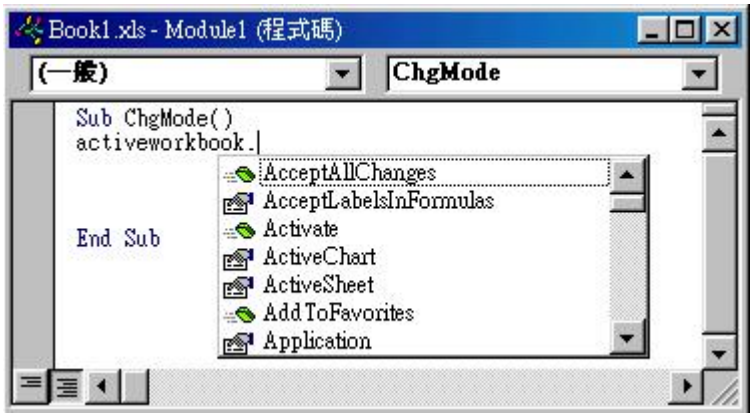
在 VBE 下，選擇「工具」、「選項」



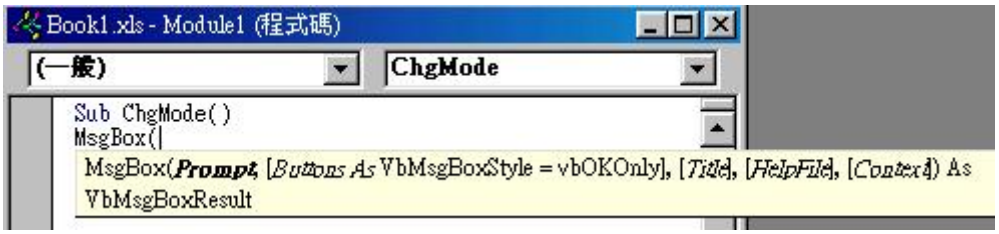

自動進行語法檢查：當程式碼寫錯卻按下 enter 時，會出現警告視窗，提示正確的內容，並把錯誤的程式碼以紅色標示。本項若未選取，則僅會出現紅色標示，而不會出現警告視窗。

要求變數宣告：核取本項，則在新增一個模組時，會加入 Option Explicit 陳述式，表示該模組中的所有變數必須先經過宣告才能使用，否則程式將無法辨識。

自動列出成員：在輸入程式碼時，excel 會提供相關資訊的清單供選用。例如：

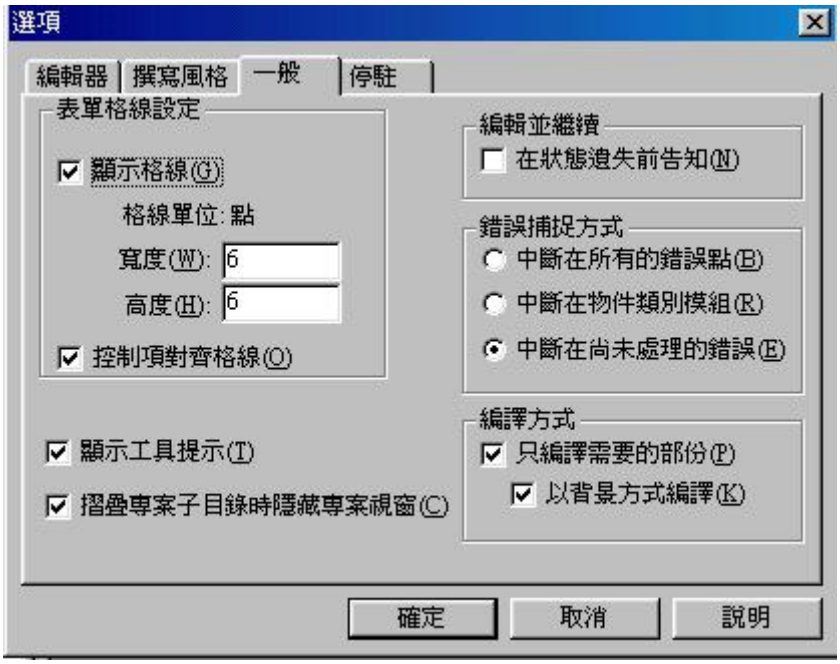



自動使用快速諮詢：在程式中使用到函數時(如 MsgBox())，會提供該函數的相關資訊，如使用的參數及是否傳回數值。例：

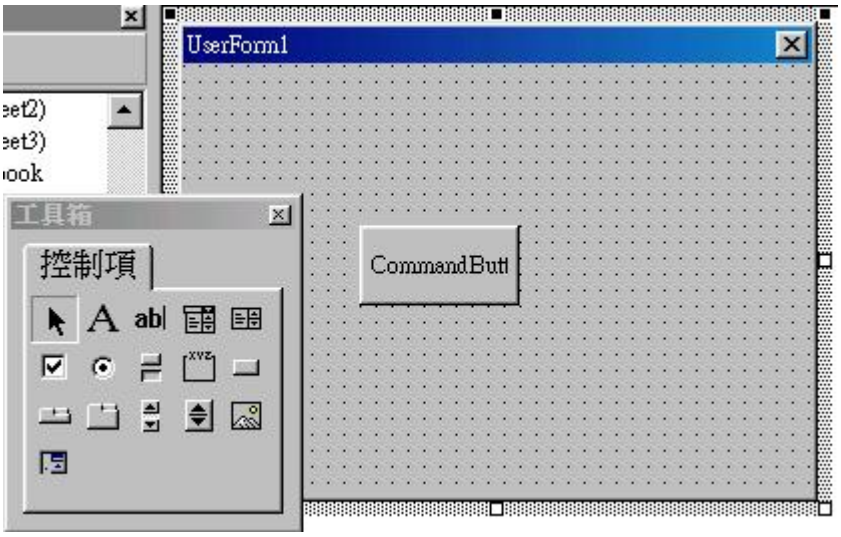


自動顯示資料提示：在程式進入除錯階段時，可以顯示出滑鼠游標所在位置的變數值。此功能只能在中斷模式下使用。
自動縮排：Sub 後之第一行程式碼若縮排，則之後的程式碼均自動縮排至同一位置。
定位點寬度：設定當使用者按下 Tab 鍵時跳格的寬度，範圍從 1~32，內定爲 4。
編輯時可使用拖放方式：(略)
預設爲全模組檢視：可將模組中所撰寫的所有程序顯示在同一個視窗內。若未勾選，則程式碼視窗一次只列示一個程序。
顯示程序分格線：可將同一模組中不同的程序之間作區隔。


可顯示程式碼文字的字型、大小、顏色(前景)、背景、邊界指示區及其顏色



顯示格線及控制項對其格線：例

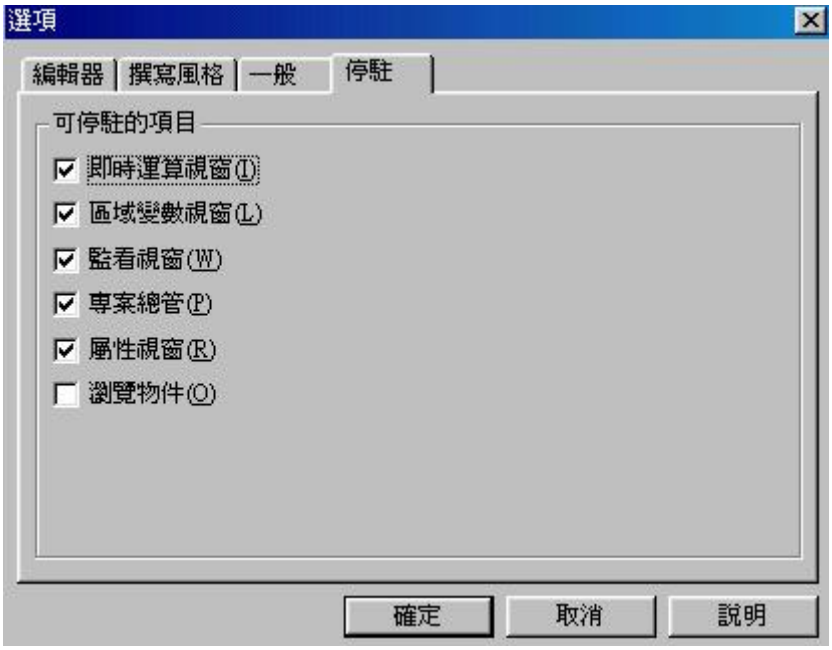



顯示工具提示：(略)

只編譯需要的部分：勾選本項，則專案在開始執行前不必完全編譯，只編譯需要的程式碼，即可執行。

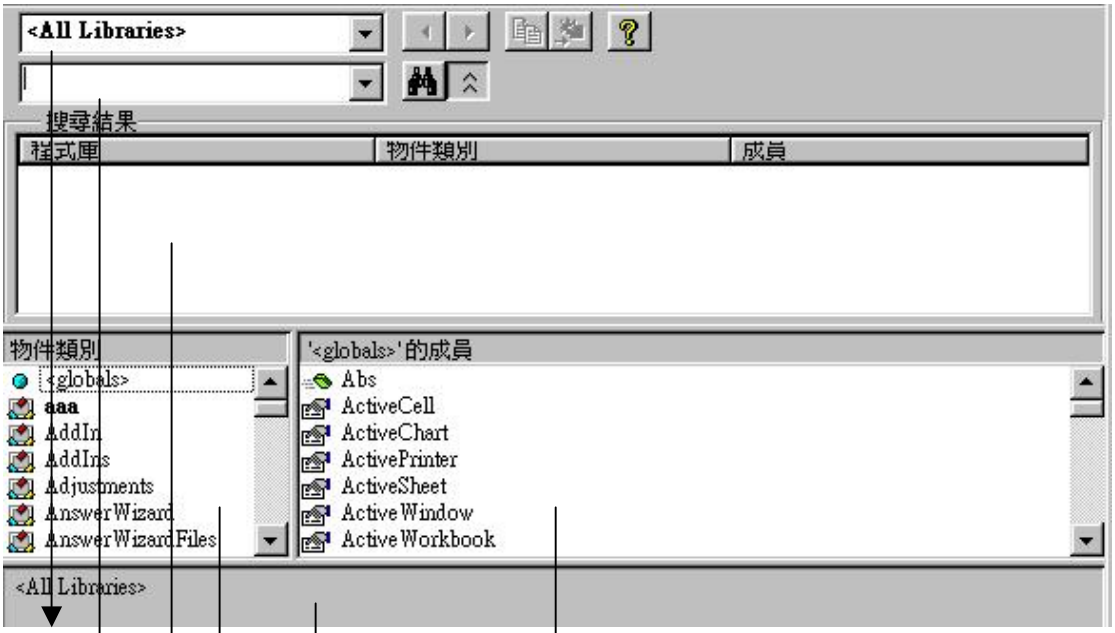
以背景方式編譯：可在執行階段內利用閒置時間完成專案的編譯。勾選本項前需先勾選「只編譯需要的部分」。

錯誤捕捉方式：(略)




**瀏覽物件視窗**

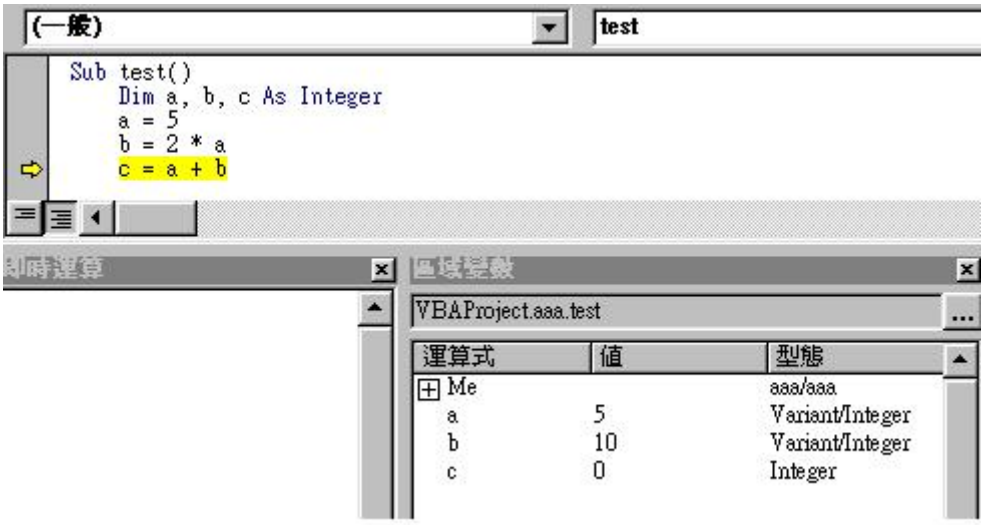
可以利用此視窗了解所使用的物件在物件層級中所屬的階層，以及物件提供的屬性、方法及事件程序。選擇「檢視」、「瀏覽物件」，即可開啓物件瀏覽視窗：




程式庫：列示 excel 中可用的物件庫
搜尋文字：鍵入文字以快速搜尋
搜尋結果：列示上述之搜尋結果
物件類別清單：列示所指定之物件庫中包含的物件
成員清單：列示所檢視之物件可用的屬性、事件及方法
細部說明：列示使用者所選取的成員清單項目是屬於何種類型

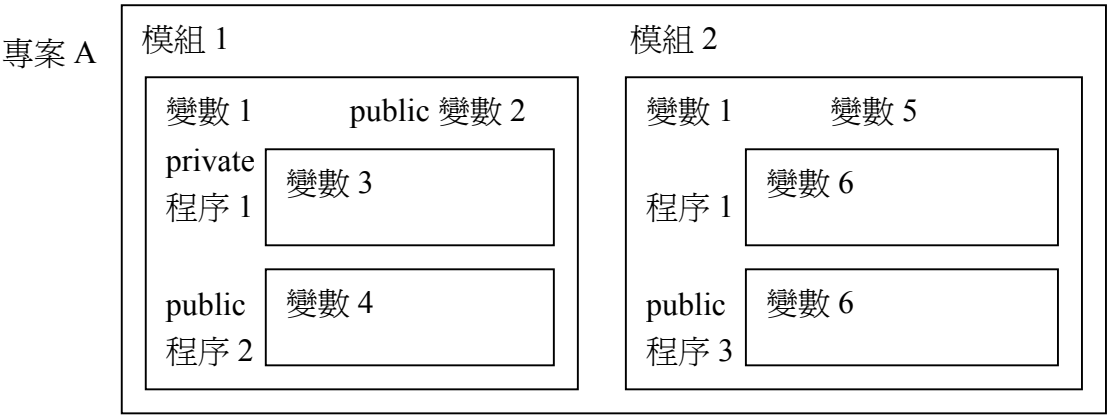
**區域變數視窗**

在撰寫程式時，若想知道所使用的變數在程式中的變化情況，可以利用本視窗來監看。在監看時可以以逐步執行(在程式碼中按滑鼠右鍵進行選取)或設定執行中斷點的方式查看巨集內所有變數的值。區域變數內的值是顯示巨集程序執行至「執行中斷位置」時各變數的狀態，而「執行中斷位置」所指的位置代表下一個執行的指令。例：



## 第三講

### 專案、模組、變數、程序




*一個專案中可以建立多個模組，一個模組中可以建立多個程序。

*變數 1、變數 2 爲模組 1 中的**全域變數**，變數 1、變數 5 爲模組 2 中的**全域變數**，可分別供該模組內之程序呼叫。不同模組中的 private 程序及 private 全域變數，可使用相同的名稱。

*變數 3、變數 4、變數 5、變數 6 均爲**區域變數**，只限於該程序中呼叫。不同程序中，可使用相同的區域變數名稱。

*程序及全域變數可以加上範圍設定值 public 或 private，若未加上則內定爲 private。有加上 public 的程序或全域變數，可供同一專案中其他模組的程序呼叫。例如，模組 2 中的程序 1 及程序 2 均可呼叫模組 1 中的 public 變數 2 及 public 程序 2。

### 程序

*VBA 中，有 Sub、Function、Property 等三種程序(procedure)。

*新增程序：在 VBE 中，選擇「插入」、「程序」

*靜態變數：當程序被呼叫時，程序內的靜態變數會沿用上次呼叫時的值，而不會將變數值清爲空白。要將個別區域變數設定爲靜態變數，需在宣告變數時在變數名稱之前加上保留字 Static。要將所有區域變數設定爲靜態變數，則需在 Sub 或 Function 程序前加上 Static。

**Sub 程序**

*Sub 程序是 VBA 中最常用到的程序。錄製巨集時，excel 即會自動插入一個模組，並將所錄製的巨集以一個 Sub 程序來表示。例：

```
[Public/Private] [Static] Sub name()

End Sub
```

*Sub 程序沒有傳回值。

**Function 程序**

*Function 程序各部分與 Sub 程序大致相同，唯一差別是如果有「傳回值」，使用者需在 Function 程序的宣告部分定義「傳回值」的型態。

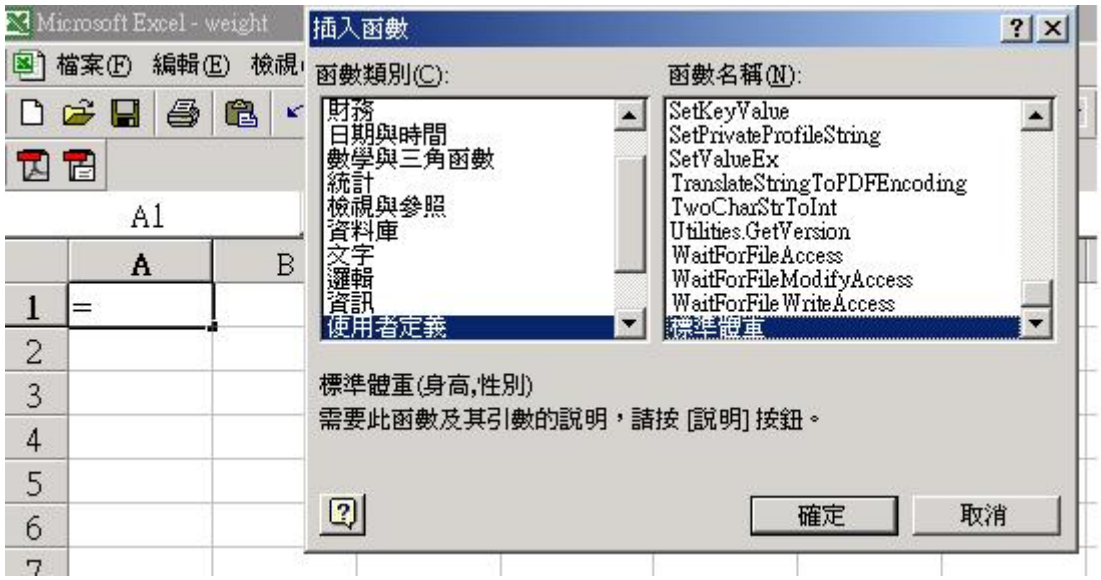
*Function 程序可將數值傳回給呼叫它的程序或引用它的公式中。

*Public Function 程序可以作爲公式選項，引用至工作表的儲存格中。通常我們會利用它來定義工作中常用的運算式。(註：Public 爲 Function 程序的預設值)

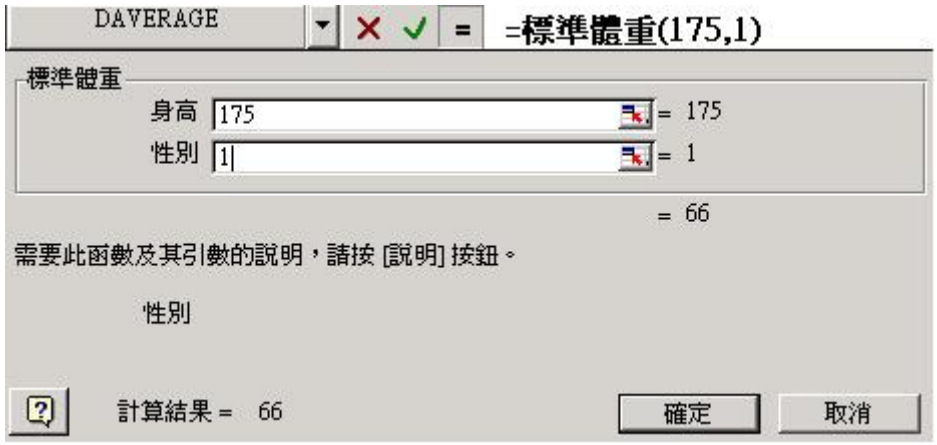
例：標準體重計算函數


*此 public function 會成爲一個「自訂函數」，可在 excel 工作表的任一儲存格中選擇「插入」、「函數」、「使用者定義」、「標準體重」，即可透過函數對話窗輸入身高、性別資料，算出標準體重。


在工作表中選用「標準體重」計算函數：



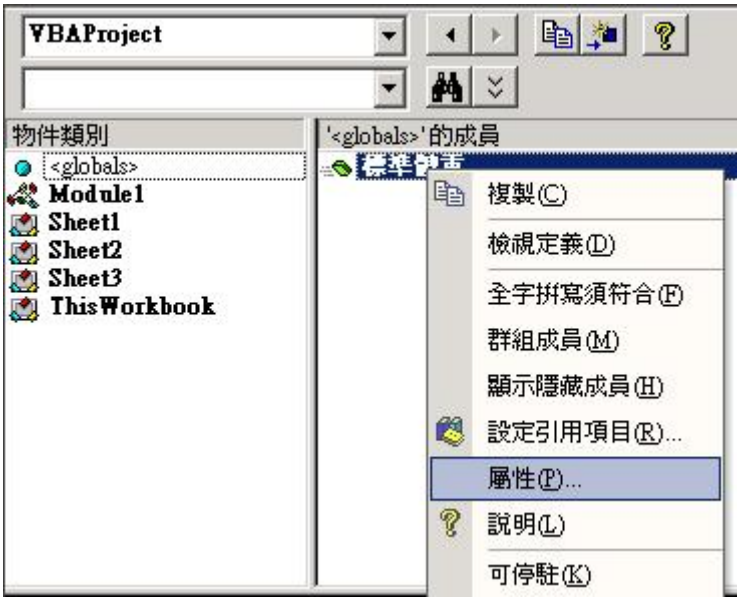

透過對話窗輸入身高、體重，得出標準體重：




*爲自訂函數加上說明：上面例子中，身高、體重的輸入方式並無說明，容易引起使用者誤會，所以需加上函數使用說明。

開啓物件瀏覽視窗，從物件庫中選擇 VBAProject：

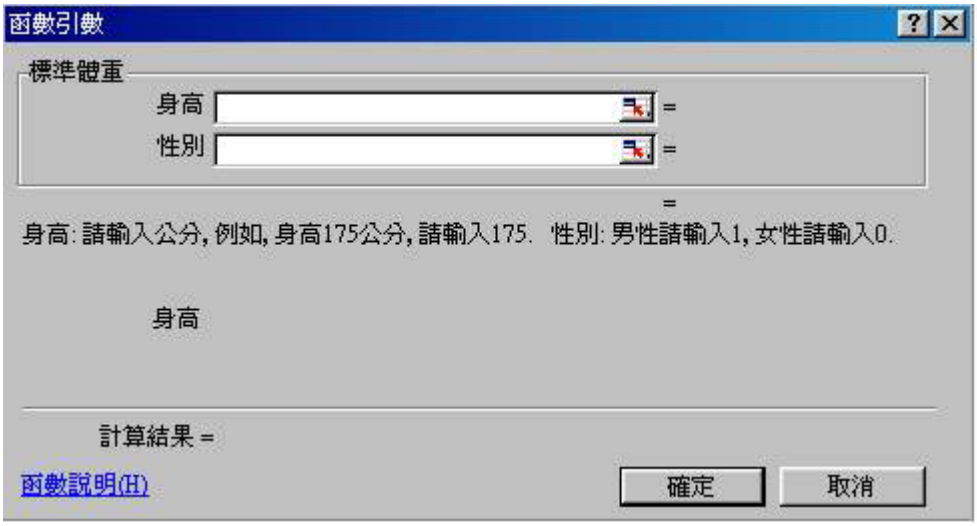

在「成員」中選擇「標準體重」，並按下滑鼠右鍵，選擇「屬性」




出現「成員選項」對話窗：


在「描述」中輸入「標準體重」函數的使用說明文字，按下「確定」:

在工作表中再次開啓「標準體重」函數時，就可以顯示使用說明：




**物件(Objects)、屬性(Properties)、方法(Methods)、事件(Events)**

*物件是由屬性、方法及事件三種元件所構成。屬性是指一個物件所包含的特性；方法則是物件可以執行的動作及行爲；事件是指發生在物件上的狀況，例如滑鼠按鍵是一個物件，在這個物件上能發生的事件共有五種，分別是按下(down)、釋放(up)、移動(move)、按一下(click)、按兩下(double click)，每一個事件發生時，滑鼠物件都有相對應的反應。要引用物件的屬性及方法，只要在物件型稱與屬性或方法名稱之間加上「.」即可。例：

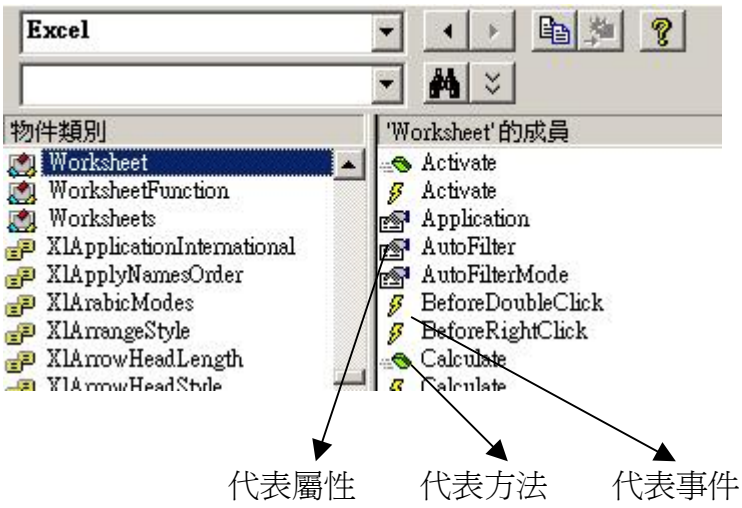
```
Sub ChangeName()
    Worksheets(“sheet1”).Name = ”工作表 1”
End Sub
```

上述的巨集，把 sheet1 工作表的 Name 屬性的值定爲「工作表 1」。除了寫巨集外，也可以直接透過屬性視窗來改變物件的特定屬性值。

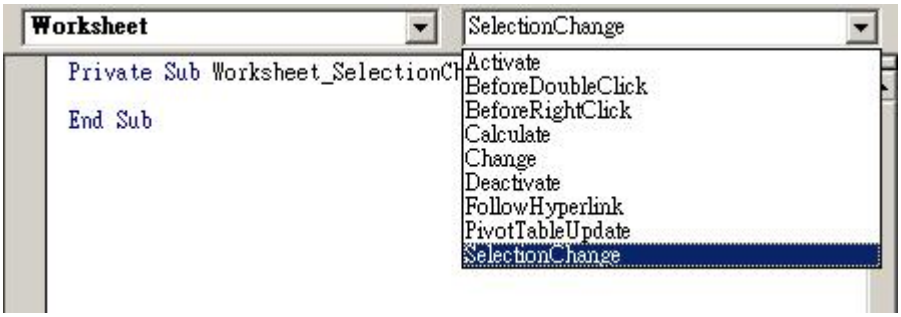
下面這個巨集，則透過 Activate 方法把 sheet3 工作表設定爲作用工作表：

```
Sub Activate()
    Worksheets(“sheet3”).Activate
End Sub
```

可以透過物件瀏覽視窗找出 excel 內建物件的屬性、方法及事件：




*如果開啓專案中某一特別物件，如表單、工作表或活頁簿，在「模組視窗」的「物件標示區」中會列出「一般」及所屬的物件名稱兩個選項，如果選擇物件名稱則代表撰寫物件的事件處理程序，此時在「巨集標示區」中會列出該物件所提供的事件程序：




若按下某一事件程序，則程式撰寫區會自動出現一個事件程序，名稱爲「物件名稱_事件名稱」，例如：


**Excel 物件**

*Excel 物件都具有層級性，舉例而言：

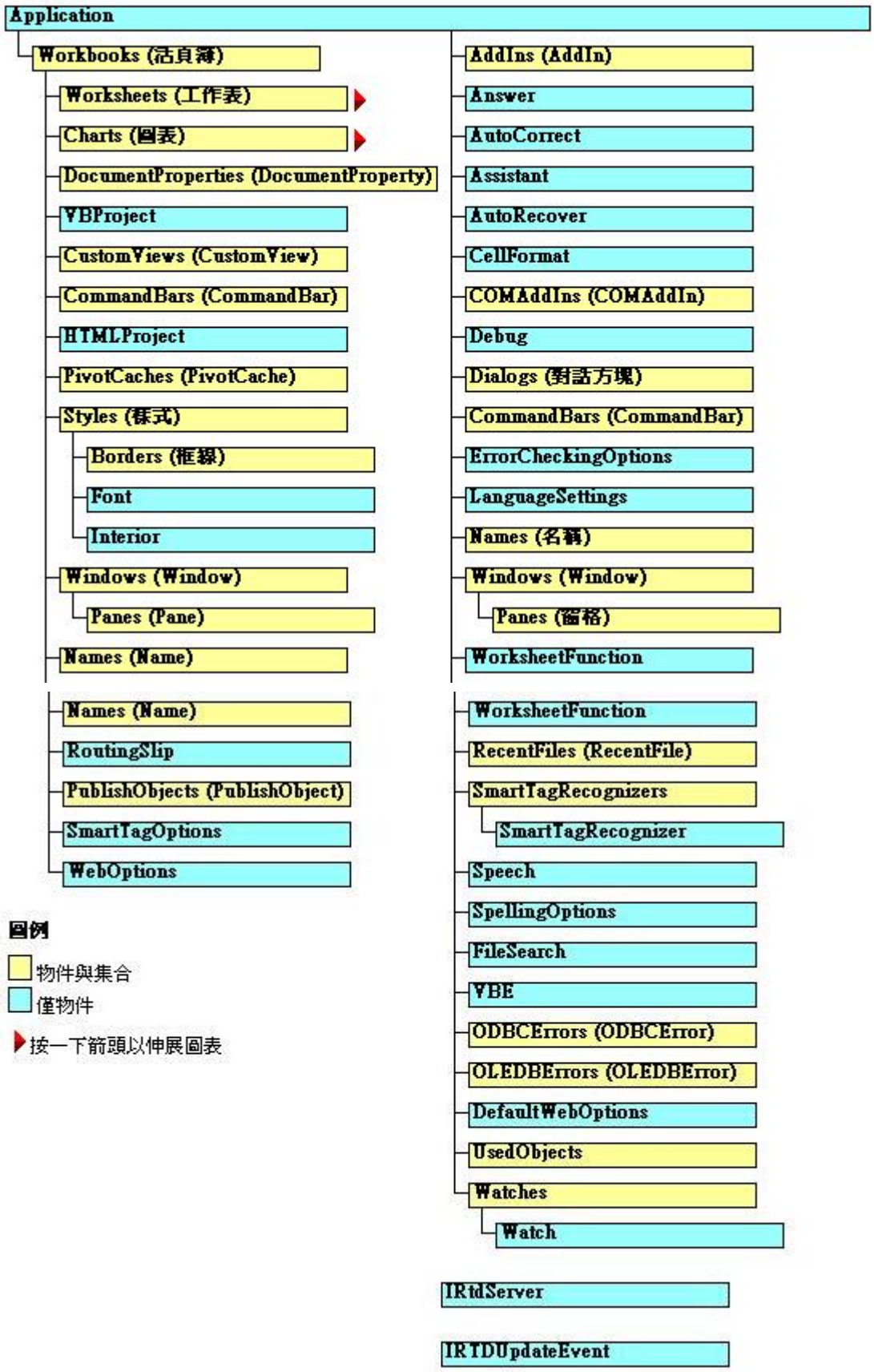
Application　本物件是 Excel 物件層級中的最上層物件，代表 Excel 應用程式

> Workbook 活頁簿物件

> Worksheet 工作表物件

> Range 儲存格物件

透過 Microsoft Visual Basic 說明功能或小幫手，在查詢框中輸入「Excel 物件」，可開啓 Excel 物件層級圖：




上述物件名詞爲複數者(底色爲黃色)，爲物件集合，例如 Workbooks 爲個別 Workbook 物件的集合。選按物件長方框，即進入各物件之說明，可了解各物件

的基本用法及屬性、方法及事件：


## 表單(Form) 簡介

*表單主要是用來做爲使用者介面，最常用的有 Excel 內建的 MsgBox 函數及 InputBox 函數，以及使用者或程式設計師自行設計的自訂表單。

**MsgBox 函數**：將訊息顯示在對話方塊中，等使用者按下按鈕，即傳回一 Integer 來指出使用者按下的是那一個按鈕。語法：

MsgBox( *prompt* [, *buttons*] [, *title*] [, *helpfile*, *context* ] )

或 MsgBox *prompt* [, *buttons*] [, *title*] [, *helpfile*, *context* ]

如果「MsgBox」包含在其他程式敘述裡，則必須使用第一種方式，加上( )號，讓它成爲一個真正的函數；如果是獨立的程式敘述，則使用第二種方式。函數中[ ]內的敘述可以省略。

| | |
|---|---|
| *prompt* 訊息 | 必要引數。顯示於訊息對話方塊內的文字，必須用「" "」將這些文字包起來。*prompt* 的最大長度大約是 1024 個字元，由使用字元的寬度決定。如果 *prompt* 超過一行，可以在每一行之間用復位字元 (Chr(13))、換行字元 (Chr(10)) 或是復位字元與換行字元的組合 (Chr(13) & Chr(10)) 來做區隔。 |
| *buttons* | 選擇性引數。數值運算式，用來指出顯示按鈕的數目及形式，使用 |

的圖示樣式，預設按鈕爲何，以及訊息方塊的強制回應等。如果沒有指定，則 *buttons* 的預設值是 0。

*title* 選擇性引數。顯示在對話方塊標題列中的字串運算式，必須用「" "」將這些文字包起來。如果沒有 *title*，則將應用程式的名稱放在標題列中。

*helpfile* 選擇性引數。用來辨識提供給對話方塊文字感應說明的說明檔案的字串運算式。如果指定了 *helpfile*，則也必須指定 *context*。

*context* 選擇性引數。數值運算式，由說明檔案的作者來指定適當的說明主題的說明主題代碼 。如果指定了 *context*，則也必須指定 *helpfile*

*Buttons* 的引數設定有下列多種：

| 常數 | 值 | 說明 |
|---|---|---|
| vbOKOnly | 0 | 只顯示 OK 按鈕。 |
| vbOKCancel | 1 | 顯示 OK 及 Cancel 按鈕。 |
| vbAbortRetryIgnore | 2 | 顯示 Abort、 Retry 及 Ignore 按鈕。 |
| vbYesNoCancel | 3 | 顯示 Yes、No 及 Cancel 按鈕。 |
| vbYesNo | 4 | 顯示 Yes 及 No 按鈕。 |
| vbRetryCancel | 5 | 顯示 Retry 及 Cancel 按鈕。 |
| vbCritical | 16 | 顯示 Critical Message 圖示。 |
| vbQuestion | 32 | 顯示 Warning Query 圖示。 |
| vbExclamation | 48 | 顯示 Warning Message 圖示。 |
| vbInformation | 64 | 顯示 Information Message 圖示。 |
| vbDefaultButton1 | 0 | 第一個按鈕是預設值。 |
| vbDefaultButton2 | 256 | 第二個按鈕 是預設值。 |
| vbDefaultButton3 | 512 | 第三個按鈕是預設值。 |
| vbDefaultButton4 | 768 | 第四個按鈕是預設值。 |
| vbApplicationModal | 0 | 應用程式強制回應：使用者必須先回應此訊息方塊，才能在目前的應用程式中繼續工作。 |

| | | |
|---|---|---|
| vbSystemModal | 4096 | 系統強制回應；所有的應用程式都會暫停，直到使用者回應此訊息方塊。 |
| vbMsgBoxHelpButton | 16384 | 將 Help 按鈕新增到訊息方塊中。 |
| VbMsgBoxSetForeground | 65536 | 指定訊息方塊視窗作爲前景視窗。 |
| vbMsgBoxRight | 524288 | 文字爲靠右對齊。 |
| vbMsgBoxRtlReading | 1048576 | 指定文字應爲在希伯來和阿拉伯語系統中的從右到左顯示。 |

第一組值 (0-5) 用來決定對話方塊中按鈕的形式與數目；第二組 (16, 32, 48, 64) 用來決定圖示的樣式；第三組 (0, 256, 512) 決定出那一個按鈕是預設值；而第四組 (0, 4096) 則決定訊息方塊的強制回應性。「將這些數字相加」以產生 *buttons* 引數值的時候，只能由每組取用一個數字。

*MsgBox 是函數，所以它會傳回一個數值，以顯示使用者選取的按鈕，程式設計師可利用傳回的值做一些程式上的處理。傳回值如下 (注意：此傳回值與上述 Buttons 引數的數值無關！！)

| 常數 | 值 | 描述 |
|---|---|---|
| vbOK | 1 | OK |
| vbCancel | 2 | Cancel |
| vbAbort | 3 | Abort |
| vbRetry | 4 | Retry |
| vbIgnore | 5 | Ignore |
| vbYes | 6 | Yes |
| vbNo | 7 | No |

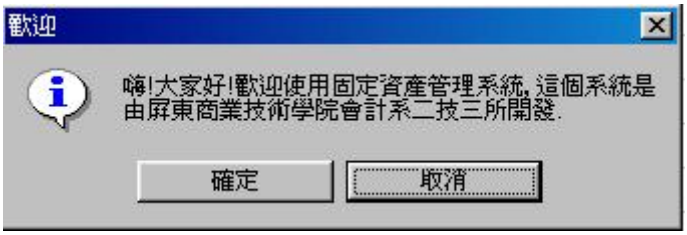
**MsgBox 函數釋例 1**(本例中，MsgBox 爲獨立的程式敘述)：

```
Sub Welcome()
    Msgbox "嗨!大家好!歡迎使用固定資產管理系統, 這個系統是" + Chr(10) _
    + "由屏東商業技術學院會計系二技三所開發. ", 1 + 64 + 256, "歡迎"
End Sub
```

Chr(10)之後的訊息會換行顯示

空白加底線( _ )則是程式碼中的換行符號，表示次行的內容與本行一氣呵成

執行結果：




**MsgBox 函數釋例 2**(本例中，MsgBox 包含在其他程式敘述裡)：

```
Sub Warning()
    Cells(1, 1) = Msgbox("嗨!大家好!", 0 + 16 + 0, "歡迎")
    Cells(1, 2) = Msgbox("嗨!大家好!", 1 + 32 + 256, "歡迎")
    Cells(1, 3) = Msgbox("嗨!大家好!", 2 + 48 + 512, "歡迎")
    Cells(1, 4) = Msgbox("嗨!大家好!", 3 + 64 + 256, "歡迎")
End Sub
```

執行結果：

程式碼：Cells(1, 1) = Msgbox("嗨!大家好!", 0 + 16 + 0, "歡迎")


若按下「確定」，A1 儲存格=1

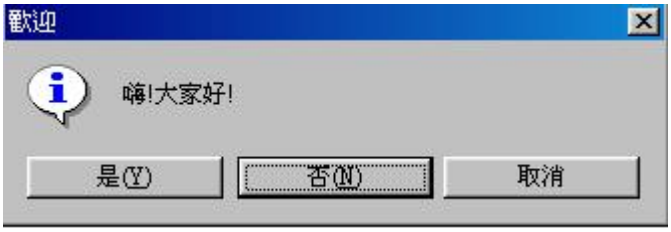
程式碼：Cells(1, 2) = Msgbox("嗨!大家好!", 1 + 32 + 256, "歡迎")


若按下「確定」，B1 儲存格=1；若按下「取消」，B1 儲存格=2

程式碼：Cells(1, 3) = Msgbox("嗨!大家好!", 2 + 48 + 512, "歡迎")

若按下「異常終止」，C1 儲存格= 3；若按下「重試」，C1 儲存格= 4；若按下「略過」，C1 儲存格= 5

程式碼：Cells(1, 4) = Msgbox("嗨!大家好!", 3 + 64 + 256, "歡迎")




若按下「是」，D1 儲存格= 6；若按下「否」，D1 儲存格= 7；若按下「取消」，D1 儲存格= 2

**InputBox 函數**：用來提示並讓使用者輸入文字的介面，可以透過這個介面取得使用者輸入的值。語法：

InputBox(*prompt*[, *title*] [, *default*] [, *xpos*] [, *ypos*] [, *helpfile*, *context*])

| | |
|---|---|
| ***prompt***<br>提示訊息 | 必要引數。用來做爲對話方塊訊息的字串運算式，必須用「" "」將這些文字包起來。***prompt*** 的最大長度大約是 1024 個字元，由使用字元的寬度來決定。如果 ***prompt*** 超過一行，您可以在各行之間用復位字元(**Chr**(13))、換行字元(**Chr**(10))或是復位字元與換行字元的組合(**Chr**(13) & **Chr**(10)) 來做區隔。 |
| ***title***<br>標題 | 選擇性引數。顯示在對話方塊標題列的字串運算式，必須用「" "」將這些文字包起來。如果沒有 ***title***，則以應用程式的名稱做爲標題。 |
| ***default***<br>預設值 | 選擇性引數。顯示在文字方塊中的字串運算式，必須用「" "」將這些文字包起來，在沒有提供其他輸入時做爲預設值。如果沒有 ***default***，則文字方塊就是空白的。 |
| ***xpos***<br>x 座標 | 選擇性引數。成對指定的數值運算式，用來指定對話方塊的左緣與螢幕左緣的水平距離。如果沒有 ***xpos***，則對話方塊會出現在水平方向的中間。 |
| ***ypos***<br>y 座標 | 選擇性引數。成對指定的數值運算式，用來指定對話方塊的上緣與螢幕的上緣的距離。如果沒有 ***ypos***，對話方塊會放置於螢幕垂直方向三分之一的位置。 |
| ***helpfile*** | 選擇性引數。字串運算式，用來指定對話方塊文字感應說明的說明檔案。如果指定了 ***helpfile***，則您也必須指定 ***context***。 |
| ***context*** | 選擇性引數。數值運算式，由說明檔案的作者指定給某個說明主題的 |

說明主題代碼 。如果指定了 ***context***，則也必須指定 ***helpfile***。

**InputBox 函數範例**：

```
Public Sub Inputname()
    Dim Guest As Variant
    Guest = InputBox("您好, 請輸入您的姓名", "輸入姓名", "我的姓名")
    MsgBox Guest + ", 您好! 歡迎使用財產管理系統. ", 0 , "歡迎"
End Sub
```

程式執行結果：




**自訂表單：**可藉由在表單物件內添加各式控制項物件，以便製作各式自訂對話方塊，促進使用者與應用程式間的雙向溝通。

*自訂表單的三個步驟：1.在 VBE 的專案視窗內新增一份自訂表單物件並設定該表單的物件屬性；2.在表單中增添控制項物件並設定該控制項物件的物件屬性；3.撰寫表單及各控制項物件的事件程序。

(詳後述)

# 第四講 資料型態及變數、常數

**VBA 的資料型態(Data Types)**

| 資料型態 | 使用的記憶體空間 | 數值範圍 |
|---|---|---|
| Byte(短整數) | 1 Byte | 0~255 |
| Integer (整數) | 2 Bytes | -32768~32767 |
| Long (長整數) | 4 Bytes | -2147483648~2147483647 |
| Single (單倍精準浮點數) | 4 Bytes | 負數：-3.4E38~-1.4E-45<br>正數：1.4E-45~3.4E38 |
| Double (雙倍精準浮點數) | 8 Bytes | 負數：-1.7E308~-4.9E-324<br>正數：4.9E-324~1.7E308 |
| String<br>(可變長度字串) | 10 Bytes+字串長度 | 最多 2 的 31 次方(約 20 億)個字元 |
| String*(長度)<br>(固定長度字串) | 與字串長度相等 | 最多 2 的 16 次方(32768)個字元 |
| Boolean(布林) | 2 Bytes | 「True」or「False」 |
| Variant (變數)<br>註：此 variant 變數與 variable 變數不同 | 數值：16 Bytes<br>字串：22 Bytes+<br>字串長度 | 數值型態時，與「Double」相同<br>字串型態時，與「可變長度字串」相同 |
| Currency (貨幣) | 8 Bytes | 922,337,203,685,477.5807~<br>-922,337,203,685,477.5808 |
| Date (日期與時間) | 8 Bytes | 日期：January 1,100~<br>December 31,9999<br>時間：00:00:00~23:59:59 |

## Boolean 資料型態

Boolean 變數係以 16 位元( 2 個位元組)數字的形式儲存，但只能是 True 或是 False。Boolean 變數的值不是 True 就是 False。使用關鍵字 True 與 False 可將 Boolean 變數指定爲這兩個狀態中的一個。

當轉換其他的數值型態爲 Boolean 時，0 會轉成 False，而所有其他的值則變成 True。當轉換 Boolean 值爲其他的資料型態,時，False 成爲 0 ，而 True 成爲 -1。

## Byte 資料型態

Byte 變數係以範圍在 0 至 255 之單精度、無正負號、8 位元 ( 1 個位元組 ) 數字的形式儲存。

## Currency 資料型態

Currency 變數係以 64 位元（8 個位元組）整數格式的數字形式儲存，在小數點左邊有 15 位數，右邊 4 位數的數字。這種表示法的範圍可以從 -922,337,203,685,477.5808 到 922,337,203,685,477.5807。

**Currency** 資料型態適用於精確度特別重要的貨幣與固定點計算，是兼具長整數及浮點數優點的一種型態。

### Date 資料型態

Date 變數係以 IEEE 64 位元(8 個位元組) 浮點數字的形式儲存，其可以表示的範圍從 1 January 100 到 31 December 9999，而時間可以從 0:00:00 到 23:59:59。任何可辨認的文字日期都可以指定給 **Date** 變數。短式日期須以數字符號 (**#**) 包住，例如，#January 1, 1993# 或 #1 Jan 93#。

**Date** 變數會根據電腦中的短日期格式來顯示；時間則根據電腦的時間格式（12 或 24 小時制）來顯示。

當其他的數值型態要轉換成 **Date**，小數點左邊的值表示日期資訊，而小數點右邊的值則表示時間。午夜爲 0 而中午爲 0.5。負整數表示 30 December 1899 之前的日期。

### Double 資料型態

Double (雙精度浮點數)變數係以 IEEE 64 位元 (8 個位元組) 浮點數字的形式儲存，它的範圍在負數的時候是從 -1.79769313486231E308 到 -4.94065645841247E-324，而正數的時候是從 4.94065645841247E-324 到 1.79769313486232E308。

### Integer 資料型態

Integer 變數係以範圍爲 -32,768 到 32,767 之 16 位元 (2 個位元組) 數字的形式儲存。

### Long 資料型態

Long (長整數)變數係以範圍從 -2,147,483,648 到 2,147,483,647 之 32 位元 (4 個位元組) 有號數字形式儲存。

### Single 資料型態

Single (單精度浮點數) 變數係以 IEEE 32 位元 (4 個位元組) 浮點數字的形式儲存，它的範圍在負數的時候是從 -3.402823E38 到 -1.401298E-45，而在正數的時候是從 1.401298E-45 到 3.402823E38。

### String 資料型態

字串有兩種：可變長度與固定長度的字串。

- 可變長度字串最多可有大約 20 億 ( 2^31)個字元。
- 固定長度的字串可有 1 到大約 64K ( 2^16 ) 個字元。

**Variant 資料型態**

Variant 資料型態是所有沒被明確宣告爲其他型態 ( 用如 Dim、Private、Public 或 Static 的陳述式)之變數的資料型態。Variant 是一種特殊的資料型態，除了固定長度 String 的資料及使用者自訂型態外，也可以包含任何種類的資料。數值資料可以是任何整數或實數值，負數時範圍從-1.797693134862315E308 到 -4.94066E-324，正數時則從 4.94066E-324 到 1.797693134862315E308。通常，數值 Variant 資料維持在其 Variant 中原來的資料型態。例如，如果您指定一 Integer 給 Variant，則接下來的運算會把此 Variant 當成一 Integer 來處理。然而，如果一算術運數針對含 Byte、Integer、Long 或 Single 之一 Variant **執行**，而當結果超過原來資料型態的正常範圍時，則在 Variant 中的結果會提升到較大的資料型態。如 Byte 則提升到 Integer，Integer 提升到 Long，而 Long 及 Single 則提升爲 Double。當 Variant 變數中有 Currency、Decimal 及 Double 值超過它們個別的範圍時，會發生錯誤。

您可以用 Variant 資料型態來取代任何的資料型態，以更有彈性的方式來運算。如果 Variant 變數的內容是數字，它可以用字串來表示數字或是用它實際的值來表示，由內容來決定，例如：

```
Dim MyVar As Variant
MyVar = 98052
```

在前面的例子中，MyVar 內有一值 98052 的數字表示。算術運算子將內有數值或字串資料的 Variant 變數視爲一數字。如果您用 + 運算子來將 MyVar 與其他含有數字或數字型態變數的 Variant 相加，結果便是一算術和。

**使用者自訂資料型態**

任何用 Type 陳述式定義的資料型態。使用者自訂型態可包含一或多個某種資料型態的元件，陣列，或一個先前定義的使用者自訂型態。例如：

```
Type MyType
    MyName As String     ' String variable stores a name.
    MyBirthDate As Date     ' Date variable stores a birthdate.
    MySex As Integer     ' Integer variable stores sex (0 for
End Type     ' female, 1 for male).
```

**型態轉換函數**

這些函數可強制一個運算式轉成特定的資料型態。

**語法**

CBool(*expression*)、CByte(*expression*)、CCur(*expression*)、CDate(*expression*)、CDbl(*expression*)、CDec(*expression*)、CInt(*expression*)、CLng(*expression*)、CSng(*expression*)、CStr(*expression*)、CVar(*expression*)、CStr(*expression*)，其中，*expression* 引數爲任何字串運算式或數值運算式。

**傳回型態**

下列顯示函數對應的傳回型態：

| 函數 | 傳回型態 | *expression* 引數範圍 |
|---|---|---|
| CBool | Boolean | 任何可使用的字串或數值運算式。 |
| CByte | Byte | 0 至 255。 |
| CCur | Currency | -922,337,203,685,477.5808 至 922,337,203,685,477.5807。 |
| CDate | Date | 任何可使用的日期運算式。 |
| CDbl | Double | 負數從 -1.79769313486231E308 至 -4.94065645841247E-324；正數從 4.94065645841247E-324 至 1.79769313486232E308。 |
| CInt | Integer | -32,768 至 32,767；小數部份將被轉換。 |
| CLng | Long | -2,147,483,648 至 2,147,483,647；小數部份將被轉換。 |
| CSng | Single | 負數爲 -3.402823E38 至 -1.401298E-45；正數爲 1.401298E-45 至 3.402823E38。 |
| CStr | String | 字串 CStr 傳回值是依據_*express* 引數。 |
| CVar | Variant | 若爲數值，則範圍與 Double 相同；若非數值，其範圍與 String 相同。 |

**變數**

*變數(variable)就是可以改變其內容資料的數，在撰寫程式時，將所設定的變數宣告合適的資料型態，可以讓變數在執行時立刻得到適當的記憶體空間配置。如果在撰寫程式時並未明確定義變數的資料型態，VBA 會自動將該變數的資料型態視爲 Variant 資料型態。

*變數宣告語法：

**Public/Private/Dim [Static] 變數名稱 [As 資料型態]**

在宣告變數時，如果省略**[As 資料型態]**內的部分，VBA 會自動將該變數的資料

型態視爲 Variant 資料型態。
字母大小寫會被系統視爲是一樣的，所以 QwErT 與 qWeRt 會被 VBA 視爲完全相同的變數。
Public、Private、Dim 的目的是要指定變數的使用範圍，由於 Dim 與 Private 的使用方式及結果完全相同，所以比較常用 Dim 而少用 Private。
*變數在宣告時會自動將變數的內容初始化，內定數值型態初始值爲 0，字串型態初始值爲空字串，布林型態初始值爲 False。
***[Static]**：一般在程序中呼叫 Sub 或 Function 程序時，被呼叫的程序會先將該程序內所宣告的變數初始化再執行程序內容。然而在某些特殊的情況下(例如計數器)，會希望將程序變數的內容值保留下來，以供下次執行該程序時使用，這種會保留上次執行內容值的變數就是靜態變數，其宣告方式是在變數名稱之前加上 Static 字樣。
*如果全域變數和區域變數的名稱相同，則程序會引用區域變數的內容。

**變數(補充)**
*變數命名規則：變數名稱必須以英文字母開頭，其後可以加上數字、英文字母或底線( _ )，也可以以中文當作變數名稱。
通常會使用 **Dim** 陳述式來宣告變數。一個宣告陳述式可以放到程序中以建立屬於程序層次的變數(即：區域變數)，或放到模組頂端的 Declarations 區段裡面，以建立屬於模組層次的變數(即：全域變數)。
下面的範例建立了變數 strName 並且指定爲 String 資料型態。

```
Dim strName As String
```

如果這個陳述式出現在程序中，則變數 strName 只可以在此程序中被使用。如果這個陳述式出現在模組中的 Declarations 區段，則變數 strName 可以被此模組中所有的程序所使用，但是不能被同一專案中不同的模組所含程序來使用。爲了使變數可被專案中所有的程序所使用，則在變數名稱前加上 **Public** 陳述式，如同下面的範例：

```
Public strName As String
```

可以在一個陳述式中宣告幾個變數；而爲了指定資料型態，必須將每一個變數的資料型態包含進來。在下面的陳述式中，變數 intX、intY、與 intZ 被宣告爲 **Integer** 型態。

```
Dim intX As Integer, intY As Integer, intZ As Integer
```

在下面的陳述式中，變數 intX 與 intY 被宣告爲 **Variant** 型態 ；只有 intZ 被宣告爲 **Integer** 型態。

```
Dim intX, intY, intZ As Integer
```

在宣告陳述式中，不一定要提供變數的資料型態，若省略資料型態的則會將變數設成 **Variant** 型態。
**使用 Public 陳述式**

可以使用 Public 陳述式去宣告公用的模組層次變數。

```
Public strName As String
```

公用變數可使用於專案中所有的程序中。

**使用 Private 陳述式**

可以使用 Private 陳述式去宣告私有的模組層次變數。

```
Private MyName As String
```

私有變數只可使用於同一模組中的程序。

**附註** 在模組層次中使用 Dim 陳述式與使用 Private 陳述式是相同的。不過使用 Private 陳述式可以更容易的去讀取並且解譯程式碼。

**使用 Static 陳述式**

當使用 Static 陳述式取代 Dim 陳述式，則所宣告的變數在程序呼叫時仍會保留它原先的值。

**使用 Option Explicit 陳述式**

在 Visual Basic 中可以輕易的透過一個指定陳述式來隱含性的宣告變數。所有隱含性宣告的變數都爲 Variant 型態，而 Variant 型態比一般型態的變數需要更多的記憶體來源。如明確的宣告變數爲某一特定的資料型態，則應用程式將更加有效率，明確宣告所有變數減少了名稱衝突以及拼字錯誤的發生率。

如果不想要 Visual Basic 產生隱含性宣告的話，可以將 Option Explicit 陳述式放置於模組層次中所有的程序之前。這一個陳述式會要求您對模組中所有的變數做明確的宣告。如果模組包含 Option Explicit 陳述式，則當 Visual Basic 遇到一個先前未定義的變數或拼字錯誤，它會發生編譯時間的錯誤。

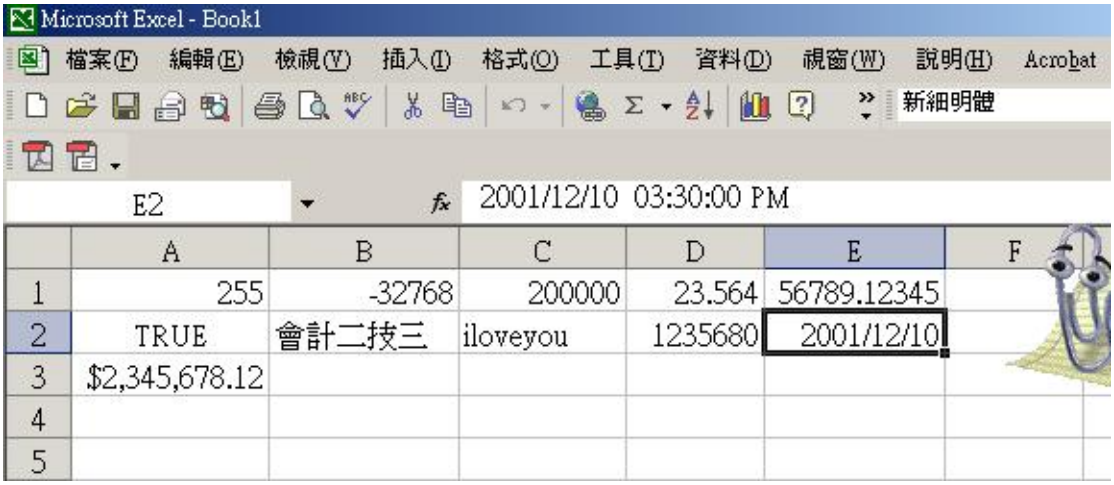
可以設定 Visual Basic 程式環境中的某個選項，使得可以自動在所有模組中加上 Option Explicit 陳述式。

**變數宣告範例**

```
Sub DeclareVariable()
    Dim a As Byte
    a = 255
    Dim b As Integer
    b = -32768
    Dim c As Long
    c = 200000
    Dim d As Single
    d = 23.564
    Dim e As Double
    e = 56789.12345
    Dim f As Boolean
    f = True
    Dim g As String    (接下頁)
```

```
    (續上頁)
    g = "會計二技三"
    Dim h As String * 8
    h = "iloveyou"
    Dim i As Variant
    i = 1235680
    Dim j As Date
    j = #12/10/2001 3:30:00 PM#
    Dim k As Currency
    k = 2345678.1234        (接下頁)
    Cells(1, 1) = a
    Cells(1, 2) = b
    Cells(1, 3) = c
    Cells(1, 4) = d
    Cells(1, 5) = e
    Cells(2, 1) = f
    Cells(2, 2) = g
    Cells(2, 3) = h
    Cells(2, 4) = i
    Cells(2, 5) = j
    Cells(3, 1) = k
End Sub
```

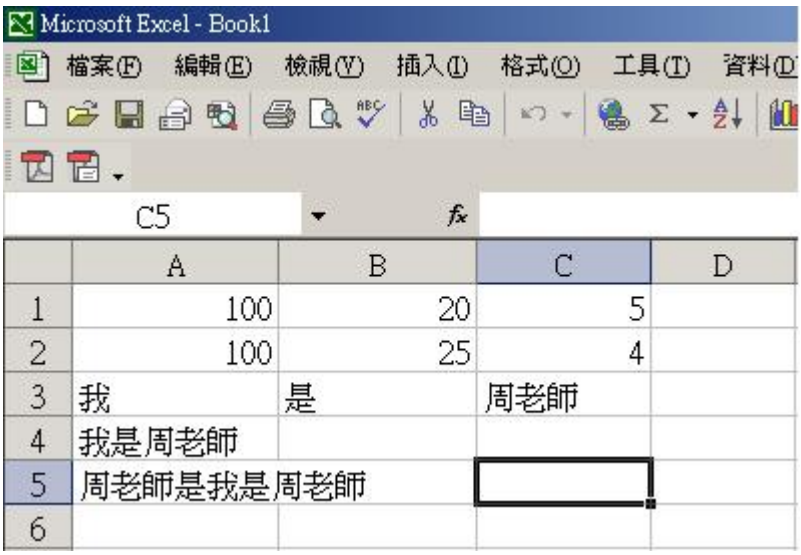
執行結果：




範例 2(變數的運算)

```
Sub Calculate()
    Dim a As Integer, b As Integer, c As Integer
    Dim d As String, e As String, f As String
    a = 100
    b = 20
    c = a / b
    Cells(1, 1) = a
    Cells(1, 2) = b
    Cells(1, 3) = c
    b = b + c
    c = a / b
    Cells(2, 1) = a
    Cells(2, 2) = b
    Cells(2, 3) = c
    d = "我"
    e = "是"
    f = "周老師"
    Cells(3, 1) = d
    Cells(3, 2) = e
    Cells(3, 3) = f
    d = d + e + f
    Cells(4, 1) = d
    e = f + e + d
    Cells(5, 1) = e
End Sub
```

執行結果



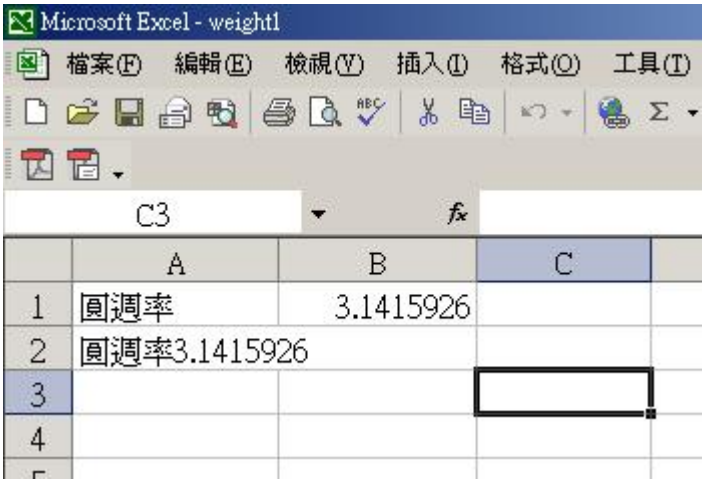
**常數**

爲一名稱，當程式在執行時用來代表一個不變的值。常數可以是字串、數值、另一常數、任何算術運算子（除指數外）或邏輯運算子（除 Is 外）的組合。每個主應用程式皆可定義自己的常數，使用者也可以 Const 陳述式來定義附加的常數。在程式中的任意位置均可使用常數以代替真正的值。

**自訂常數範例**

```
Sub 圓週率()
    Const pi = 3.1415926
    Dim x As String
    x = "圓週率"
    Cells(1, 1) = x
    Cells(1, 2) = pi
    x = x + CStr(pi)
    Cells(2, 1) = x
End Sub
```

執行結果：




**陣列(Array)**

陣列提供一個空間，讓多筆依序排列且資料型態相同的元素儲存在一起共同管理。宣告語法如下：

**Dim 陣列名稱(索引範圍) As 資料型態**

例如：「Dim Student(41) As String」，則 Student 這個陣列中就可以儲存至少 41 筆字串資料(至多 42 筆)，像是這學期修「Excel 在會計上之應用」的 41 名同學的名字。

*「索引範圍」決定了陣列的大小，但它指的是陣列索引的最大值(上界)，而非陣列元素的個數。「索引範圍」的值必須是一個 Long 型態的數字或變數。

*基本上，陣列索引值會由 0 開始算起，但也可以透過 Option Base 敘述來更改。如果設定爲 Option Base 0，則 Dim Student(41)這個敘述會保留 42 個陣列元素，陣列的索引值由 0 開始，分別是 Student(0)、Student(1)、Student(2)…到 Student(41)。如果設定爲 Option Base 1，則 Dim Student(41)這個敘述只會保留 41 個陣列元素，陣列的索引值由 1 開始，分別是 Student(1)、Student(2)…到 Student(41)。Option Base 這個敘述必須寫在模組宣告區中，如果沒寫，則內定爲 Option Base 0。

**多維陣列**

當所要儲存的元素比較複雜時，也可以使用多維陣列的方式來儲存。

**陣列宣告的例子**

| 宣告 | 陣列元素 |
|---|---|
| Dim A(10) As Integer | A(0) ~ A(10)　共 11 個元素 |
| Dim B(3 To 10) As Byte | B(3) ~ B(10)　共 8 個元素 |
| Dim C(2,3) | C(0,0) ~ C(2,3)　共 12 個元素 |
| Dim D(2,1 To 3) | D(0,1) ~ D(2,3)　共 9 個元素 |

**動態陣列**

*動態陣列在宣告時可以不必指定索引範圍，表示其大小未定，如下所示：

**Dim A( ) As Long**

等到需要使用時，再以程式碼敘述指定索引範圍即可。

# 第五講 運算子

## 運算式、運算子、運算元

*一個運算式(Expression)是由運算元(Operant)和運算子(Operator)所共同組成。
*運算式中做爲運算的資料稱爲運算元，運算元可以是常數、變數、函數或運算式。
*介於運算元間的運算符號稱爲運算子，如「+」、「-」、「*」、「/」是最典型的運算子。
*運算子根據其所需運算元的多寡可分爲「二元運算子」和「一元運算子」，其語法分別如下：

二元運算子： operant1 **operator** operant2 例如：1+6, 5*3, 8/2
一元運算子： **operator** operant 例如：-7, Not A

## 算術運算子

| 運算子 | 作用 | 優先次序 | 種類 | 範例 |
|---|---|---|---|---|
| ^ | 指數運算 | 1 | 二元 | 5^3=125 |
| - | 負數運算 | 2 | 一元 | -20 |
| * , / | 乘, 除 | 3 | 二元 | 10*6=60, 8/2=4 |
| \ | 兩數相除取商數<br>(商數爲整數值) | 4 | 二元 | 10\3=3 |
| Mod | 兩數相除取餘數 | 5 | 二元 | 10 Mod 3 = 1 |
| + , - | 加, 減 | 6 | 二元 | 5+3-2=6 |
| 範例：-5+2^3*3 Mod 4/2 = ? | | | | |

## 比較運算子

*比較運算子是用來做爲比較兩個運算元之用，其運算結果只有「True」及「False」兩種。比較運算子和後述之邏輯運算子常用於流程控制及迴圈（詳第六講）之條件判斷式中，以做爲分支跳躍或控制迴圈的條件。

| 運算子 | 作用 | 優先次序 | 範例(假設 x = 5) | 傳回結果 |
|---|---|---|---|---|
| = | 等於 | 1 | x=5 | True |
| <> | 不等於 | 2 | x<>5 | False |
| < | 小於 | 3 | x<6 | True |
| > | 大於 | 4 | x>7 | False |
| <= | 小於等於 | 5 | x<=5 | True |
| >= | 大於等於 | 6 | x>=8 | False |

## 邏輯運算子

*用來執行運算式之間的邏輯運算，判斷運算式的真僞。其運算結果只有「True」

及「False」兩種。

| 運算子 | 作用 | 優先次序 | 種類 | 範例(假設 x=5,y=7) | 傳回結果 |
|---|---|---|---|---|---|
| Not | 不是 | 1 | 一元 | Not (x=5) | False |
| And | 且 | 2 | 二元 | (x=5) And (y=7) | True |
| Or | 或 | 3 | 二元 | (x=4) Or (y=7) | True |
| Xor | 互斥或 | 4 | 二元 | (x=5) Xor (y=7) | False |
| Eqv | 相等 | 5 | 二元 | (x=3) Eqv (y=9) | True |
| Imp | 推論 | 6 | 二元 | (x=4) Imp (y=7) | True |

說明：

*Not：當運算式成立，則傳回「False」；如果運算式不成立，則傳回「True」。

*And：當兩個運算式都成立，才會傳回「True」；否則就傳回「False」。

*Or：當兩個運算式中有任何一個成立，就傳回「True」；只有當兩個運算式都不成立時，才傳回「False」。

*Xor：當兩個運算式中，一個成立且另一個不成立時，就傳回「True」；當兩個都成立或兩個都不成立時，則傳回「False」。

*Eqv：當兩個運算式都成立或兩個都不成立時，就傳回「True」；否則傳回「False」。

*Imp：當兩個運算式中，第一個成立且第二個不成立時，就傳回「False」；否則傳回「True」。

**練習：真值表(請在空格中填入 T 或 F)**

| 運算元 | | 邏輯運算子及運算結果 | | | | | |
|---|---|---|---|---|---|---|---|
| X | Y | And | Or | Xor | Eqv | Imp | Not X |
| F | F | | | | | | |
| F | T | | | | | | |
| T | F | | | | | | |
| T | T | | | | | | |

**字串運算子**

| 運算子 | 作用 | 範例 |
|---|---|---|
| + | 字串連結 | A=“1”+”2”, A=”12”; B=1+2, B=3 |
| & | 可把非字串連成字串<br>但通常&用來取代+ | x=2, y=”5”, x & y=”25”<br>A=“1”&”2”, A=”12” |
| Like | 比較兩字串是否相符 | Boolean1=”Hello” Like “H*o”, Boolean1=True |

Like 的語法：

**布林變數＝字串 1　Like　字串 2**

字串 2 中可以使用下列的字元來符合字串 1：

| 字串 2 中的字元 | 符合字串 1 中的 |
|---|---|
| ? | 任意的單一字元 |
| * | 零個以上的字元 |
| # | 任一個數字（0-9） |
| [charlist] | 在[charlist]中的任一字元 |
| [!charlist] | 不在[charlist]中的任一字元 |

Like 範例：

| 例(Bool 爲 Boolean 型態變數) | Bool 的內容 |
|---|---|
| Bool=”aQQQa” Like “a*a” | True |
| Bool=”G” Like “[A-Z]” | True |
| Bool=”H” Like “[!G-N]” | False |
| Bool=”a3a” Like “a#a” | True |
| Bool=”aM5b” Like “a[L-P]#[!c-e]” | True |
| Bool=”BiG123qwe” Like “B?G*” | True |
| Bool=”PingDong” Like “P?D*” | False |

**指定運算子**

*有「＝」及「.」兩種。

語法：變數=變數

變數=常數

變數=運算式

物件.方法

物件.屬性

範例：X=Y

X=10

X=Y+10

Worksheet.Calculate　(Worksheet 物件的 Calculate 方法)

Worksheet.AutoFilter　(Worksheet 物件的 AutoFilter 屬性)

# 第六講 流程控制與迴圈

**If … Then**

語法：
```
If 條件敘述 Then
程式敘述
…….
End If
```

涵義：當「條件敘述」成立，則執行 If 與 End If 間的程式敘述，並於執行完後繼續執行 End If 之後的敘述；若「條件敘述」不成立，則略過這些程式敘述，直接執行 End If 後面的程式敘述。

**If … Then … Else**

語法：
```
If 條件敘述一 Then
程式敘述一
…….
Else
程式敘述二
…….
End If
```

涵義：當「條件敘述一」成立，則執行 If 與 Else 間的程式敘述一，並於執行完後繼續執行 End If 之後的敘述；若「條件敘述一」不成立，則略過程式敘述一，直接執行程式敘述二，並於執行完後繼續執行 End If 後面的程式敘述。本流程設計至少會執行「程式敘述一」或「程式敘述二」中的一種。

**If … Then … ElseIf …Else**

語法：
```
If 條件敘述一 Then
程式敘述一
…….
ElseIf 條件敘述二 Then
程式敘述二
…….
[ElseIf 條件敘述 n-1 Then
程式敘述 n-1
…….]
[Else
程式敘述 n
…….]
End If
```

涵義：當「條件敘述一」成立，則執行程式敘述一，並於執行完後繼續執行 End If 之後的敘述；若「條件敘述一」不成立，則略過程式敘述一，並判斷「條件敘述二」是否成立，若成立則執行程式敘述二，並於執行完後繼續執行 End If 後面的程式敘述，依此類推。在流程設計中若有[Else….]的部分，則本流程設計至少會執行「程式敘述一」到「程式敘述 n」中的一種。若沒有[Else….]的部分，且在[Else….]之前的各個 ElseIf 後的條件敘述均不成立，則本流程設計中的程式敘述都不會被執行，直接跳至 End If 後的敘述。

**Select Case**

語法：
```
Select Case 變數
Case 變數值 [to 變數值]
程式敘述一
…….
Case 變數值 [to 變數值]
程式敘述二
…….
[Case 變數值 [to 變數值]
程式敘述 n-1
…….]
[Case Else
程式敘述 n
…….]
End Select
```

涵義：Select Case 敘述，必須指定一個變數做爲選擇的指標，程式會依據變數的內容做爲判斷的標準，如果某特定「Case」後的變數值與變數內容相同，就執行該 Case 程式區塊的程式敘述。變數的內容可以是數字或字串，也可以在「Case」敘述中使用「to」指定變數值範圍，當變數內容在某個範圍值內時，就執行特定的程式敘述。在流程設計中若有[Case Else….]的部分，則本流程設計至少會執行「程式敘述一」到「程式敘述 n」中的一種。若沒有[Case Else….]的部分，且變數內容均未落在[Case Else….]之前的各個 Case 後的變數值範圍內，則本流程設計中的程式敘述都不會被執行，直接跳至 End Select 後的敘述。

**Do While… Loop**

語法：
```
Do While 條件敘述
程式敘述
…….
Loop
```

涵義：當 Do While 後面的條件敘述成立(True)時，就執行程式敘述，執行完畢遇到 Loop 時，再跳回 Do While 後面的條件敘述，若條件仍成立，則繼續執行程式敘述。當 Do While 後面的條件敘述「不再成立」或「自始就不成立」(False)時，就直接跳至 Loop 後面的敘述。此結構屬於「前測型」的迴圈，當條件敘述自始就不成立時，迴圈內的程式敘述完全不會被執行。

**Do Until… Loop**

語法： **Do Until 條件敘述**
**程式敘述**
**…….**
**Loop**

涵義：Until 條件敘述與 While 條件敘述爲互補關係。當 Do Until 後面的條件敘述「不成立」(False)時，就執行程式敘述，執行完畢遇到 Loop 時，再跳回 Do Until 後面的條件敘述，若條件仍不成立，則繼續執行程式敘述。當 Do Until 後面的條件敘述「成立」(True)時，就直接跳至 Loop 後面的敘述。此結構屬於「前測型」的迴圈，當條件敘述自始就成立時，迴圈內的程式敘述完全不會被執行。

**Do … Loop While**

語法： **Do**
**程式敘述**
**…….**
**Loop While 條件敘述**

涵義：此結構屬於「後測型」的迴圈，迴圈內的程式敘述會無條件執行一次，當程式遇到 Loop While 後面的條件敘述時，若條件成立，則再執行迴圈內的程式敘述一次，如此週而復始，直至條件敘述不再成立時，才離開迴圈。

**Do … Loop Until**

語法： **Do**
**程式敘述**
**…….**
**Loop Until 條件敘述**

涵義：此結構屬於「後測型」的迴圈，迴圈內的程式敘述會無條件執行一次，當程式遇到 Loop Until 後面的條件敘述時，若條件不成立，則再執行迴圈內的程式敘述一次，如此週而復始，直至條件敘述成立時，才離開迴圈。

**Do … Loop**

語法： **Do**
**程式敘述**
**…….**
**Loop**

涵義：此結構稱爲「無窮迴圈」，迴圈內的程式敘述會無止境的重複執行下去。前述的四種 Do Loop 迴圈，若條件敘述永遠成立(或不成立)，也會成爲無窮迴圈。無窮迴圈的程式敘述中通常會有其他的流程控制結構，當這些流程控制結構中的條件滿足時，就以「Exit Do」敘述跳離整個 Do Loop 迴圈。

**For … Next**

語法： **For 數值變數 = 初始值 To 終止值 [Step 增量]**
**程式敘述**
**…….**
**Next [數值變數]**

涵義：

*當程式遇到 For 敘述時，先將「數值變數」設定爲「初始值」，然後執行迴圈內的程式敘述；當程式遇到 Next 敘述時，就回到 For 敘述，並把「數值變數」加上「增量」，並判斷是否大於終止值，若爲否(False)，則重複執行迴圈內的程式敘述，如此週而復始；直到數值變數的值大於終止值時，就跳離 For 迴圈。

*若「增量」爲負，則須判斷數值變數的值是否小於終止值，若爲否(False)，則重複執行迴圈內的程式敘述，如此週而復始；直到數值變數的值小於終止值時，就跳離 For 迴圈。

*[.]中的敘述可以省略，若省略「Step 增量」敘述，則電腦會自動設定增量爲 1。

**練習：**用 For … Next 迴圈設計一個 1 加到 100 的程式。

**For Each … Next**

語法： **For Each 元素 In 群組**
**程式敘述**
**…….**
**Next [元素]**

涵義：

*「元素」是一個變數，用來對應群組中的各個元素。

*「群組」是由許多個別元素所組成的陣列或物件集合。

*當程式遇到 For Each 敘述時，就把群組中的第一個元素指定給「元素」變數，

然後執行迴圈內的程式敘述；當程式遇到 Next 敘述時，就回到 For Each 敘述，並將群組中的下一個元素指定給「元素」變數，然後重複執行迴圈內的程式敘述，如此週而復始，直到群組中的元素依序指定完畢爲止。

**While … Wend**

語法： **While 條件敘述**
**程式敘述**
**…….**
**Wend**

涵義：本迴圈用法與「Do While … Loop」迴圈完全相同。

**跳離指令**

VB 中有許多跳離指令，可以強制程式終止或離開特定程式片段。爲了避免出現「無窮迴圈」的困擾，可以在迴圈中設計跳離指令，當特定條件滿足或不滿足時(通常用 If 敘述做判斷)，即執行跳離指令，離開迴圈。VB 中的跳離指令如下：

*Exit Do：強制離開 Do Loop 迴圈。
*Exit For：強制離開 For Next 迴圈。
*Exit Sub：強制跳離 Sub 程序。
*Exit Function：強制跳離 Function 程序。
*Goto：(略)　不鼓勵使用(需在程式碼前加上行號)
*End：無條件結束應用程式。