第一章

電腦概要與 VBA 介紹

財務計算的需求自古即有，但近代財務的演進，對此需求日益殷切。電腦的優點之一，在於快速正確的計算能力。配合近代程式語言的發展，財務計算的工作，已經發展成一項專業的能力。我們一方面回顧此一發展的過程，另一方面介紹本書使用的開發工具，Excel VBA 的工作環境。

第一節 財務與計算

現代的財務理論始於 1950 年代，由 Markowitz 所提出的投資組合理論(Portfolio Theory)，與 Treynor、Sharpe、Linter 和 Mossin 進一步的發展所提出的資本資產訂價理論(CAPM, Capital Asset Pricing Model)，成為財務中研究證券資產價格與風險的數量模型。在這些理論中，基本上是使用隨機變數的平均數(Mean)與變異數(Variance)去表達證券的報酬與風險，這種數量上的進展，大幅地提升財務的定量研究。

然而，在 1950 年代中，還有一個重要的理論發展，對目前財務數量模型有著巨大影響的，那便是由 Miller 與 Modigliani 所提出的無套利假說(Arbitriage Free Hypothesis)。此假說主張在均衡的市場情況下，套利組合是不存在的。這一主張對衍生商品的訂價模型，有著主導性的角色。

1970 年代可說是財務工程的起始時段，這是由於 Black、Scholes 與 Merton (BSM)提出了著名的選擇權訂價理論(Option Pricing Theory)。事實上，早在 1900 年便有學者提出了選擇權契約的訂價公式，然而該公式並沒有被實務工作者所採用。BSM 的模型之所以廣為市場所接受，在於其理論假設的合理性，與實務操作上的可行性。BSM 應用無套利假說，將選擇權與標的股票形成無套利組合，利用此均衡條件，設定此投資組合的應有報酬為無風險報酬。BSM 另一個重要的突破，在於對股票價格行為的假設上，選擇了一個相當良好的隨機過程(Stochastic Process)—幾何布朗運動型式(Geometrical Brownian Motion, GBM)，配合隨機積分(Stochastic Integral)的伊藤補理(Ito's Lemma)，得到選擇權價格的偏微分方程式(Partial Differential Equation, PDE)。

在運用微分方程求解的技巧後，BSM 求得了下面著名的選擇價格公式，

$$C = SN(d_1) - Ke^{-rT}N(d_2) \quad \text{.....(1.1.1)}$$

$$P = Ke^{-rT}N(-d_2) - SN(-d_1) \quad \text{.....(1.1.2)}$$

其中

$$d_1 = \frac{\ln(S/K) + (r + \sigma^2/2)T}{\sigma\sqrt{T}}$$

$$d_2 = \frac{\ln(S/K) + (r - \sigma^2/2)T}{\sigma\sqrt{T}} = d_1 - \sigma\sqrt{T}$$

S = 標的資產目前的股價，

K = 契約執行價格，

r = 無風險資產的收益率(YTM)為即期利率，

T = 到期日的時間，

σ = 股價之波動性，

函數 $N(x)$ 為標準常態變數之累積機率密度函數(Cumulative Probability Density Function, CDF)。此一定價公式改寫了財務工作者的命運，也創造出一種全新的工作型態，即財務工程師的職業與財務計算的需求。

到了 1980 年代，各式各樣新奇(Exotic)的衍生商品大量出現，這些新奇商品定價公式的複雜度，已非前述 BSM 陽春型(Vanilla)的選擇權可以比擬。如不借助電腦程式的協助，這種單一交易複雜的運算，很難得到正確的數值。及至 1990 年代風險管理的涉險值(Value at Risk, VaR)模型的出現，管理者需要快速地評估資產組合的可能損失，這種大量交易的複雜運算需求，都使得財務計算

成為一項重要的課題。

而在 21 世紀之初，結構型商品(Structured Product)又成為市場的風騷，這些契約大都有複雜的償付條件，而且對於標的資產的行為模式，有進一步較為複雜的程序在加以描述，因此需要仰賴大量的模擬運算來解決評價與風險計算的功能。此時，大量的超級電腦已被金融機構用於產品的開發之上。另外，電玩遊戲與人工智慧帶動的高運算需求，使得繪圖卡(Graphic Processing Unit)通用運算(General Purpose)技術得以實現，這帶給財務工程一個全新的新境界。

第二節 電腦與計算

電腦就是能夠以人類數百萬倍甚至幾十億倍的速度，進行計算和邏輯決策的一種裝置。例如，許多現代的個人電腦在每秒鐘可以進行數億——甚至數十億次的加法運算。一個人操作桌上型計算機可能需要一輩子的時間，才能完成威力強大的個人電腦在一秒鐘內所完成的計算工作。現在最快的超級電腦每一秒可以執行數千億次計算，約與數十萬人一年所做的計算量等同。有些實驗室甚至已經使用每秒數兆指令的電腦！

電腦是依賴所謂的電腦程式，以一串的指令來處理資料的。這些程式透過個人所指定的動作順序來引導電腦運作，這些個人就是所謂的電腦程式設計師。電腦是由各種裝置所組成的（例如鍵盤、螢幕、滑鼠、硬碟、記憶體、光碟機與處理單元等），就是所謂的硬體。而在電腦上執行的程式，則稱為軟體。近幾年硬體的費用已經大幅降低，使得個人電腦已經成為一種普通的商品。經過適當訓練的電腦使用者，也可以自行撰寫一些簡易的電腦程式，解決工作上的一些資料處理與複雜計算的需求。然而，在電腦發展的早期時代，事情並不是如此的容易，這是有賴於作業系統與程式語言多年的發展，才有今日的便利性。

一、作業系統的演進

早期的電腦是由專門的人員負責操作，這些人員本身也就是程式設計人員。這和現代幾乎每一個人都是電腦使用者的情況大不相同。那時的電腦每次只能執行一項工作。這些早期系統的使用者要使用打孔的卡片，將工作輸入到電腦中。通常，使用者必須等待好幾個小時或好幾天才會得到結果。為了使電腦的使用更為方便，於是發展了所謂的作業系統(Operating System)的軟體。早期的作業系統僅能監督與管理電腦在工作之間的轉換過程。透過將電腦在工作之間轉換所花費的時間縮減到最少，作業系統藉此得以增加電腦在既定時段裡可以處理的量。

當電腦越來越強大後，單一使用者的處理就顯得沒有效率了，因為電腦花費許多時間在等待緩慢的輸入／輸出裝置完成其工作。因此，人們就想到如果能使用多工程式的技術，也就是讓許多的工作共用電腦的資源，便可以達到更好的使用率。多工程式(Multi-programming)在電腦上「同時」執行許多的工作，將電腦的資源分配給各個工作使用。

在 1960 年代，幾個工業界的機構和大學開發了分時(Timesharing)作業系統。分時是一種特殊的多工程式，使用者可以透過終端機來存取電腦主機，而終端機的標準配備通常是鍵盤和螢幕。一次可以讓好幾十個甚至好幾百個人使用分時的電腦系統。電腦並不是真的同時執行所有使用者的要求。反之，電腦只會執行某個使用者的一小部分工作，然後就服務下一位使用者。然而，因為電腦執行得非常快速，每一秒鐘可以服務每一位使用者好幾次。這使得使用者的程式看起來是同時在執行。分時比以前的計算系統所具有的主要優勢在於，使用者可以得到即時的反應，不需要耗費很長的時間來等待結果。

目前廣泛使用於高等運算的 UNIX 作業系統，就是來自於實驗的分時作業系統。在 1960 年代的晚期，Dennis Ritchie 與 Ken Thompson 在貝爾實驗室開始發展 UNIX，以及用來撰寫這種作業系統的程式語言 C。他們將 UNIX 建造為開放原始碼的軟體，允許其他有需要的程式設計師自由散佈、使用、修正與擴

增其功能，因此快速發展出許多 UNIX 的社群。當 UNIX 使用者傳播他們自己的程式與工具的同時，作業系統也因此成長。在許多研究人員與發展人員的合作努力之下，UNIX 變成功能強大而有彈性的作業系統，可以處理使用者所要求的任何工作。現在已經有許多版本的 UNIX，包含非常流行的 Linux 作業系統。

二、單機計算、分散式計算和主／從式計算

在 1977 年，蘋果電腦公司推廣所謂個人電腦(Person Computer, PC)概念。剛開始，這只是玩家的一種夢想。然而，截至目前為止，電腦的價格已經大為降低，使得許多人有能力購買電腦，以做為個人用途或商業用途。在 1981 年，世界最大的電腦供應商 IBM 推出了了 IBM PC。個人電腦很快就成為商業、個人與政府組織的必需品。

蘋果電腦與 IBM 所首創的電腦是「獨立」的單元——人們在自己的機器上工作，然後來來回回地傳遞磁片以分享資訊。雖然早期的 PC 並未強大到能夠在幾個使用者之間進行分時的操作，但是這些機器仍然能夠彼此連結起來，形成電腦網路。有時會它們透過電話線、而有時會透過企業組織內部的區域網路來進行連結。

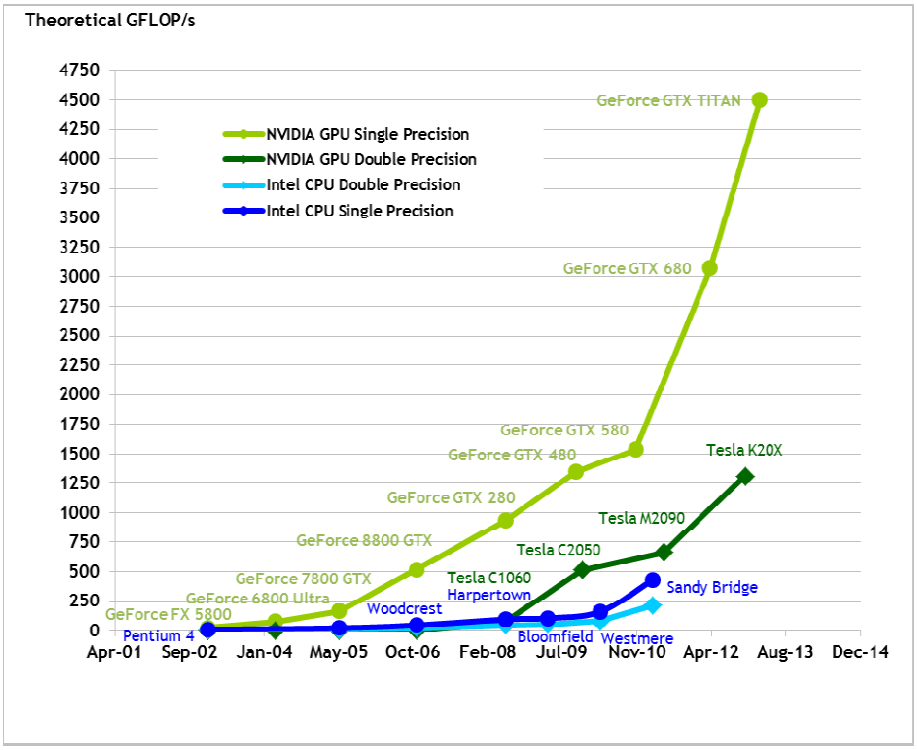
這些網路產生分散式計算(Distributed Computing)，在這種環境下，企業內部的電腦不再受限於某些中央電腦主機，而是會分散到網路的各個位置上。電腦的安裝位置可以放置到企業內部需要電腦的地方。PC 已經能夠處理個別使用者在計算上的需求，也能夠處理資訊彼此傳遞所需要的基本通訊功能。多層式應用架構(Multi-tier Application Framework)將應用程式劃分到許多電腦。例如，三層式應用架構可能在某一台電腦上有使用者介面，在第二台電腦有商務邏輯處理，然後在第三台電腦上具有資料庫。所有三層應用架構的程式互動情況與單一應用程式執行時的情況完全相同。

今日最強大的個人電腦已經可以與十年前價值幾百萬元的電腦相媲美。最強大的桌上型電腦，稱為工作站——能夠提供個別使用者大量的功能。資訊可

以很容易便跨越網路共享，其中稱為伺服器的電腦將客戶端所能使用的程式與資料加以儲存，而客戶端電腦則分散在整個網路上。這種組態就稱為主從式計算(Client/Server Computing)。現今流行的作業系統，諸如 UNIX、Linux、Solaris、MacOS、Windows 8 與 Windows 10，都具備本節討論的功能。

三、繪圖卡支援通用計算成熟

圖二 2001-2013 Intel CPU 與 NVIDIA GPU 理論浮點計算能力



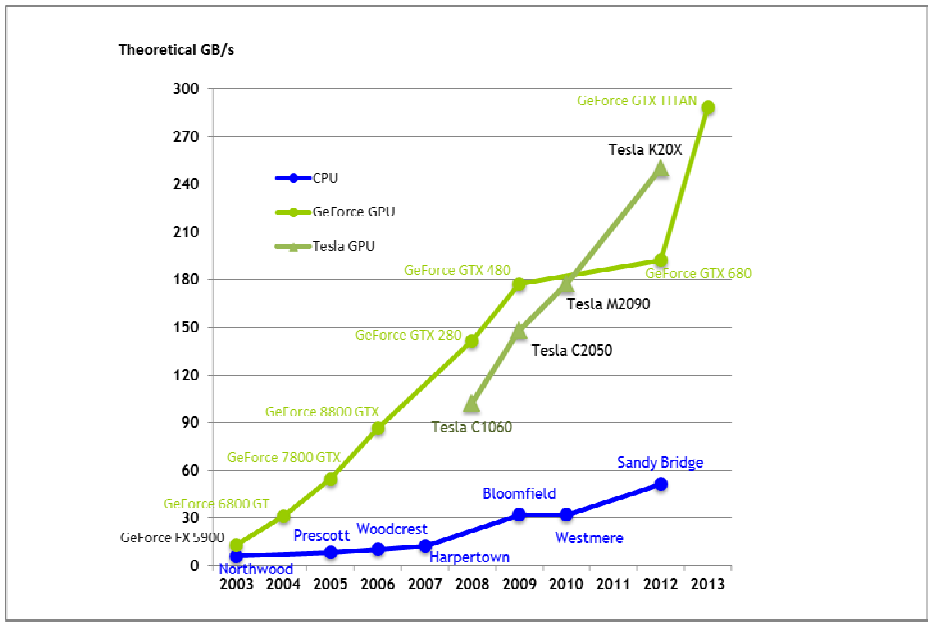
在過去的發展歷史上，對於大部分的電腦而言，顯示卡(Display Card)已是個不可獲缺的配件。顯示卡的功能一直被定位為輔助 CPU 來處理電腦上相關圖形與影像方面的計算工作。早期顯示卡上的圖形處理器(Graphics Processing Unit, GPU)中的運算核心的數目不多，約為 2 至 16 個。但近年來，隨著製程技術的改良，一方面 GPU 的運算時脈提升，另一方面 GPU 運算核心的數目大舉上升到 500 至 2000 個。這使得顯示卡每秒所能處理的浮點運算數目跟著明

顯地提升。

圖一(註一)中可以看到，NVIDIA 公司所生產的 GPU 顯示卡，在 2003 年 G80 系列之後的單精準度浮點運算的理論值，就開始大幅領先 Intel 的 CPU 了，之後逐年擴大領先差距。至於在雙精準度浮點運算方面，在 NVIDIA 推出 Tesla 系列後，GPU 的運算能力也優於 CPU 的雙精準度浮點運算效能。

除此之外，顯示卡上的記憶體容量也達到 1 至 2GB。而且，顯示卡的記憶體存取頻寬也不斷在成長。NVIDIA 的 Geforce GTX 480 顯示卡記憶體存取頻寬已達到每秒 180GB，明顯地優於 Intel CPU。顯示卡上記憶體容量的增加以及記憶體存取頻寬的提升，使得涉及大量資料存取的應用，可以獲得更高的計算效能提升。

圖二 2003-2013 Intel CPU 與 NVIDIA GPU 理論資料存取頻寬



根據牛津大學 Mike Giles 教授在 2013 年所做的調查(註四)，在所有 Top 500 的超級電腦中，有 10%是用於金融計算之上，主要分別用於投資銀行產品開發與避險基金的業務之上。

在投資銀行內，計算的重點在於各種選擇權訂價模型的應用。其中，主要的應用為蒙地卡羅模擬法(Monte Carlo Simulation Method)，佔了 60%的計算量；PDE 與有限差分法(Finite Difference Method)次之，約佔 30%的計算量；其他半解析法(Semi-Analytic Solution Method)最少，約佔 10%的計算量。這是由於近年來金融產品的創新，很多都是將權利條件加到原有的金融商品之上，產生截然不同的償付型式。以最近在台灣市場銷售的主流結構產品，都是內嵌權利條件的金融商品。至於在避險基金內，則主要使用於高頻演算法的程式交易之中。

第三節 程式語言

為了要導引電腦工作，我們需要與電腦進行溝通。程式設計師可用各式各樣的程式設計語言來撰寫指令。其中，某些語言是電腦能夠理解的，但有些則需有中繼的轉譯(Translation)步驟，雖然現代有數百種電腦語言，這些語言仍可以分成三種一般的型態：

1. 機器語言
2. 組合語言
3. 高階語言

任何電腦都能夠直接瞭解它自己的機器語言(Machine Language)。機器語言是由電腦硬體的設計所定義，因此是各類型電腦的「自然語言」。機器語言一般是由數字的字串（最後化簡成許多 1 與 0）所組成，指示電腦如何執行其最基本的運作。機器語言是與機器有關的，也就是說，一種特殊的機器語言只能夠用在某種電腦上。

下列片段的機器語言，是將超時工資與基本工資相加，並將結果儲存在總工資。對於人類而言，機器語言是難以理解的。

+1300042774
+1400593419
+1200274027

隨著電腦的普及，機器語言的程式設計對於一般使用者而言就顯得過於緩慢、冗長而且容易發生錯誤。除了使用電腦能夠直接瞭解的數值字串之外，程式設計師開始使用類似英文的縮寫字符，來代表電腦的一些基本操作。這些像英文般縮寫字形成組合語言(Assembly Languages)的基礎。稱為組譯器(Assemblers)的轉譯程式(Translator Programs)，將組合語言程式轉換為機器語言。下列的組合語言程式片段也是用來將超時工資(Overtime Pay)與基本工資(Base Pay)相加，並將結果儲存在總工資(Gross Pay)裡。但對於人類而言，這裡所呈現的步驟，比等效的機器語言要清楚多了。

```
LOAD    BASEPAY
ADD     OVERPAY
STORE   GROSSPAY
```

雖然這樣的程式碼對於人類較為清楚易懂，但是除非轉譯成機器語言，否則電腦並無法理解。

因為組合語言的出現，使得電腦的使用率快速增加，但是即使要完成最簡單的工作，仍然需要對電腦下許多的指令。為了要加快程式設計的過程，人們就發展了高階語言(High-level Languages)，我們只需要單一的敘述式(Statement)就能夠完成基本的工作。稱為編譯器(Compilers)的轉換程式將高階語言轉換為機器語言。高階語言可以讓程式設計師以近似於日常英文的用語，以及一些常用的數學符號來撰寫指令。

使用高階語言所撰寫的薪資計算程式，可能包含下述的敘述式：

```
grossPay = basePay + overtimePay
```

很明顯的，比起機器語言或組合語言，程式設計師偏愛高階語言。Visual Basic 是世界上最為流行的高階語言之一。

將高階語言程式轉成機器語言的編譯過程需要相當長的時間。透過直譯器(Interpreter)程式的開發，這個問題已經獲得解決。這種程式可以略過編譯的過程而直接執行高階的語言程式。雖然已經編譯好的程式其執行速度比直譯的程式還要快，但直譯器在程式開發環境中卻仍很流行。在這些環境裡，開發人員經常要修改程式，例如加入新功能或者更正錯誤。一旦程式完全開發完畢，就會產生編譯好的版本，使程式以最有效率的方式執行。目前市場上廣為流行的 VBA、Python 就是直譯式的程式語言。

一、結構化程式設計

1960 年代，許多大型軟體的開發都遭遇嚴酷的考驗。軟體開發的進度經常落後，成本超過預算，而且最後的產品也不可靠。人們開始認知到，軟體開發遠比人類的想像更為複雜。針對上述問題的研究，結果發展出結構化的程式設計(Structured Programming)—為產生清晰、正確、易於修改的程式，建立所需的規範。

這些研究中最實際的成果之一就是在 1971 年所開發的 Pascal 程式語言。Pascal 是根據 17 世紀的數學家和哲學家 Blaise Pascal 所命名，設計的目的是在校園中教授結構化程式設計，而且在許多大學裡，很快就成為最受歡迎的介紹程式設計的語言。早期在 Windows 下相當流行的 Delphi 語言，以及 Linux 下的 Kylix 語言，便是以 Pascal 語言為基礎的物件導向開發工具。

在高階語言的開發過程中，新的語言都以舊語言的觀點為基礎來發展。C 語言是先從 BCPL 及 B 這兩種語言發展出來。Martin Richards 在 1967 年開發 BCPL 作為撰寫作業系統、軟體與編譯器的語言。在 BCPL 之後，Ken Thompson 使他的 B 語言成型。在 1970 年，Thompson 使用 B 建立早期的 UNIX 作業系統版本。BCPL 與 B 都是「無類別資料型」的語言，這意謂著每一個資料項目都佔據記憶體的一個「字組」。使用這些語言時，程式設計師必須將每個資料項目當作整數或實數加以處理。

在貝爾實驗室的 Dennis Ritchie 根據 B 而發展出 C 語言，並且於 1973 年實

作完成。雖然 C 採用了許多 BCPL 與 B 的重要概念，卻也提供資料類別區分與其他的功能。這是 C 被廣泛地認為是 UNIX 作業系統的開發語言。然而，現在絕大多數的電腦已經可以使用 C 語言，而且今日主要的作業系統也都是用 C 或 C++所撰寫。C 是與硬體無關的語言，而且如果設計得當，可以撰寫出絕大多數電腦都可使用的程式。

雖然目前已經開發出數百種高階程式語言，但只有少數幾種廣為接受。這一節概觀諸如 Pascal、C 語言及下一節將詳述的 Basic 語言，都是存在已久而且廣為使用的高階語言。

IBM 在 1954 到 1957 年之間開發出 Fortran (FORmula TRANslator)語言，以供需要複雜數學計算的科學研究或工程應用，Fortran 至今仍然廣為使用。COBOL (COmmon Business Oriented Language)是在 1959 年由電腦製造商、政府部門和工業電腦的使用者所共同發展出來。COBOL 的主要用途是商業應用，即需要針對大量資料精確且有效率運用的領域。至今仍有相當大的一部份軟體是架構在 COBOL 中。目前仍有新的 COBOL 編譯器被推出使用。

二、物件導向語言

C++是 C 延伸，是在 1980 年代早期的貝爾實驗室，由 Bjarne Stroustrup 使用 Simula 67（一種程式設計模擬語言）的元素所開發出來的。C++提供許多的功能，使得 C 語言更為豐富，但更重要的是，這種語言具有「物件導向程式設計(Object Oriented Programming)」(OOP)的能力。

當新穎而強大的軟體需求急速增加的同時，以快速、正確而經濟的方法建立軟體仍然是難以實現的目標。然而，增加物件(Objects)或者軟體元件(Components)的重複使用，可以解決其中一部份的問題。所謂的物件可比擬為現實生活中的“東西”。軟體開發人員逐漸發現，採用一種模組化—物件導向的設計和實作方式—比先前最常使用的程式設計技術，即結構化程式設計，更能夠提高軟體開發小組的生產力。而且，物件導向的程式通常更容易瞭解、更改和修正。

除了 C++以外，市場上也出現了其他物件導向語言。這些語言有全錄的 PARC 研究中心(Palo Alto Research Center)所建立的 Smalltalk。以及昇陽公司推出 Java 語言。在 2000 年，微軟發佈由 Anders Hejlsberg 與 Scott Wiltamuth 所開發的 C#程式設計語言及其.NET 策略。這種語言是由微軟特別為.NET 平台所設計的語言。它根植 C、C++與 Java，並從這些語言各自的最佳功能加以改寫。C#是物件導向的，而且包含已預先建立好，由元件所組成強大的類別庫，使程式設計師可以快速開發應用程式。

另外，目前有支援 GPU 功能的程式語言，包括 C/C++、C#、Fortran、Java 等四類。其中，C/C++是原生支援。C#、Java、Fortran 則是透過 Wrapper，1 對 1 叫用 GPU 的內建函數。至於 Python、R 等語言，則是將 GPU 的內建函數包裝起來，包裹叫用。這自然也就反應使用執行的效率了。

第四節 Visual Basic 與 VBA

雖然 BASIC(Beginner's All-purpose Symbolic Instruction Code)語言一向被視為簡單的入門語言，但是在微軟將之整合到 Office 之後，卻成為商業自動化中一個重要的應用工具。尤其，Excel 因為具備大量的資料計算與分析的能力，以及方便的表報呈現功能，使得它在現今的交易室中，成為不可或缺的一員。VBA 可說是財工人員最為上手的方便工具，透過它的使用，讓財務計算可以被普遍地執行。

一、Basic 與 Visual Basic

BASIC 是在 1960 年代中期，由達特茅斯學院(Dartmouth College)的 Jonn Kemeny 教授與 Thomas Kurtz 教授所開發，用以撰寫簡單的程式。BASIC 的主要目標就是要讓初學者熟悉程式設計的技術。在各種型態的電腦〔有時也稱為硬體平台〕中受到廣泛使用的 BASIC，已經使得程式語言的效能大幅提昇。當比爾・蓋茲建立微軟公司時，他在許多早期的 PC 上實作了 BASIC。早期在 PC 中的 BASIC 程式，基本上是直譯的程式。由於在 1980 年代的晚期與 1990 年代

早期，微軟 Windows 圖形使用者介面(Graphic User Interface, GUI)的發展，BASIC 很自然地演變成微軟在 1991 年所引介的 Visual Basic。

直到 1991 年 Visual Basic 出現以前，開發以 Windows 為基礎的應用程式是困難而且麻煩的過程。雖然 Visual Basic 是由 BASIC 程式語言發展而來，但是兩者是完全不同的程式語言。VB 提供許多強大的功能，例如圖形使用者介面、事件處理、使用 Win32 應用程式設計介面、物件導向的程式設計，以及錯誤處理。為了加速程式的執行效率，VB 也提供編譯成可執行檔的功能。

VB 是事件驅動、視覺化的程式設計語言，必須使用整合開發環境(Integrated Development Environment, IDE)加以建立。使用 IDE 時，程式設計師很容易便能夠針對 Visual Basic 程式進行撰寫、執行、測試與偵錯。因此，使用 IDE 製造出可運作的程式所需的時間，是不使用 IDE 所需發費的時間的一小部分。這種快速建立應用程式的過程，通常稱為快速應用程式開發(Rapid Application Development，RAD)。Visual Basic 是世界上最廣為使用的 RAD 語言。

在微軟為其新一代網路架構推出的新平台.Net Framework 中，Visual Basic 已經演化成一個全新的 OOP 語言，VB.NET。另外，Visual Studio(VS)也已經進化成功能強大的 IDE。VS 已經把 C/C++/C#、Visual Basic、Python 整合到同一開發平台。

二、VB 與 VBA

若簡單地來解釋 VBA 的話，它是一種可以用來和 Excel 進行交談，然後讓 Excel 按照我們的指定，來進行操作的語言。這種語言基本上是以 Visual Basic 為基礎，配合 Excel 的物件結構而設計出來的。VBA 基本上是直譯的程式，必須在 Excel 下才可執行。早期這種操弄應用程式的語言，一般與程式語言並不相同，因此以巨集(Macro)稱呼之。然而，雖然 VBA 已與早期的巨集不同，反而成為一種程式語言，但是基於歷史因素，有時仍以巨集稱呼之。因此，要操作 Excel 時，我們便是使用這種能夠與 Excel 進行交談的「VBA」語言來建立出巨集，然後要求 Excel 來進行各式各樣的操作。

在過去的 25 年中，VBA 的發展過程大略如下：

年代	Excel 版本	VBA 版本	說　　明
1993	Excel 5.0	VBA 1.0	Excel 5.0 是 Office 組合內第一個使用 VBA 作為程式開發工具的應用程式
1995	Excel 7.0 (Excel 95)	VBA 2.X	
1997	Excel 8.0 (Excel 97)	VBA 5.0	在 Office 97 組合內已陸續使用 VBA 作為其程式開發工具
1999	Excel 9.0 (Excel 2000)	VBA 6.0	
2002	Excel 10	VBA 6.5	
2003	Excel 11	VBA 6.5	
2007	Excel 12	VBA 6.5	
2010	Excel 14	Excel 7.0	推出 32 與 64 位元兩種版本
2013	Excel 15	VBA 7.1	
2016	Excel 16	VBA 7.1	

Excel 2000 之後的產品，基本上都是配合使用 VBA 6.0 的版本。VBA 6.0 是一個成熟而且廣為市場接受的工具。另一個比較重大的改變是，自 VBA 7.0 開始，分為 32 與 64 位元兩種版本。這個變動反應電腦硬體的進步，與作業系統全面向 64 位元發展的態勢。如果我們要進行一些系統函數的呼叫，可能要注意電腦使用的版本，甚至調整 32 位元版本的程式，才可以在 64 位元的系統執行無誤。不過在本書的範例中，讀者是無需擔心的，因為我們的程式並不須要執行這些功能。

VBA 是一種專門用來建立巨集的語言，它從 Excel 95 的時代便開始內建在 Excel 之中。VBA 的全文為「Visual Basic for Application」，是微軟公司以程式開發語言「Visual Basic」為基礎，然後將它修改為應用程式的巨集語言。微軟在 Excel 之中引進了應用程式的 Visual Basic，而這就是「VBA」了！

目前，除了微軟 Office 中的應用程式，如 Words、Access、Power Point、

Outlook 都有共同的巨集程式開發工具－微軟 Visual Basic for Application，甚至其他的軟體廠商也都盡量取得微軟的 VBA 認可，採用 VBA 來作為其應用程式的巨集開發工具，目前大約有多家的 ISV 已取得 VBA 的認證，例如 VISIO、AutoCAD。

早期的 Excel 2000 使用 VBA 6.0 做為程式開發工具，而 VBA 6.0 擁有一個整合式的開發環境，我們可以在這個獨立的視窗內看到專案總管視窗、屬性視窗、程式碼視窗等。VBA 6.0 中有豐富的事件處理能力、除錯能力、以及可使用 ActiveX 控制項的表單(UserForm)等。當然，現在市場上 Excel 2016 使用的 VBA 7.5，更是將這些功能進一步擴充，讀者可以感受到 VBA 的強大功能。本書將以 Excel 2016 做為開發的平台，說明使用的方式。

第五節 Excel 的操作環境簡介

要使用 Excel VBA 當然要先啟動 Excel 的應用程式，如圖 1.5.1 所示。本書是以 Execl 2016 做為操作對象。

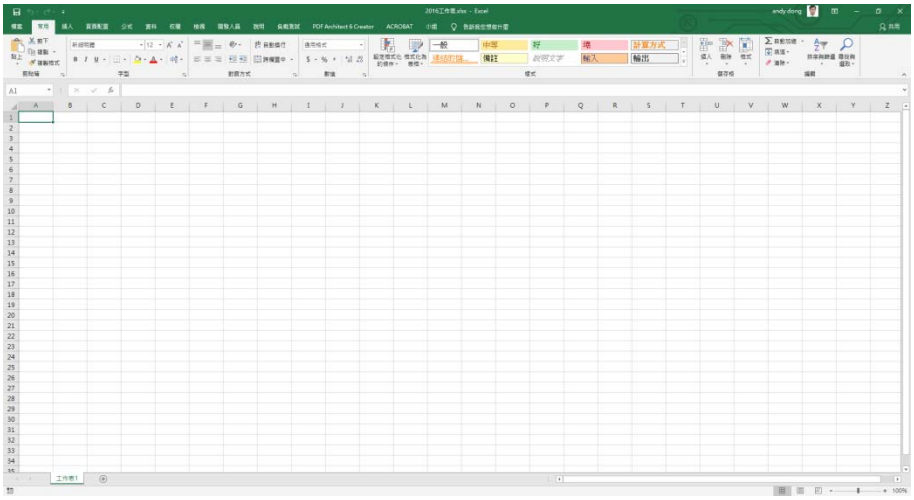
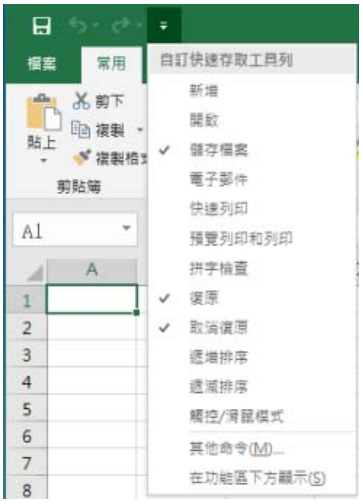


圖 1.5.1 Excel2016 的操作畫面

針對 Excel 2016 的使用介面，我們以下圖的名稱方式稱呼之。

最上方有快速存取工具，如下圖顯示的工具，使用者可以自訂。



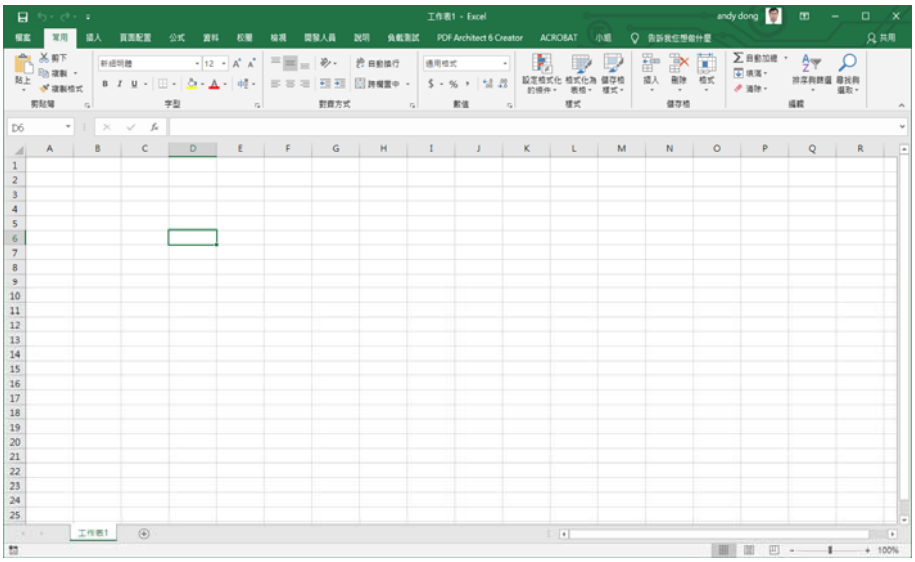
其次，是功能頁次如下圖。

接下來是，【常用功能頁次】的功能區。

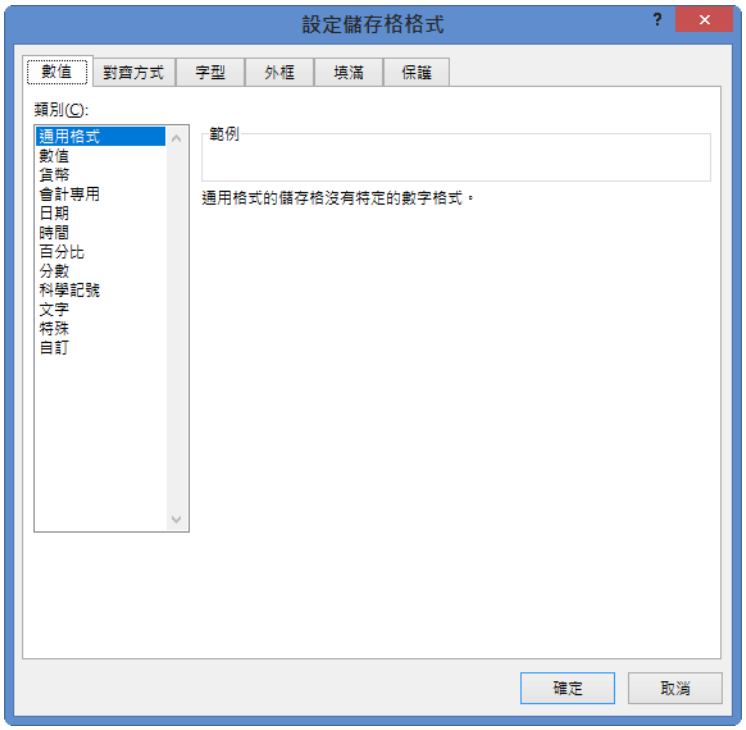
在【儲存格功能區】中的【格式按鈕】下有下圖選項。

一、儲存格

提到了儲存格，但何謂「儲存格」呢？儲存格是用來儲存資料的位置，在 Excel 最重要是我們必須對數字作運算，所以必須運用有類似表格格子的位置，此謂「儲存格」。如圖 1.5.1，縱、橫相交的直線，將整個螢幕分隔成許多的小格子，每一個格子便是一個儲存格。在下圖中，我們選到"D6"這個格子。



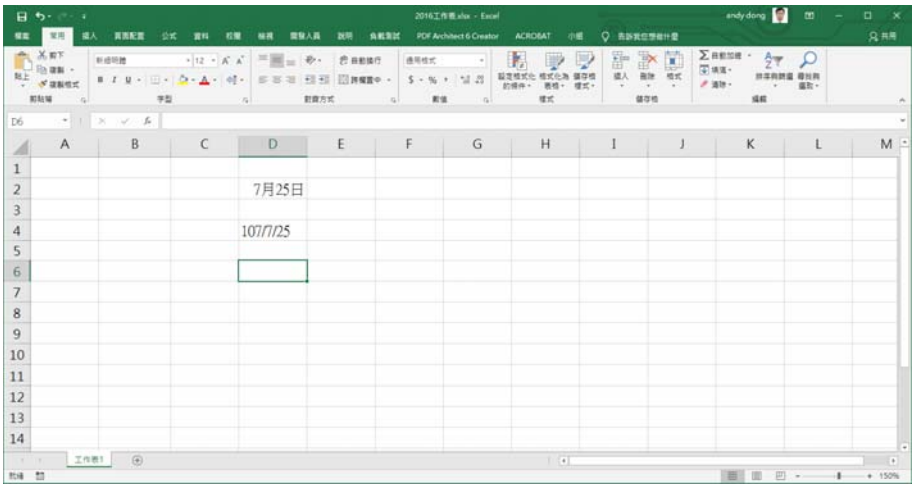
儲存格的數字類別有許多種，讀者可以點選功能選單中的【常用頁次／儲存格區／格式鈕／儲存格格式項】，便會出現下面視窗。



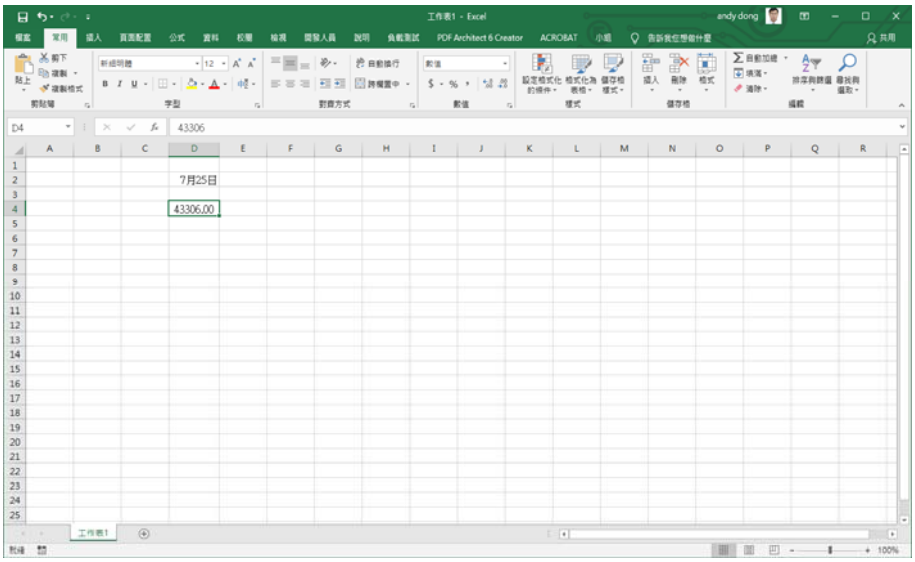
輸入數字的內容於儲存格內就會以設定的類別來呈現，例如：

(一)今年是民國 108 年(即電腦的系統年份)，只要鍵入月及日就可如：“7/25”就會以 2019/7/25 儲存，且以 7 月 25 日 出現在鍵入的儲存格（已設定好日期的類型）內。

(二)若是要輸入 107 年 7 月 25 日，則只要鍵入“107/7/25”，也就是【年/月/日】的電腦格式，且以 107 年 7 月 25 日 出現在鍵入的儲存格內。



然而，有時輸入“2018/7/25”卻出現一串數字，如 43306.00，這又是什麼原因呢？這是因為該儲存格的類別被設定在數值上，因此不以日期格式顯現。在 Excel 中，以 1900/1/1 零時，當成相對日期的起始點，433060 代表 2018/7/25，為 1900/1/1 後第 42210 日，因此 43306 便是以序列值顯示的日期表示方式。每往後一日，序列值便增加一。



選取數值類別，便可以看到 43306.00 的表式方式，如下圖所式。至於每一類別的資料格式，擇摘要如表 1.5.1 所示。

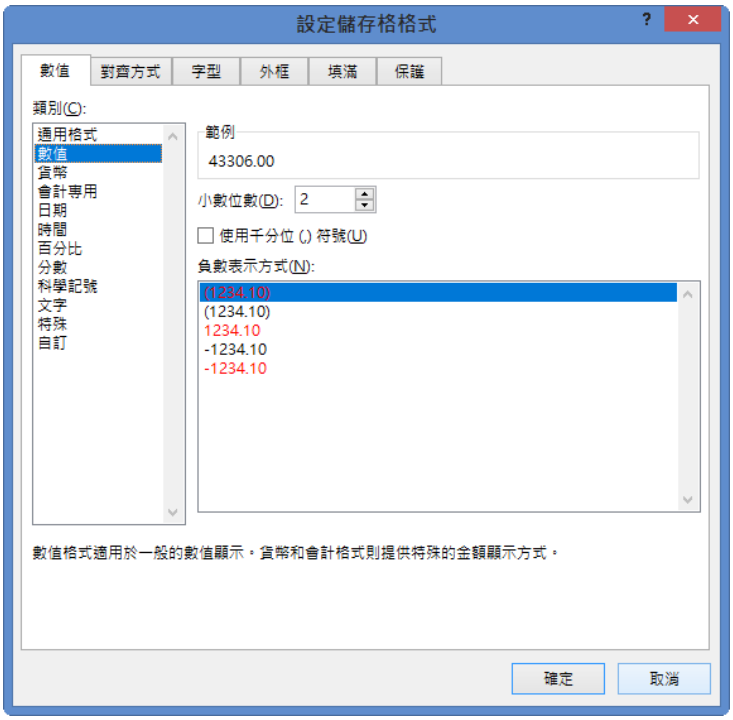


表 1.5.1 儲存格的數字類別

數字類別	說明例子
G/通用	沒有特定為一般數字，是為預設值。
數值	為小數點二位的，如 1234.10，-1234.12。
貨幣	有 $ 符號、千分位符號且具有小數點二位的數字，如$1,234.10。
會計專用	有 $ 符號、千分位符號且具有小數點二位的數字，如$1,234.10。會將同欄數值的貨幣 $ 符號與小數點對齊，並且其對齊方式與貨幣類別不同。
日期	輸入簡易日期如 88/9/25，即可轉為 88 年 9 月 25 日等。
時間	輸入簡易時間如 13:35:55，即可轉為 13 時 30 分 55 秒等。
百分比	會將儲存格的值乘以 100，並且顯示值加百分比的符號。
分數	以分母為設定，如 1/2，2/4，3/6。
科學記號	具有小數點二位的數字，如 3.33333E+12。
文字	會將數值設為文字，如 002586，0112-568-20。
特殊	郵遞區號、支票金額如：壹萬貳仟參佰肆拾伍。
自訂	設定自訂代碼之鍵入數字格式代碼，如 US$1,234.35。

二、工作簿、工作表

Excel 檔案我們稱之為「活頁簿」(WorkBook)，而活頁簿中有一張張的工作表，「工作表」(Sheet)就如書中的「頁」，而工作簿就如書。一本書中可以有很多頁，一個工作簿中當然也可以有很多工作表，開啟的工作簿中預設有一個工作表。

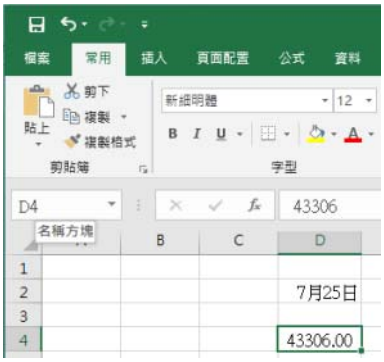
「工作表」是由欄和列所組成的。「欄」(Column)是直的，由 A、B、C...至 XFD，共有 16,384 欄。而「列」(Row)是橫的，由 1、2、3...至 1,048,576 列

。所以共有 16,384x1,048,576 個儲存格於一張工作表中，而每一個儲存格最多可輸入 32,767 個字元。雖然工作表可以有那麼大的範圍，實際上並沒有那麼多。

儲存格也可以輸入公式，所謂公式就是以等號“=”作為開始，用來對儲存格參照運算作用的字串，但是文字最多只能有 8,192 個字元。所謂的儲存格參照，就是方才提到「儲存格」(Cell)是利用「欄」和「列」的交叉點所組成，所以每一個儲存格都有座標位址名稱，例如：A1、B3、D9……等等。儲存格既然有位址名稱，就可以用來作運算，例如：“=A1+B1”，“=SUM(A1:C10)”等等，而這些輸入就是公式。

三、作用中儲存格

作用中儲存格資料，是會顯示在資料編輯列中。所謂「作用中儲存格」就是目前目標中的儲存格，也就是現在目光焦點選取的儲存格，在眾多的儲存格中是以黑框框住來標記的。然而又如何成為選取的作用中儲存格呢？也就是當滑鼠游標為粗白十字狀時，以滑鼠左鍵按一下作為點選，其選取的儲存格就以會黑框框住，在其右下角會有一小黑方塊，並會在左上方【名稱方塊】上出現它的欄名列號的儲存格座標位置，例如 ”D4”，如圖 1.5.2 所示。



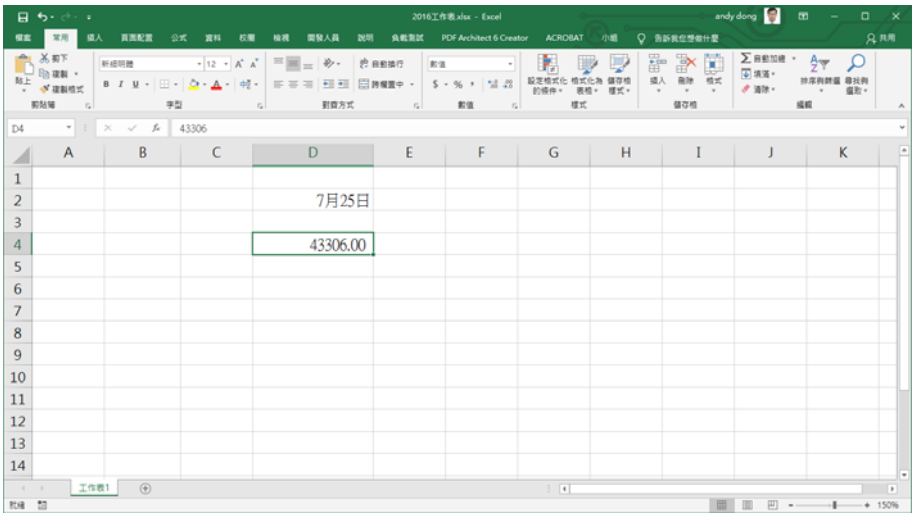


圖 1.5.2 作用中的儲存格

當儲存格於作用中，尚不足以輸入，必須將插入點移至所要輸入的位置才可，如圖 1.5.3 所示，移入插入點的方式有兩種：

第一種：對儲存格快速用滑鼠左鍵連按兩下，其插入點會在儲存格上出現。

第二種：用滑鼠直接在資料編輯列按一下，其插入點會在資料編輯列上出現。

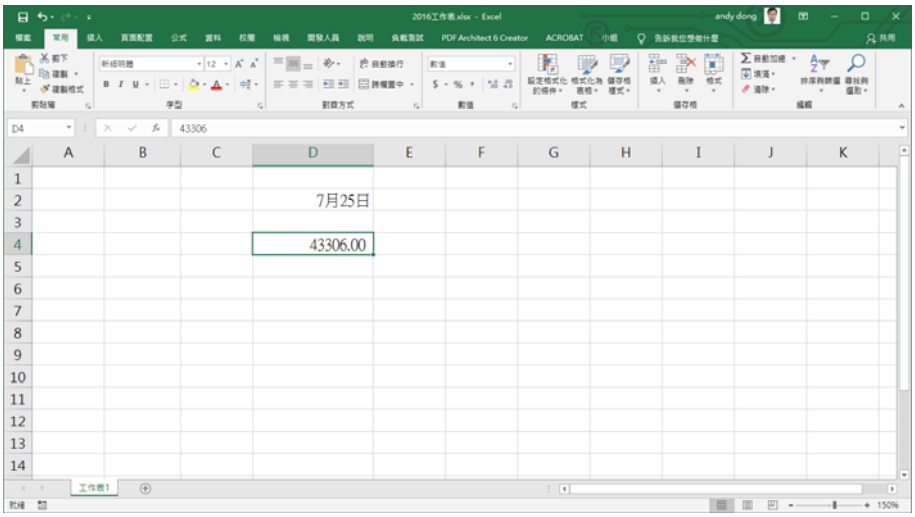
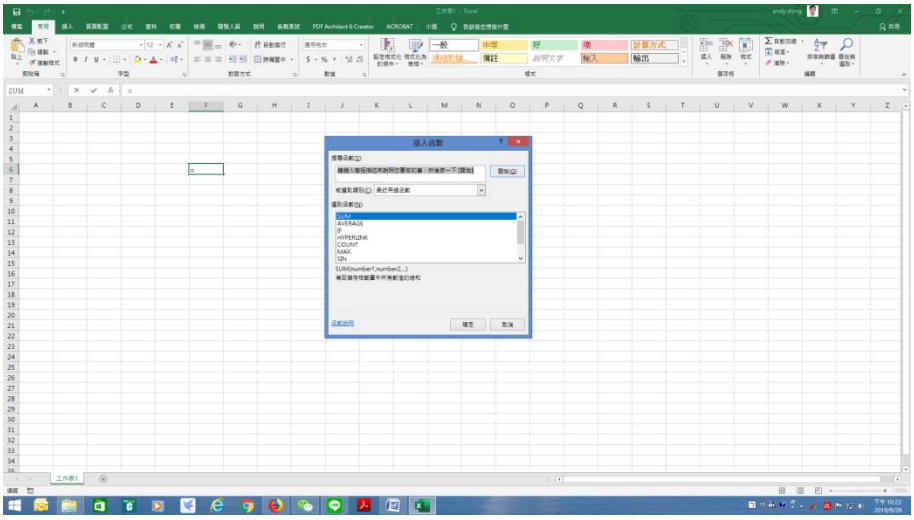


圖 1.5.3 儲存格的輸入

資料編輯列左方有 3 個按鈕，中間 ☑ 鈕代表輸入電腦，其功用就和按 ENTER 鍵相同；左邊 ☒ 代表取消輸入，也就是放棄方才輸入其功用就和按 ESC 鍵相同；右邊 *fx* 代表公式的按鈕，按下之後會出現＝號，另外會跳出視窗，提供可供選擇的函數，以及搜尋功能，如圖 1.5.4 所示。



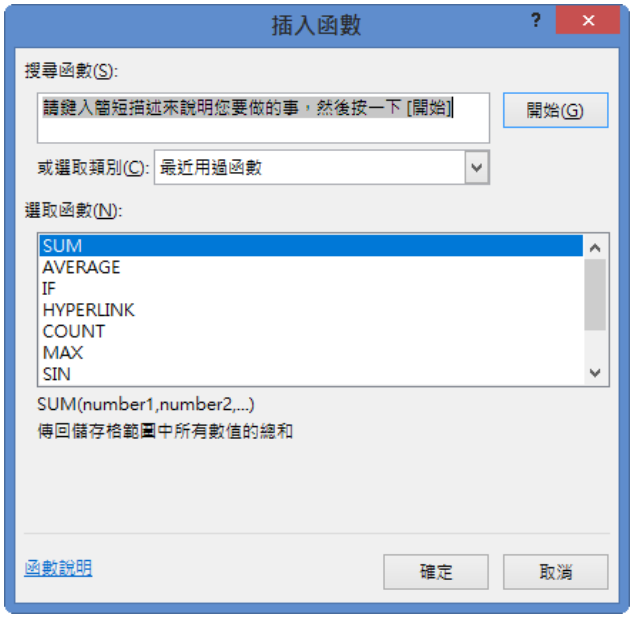


圖 1.5.4 【選取類別】出現公式清單

在圖 1.5.4 中用滑鼠左鍵對【選取類別】右方的箭號鈕按一下，即出現較常用的函數可供選擇。在圖 1.5.5 中用滑鼠左鍵對【選取類別】右方的箭號鈕按一下，選擇全部，即會出現所有可用函數，提供選擇。

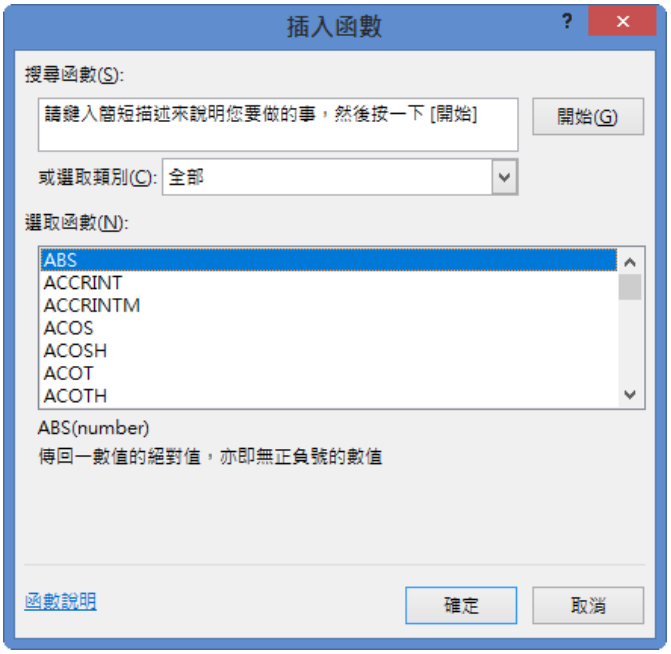


圖 1.5.5 【選取類別】出現公式清單

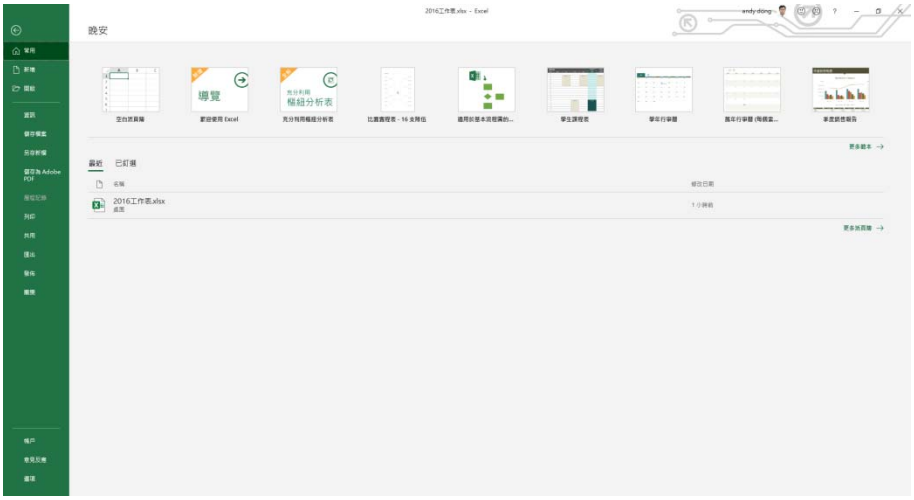
第六節 VBA 的巨集錄製

由於 Excel 2016 的介面明顯與早期的 Excel 2000 不同。從 Office 2007 之後，微軟導入了「Ribbon 圖形用戶介面」取代傳統選單與工具列介面。內建於 Excel 2016 之中的 VBA，擁有與 Visual basic 進乎相同的功能。而且隨著 VBA 引進了採用最新技術的「ActiveX 控制項」後，透過 VBA 您不僅能控制 Microsoft Office 的所有功能，同時您也可以進一步地再來控制 Windows 的一部份功能了。

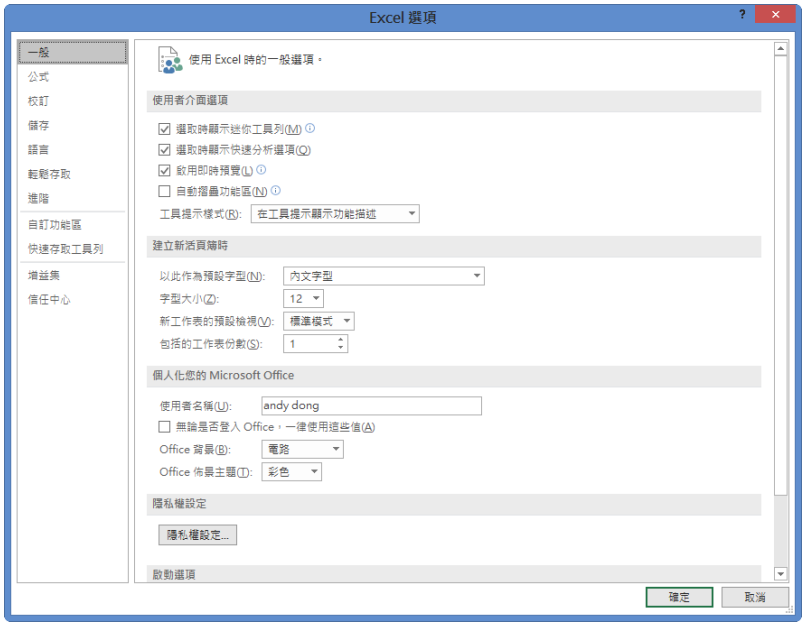
儘管 VBA 擁有強大的功能，但在另一方面它仍兼備了 BASIC 語言簡單易學的優點。同時它還提供了一個專屬的編輯環境與輸入輔助功能，這可讓您更

輕鬆地來建立出巨集。此編輯環境稱之為 Visual Basic Editor，簡稱 VBE。下面利用巨集錄製功能，為您介紹 Excel 2016 VBA 所擁有的一些功能與工作環境。

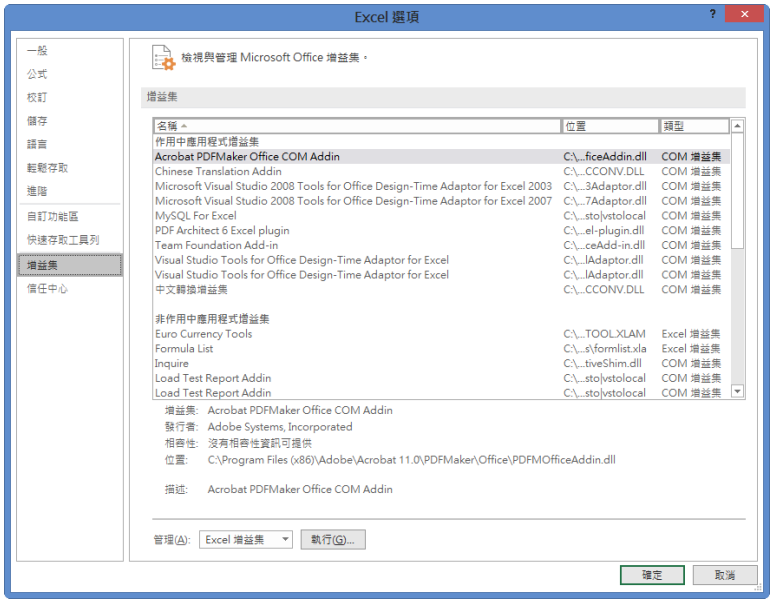
首先，選取左上方【檔案】頁次，出現下面圖案。



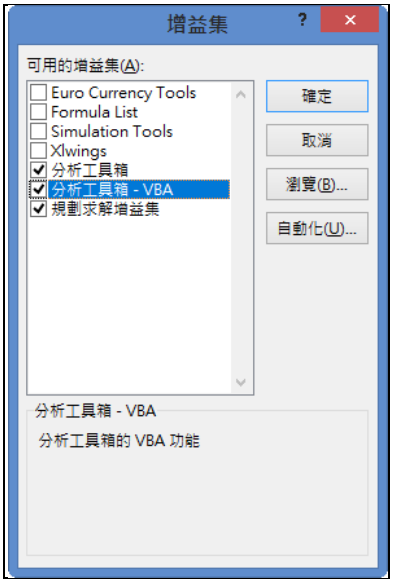
然後，選取左下方【選項】，出現下面視窗。



按取左方【增益集】，出現右方面板，下方有管理 Excel 增益集 執行(G)，

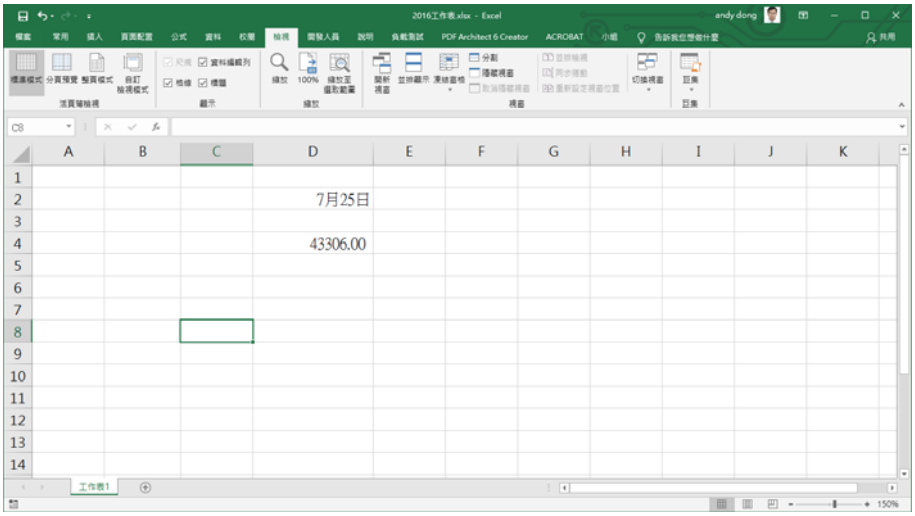


按下管理 Excel 增益集 執行(G)，出現下面畫面。告訴我們引用了多少增益集。



一、巨集的錄製

按下選單中【檢視】頁次，在最右方有【巨集】功能區。



按下【巨集】功能，出現錄製聚集(R)等選項，如下圖。Excel 的巨集錄製器使用起來就如同利用電話答錄機來操作，要使用電話答錄機之前必須要先預演和準備錄音的內容，當有電話進來的時候就可以啟動歡迎的內容。而這些操作的程序都是固定的，縱使您打了電話之後而不聽電話的連線仍然會繼續播放，因為這些動作都是自動而且固定的，在 Excel 中可將這些過程經由按鈕來啟動。

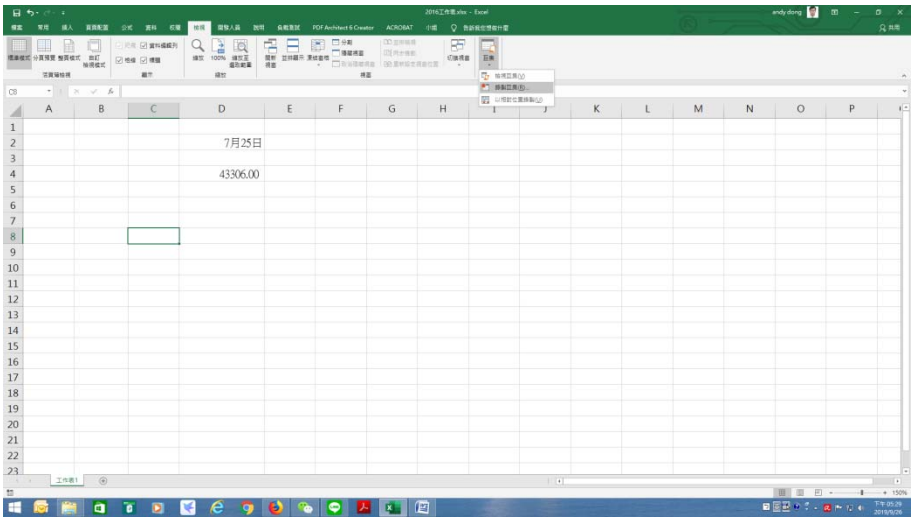
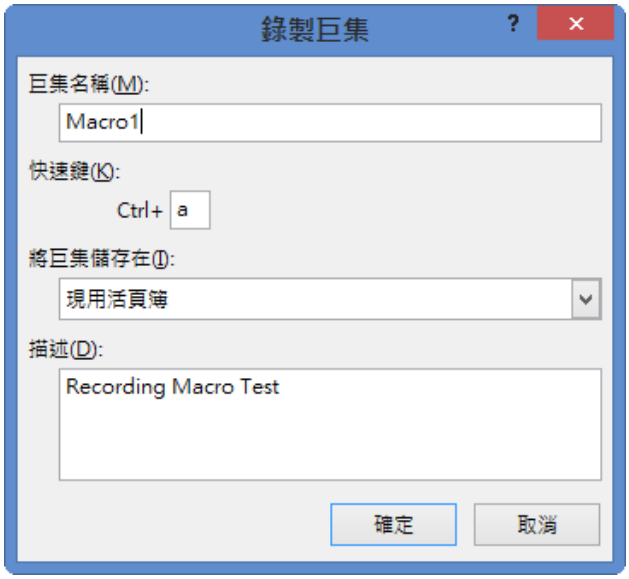


圖 1.6.1 錄製新巨集

下面的實例演練將示範如何使用 VBA 的巨集錄製和查看所錄製的內容。

1. 首先開啟 Excel ，再執行功能表指令【檢視／巨集／錄製新巨集】，如圖 1.6.1 所示，就會出現下面的對話方塊。



2. 在快速鍵下方空隔內輸入“a”，按【確定】鈕，在狀態列就會出現【錄製巨集中】的資料如圖 1.6.2 所示。

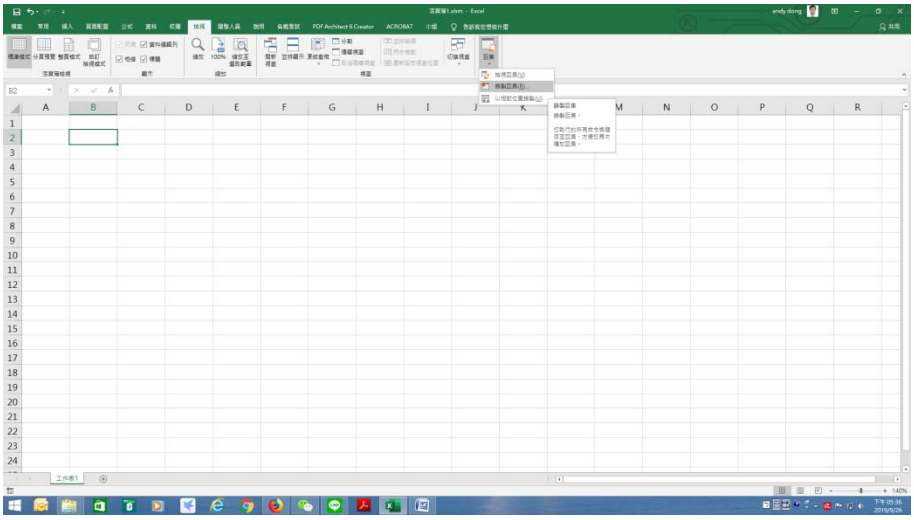
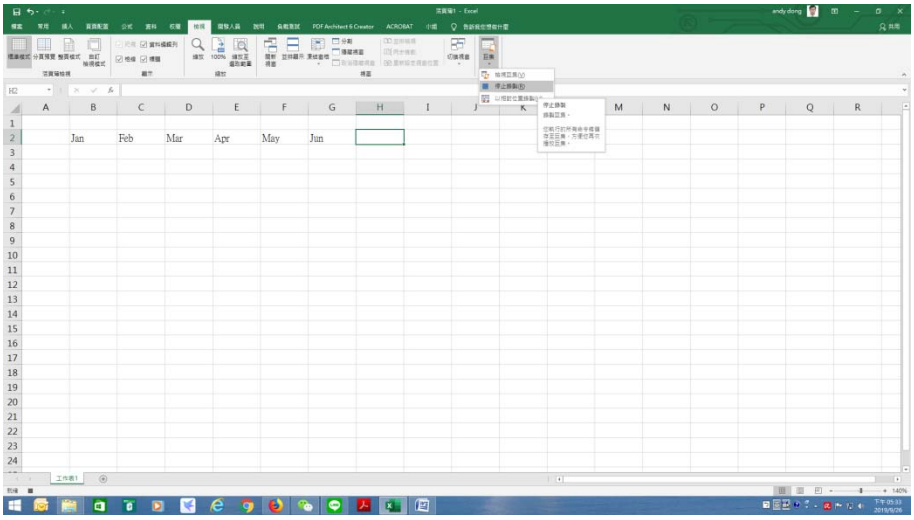


圖 1.6.2 【錄製巨集中】的畫面

3. 點選B2輸入“Jan”，如圖1.6.2依序在C2輸入“Feb”、D2輸入“Mar”、E2 輸入“Apr”、F2 輸入“May”、G2 輸入“Jun”，然後按浮動按鈕中的方形按鈕，如下圖停止錄製。



4. 執行【檔案】內的【另存新檔】將檔案儲存為 Book1.xlsm，再執行【檢視／巨集／檢視巨集】就會出現【巨集】對話方塊，如圖 1.6.3 所示。

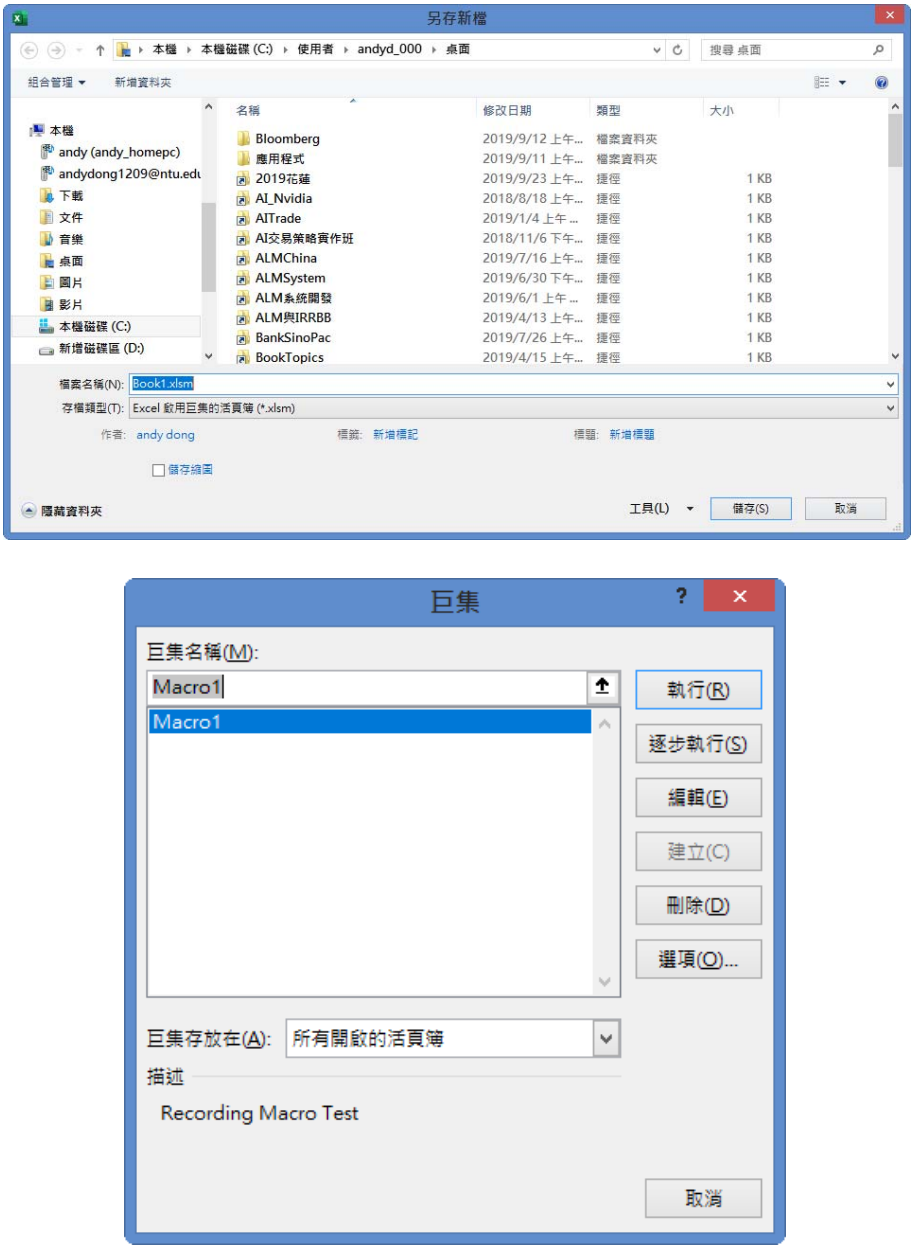


圖 1.6.3 【巨集】對話方塊

5. 在圖 1.6.3 中就可看出目前已經有方才所錄製完成的巨集。選取【Macro1】再按【編輯】鈕，就又進入 Excel VBA 編輯器中，並顯示出方才所選取的巨集之內容，如圖 1.6.4 所示。

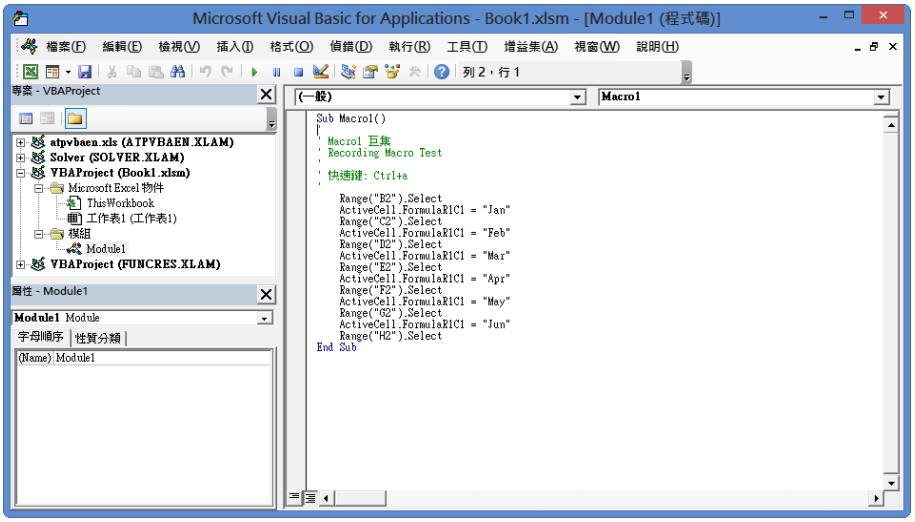


圖 1.6.4 Excel VBA 編輯器的操作畫面

6. 在上述的程式碼中，可看出整個操作的步驟都已被錄製下來，並且轉換為程式碼，而這些程式碼都是 VBA 的程式指令，預設是應用在 Excel 應用程式下，所以不用再宣告應用程式的名稱。不同版本的 Excel 程式碼會有所差異，但不影響執行的結果。

7. 在 VBA 的編輯環境中，執行【檔案／關閉並回到 Microsoft Excel】，或按左上方 Excel 小圖像，回到 Excel 視窗。將 Sheet1 中輸入的資料全部刪除，再按快速鍵 Ctrl + a 發現之前刪除的資料又全部出現。

8. 也可由【檢視／巨集／檢視巨集…】就會出現【巨集】對話方塊，選取【Macro1】，再按【執行】鈕來執行巨集。

二、試算表與 VBA 編輯器的切換

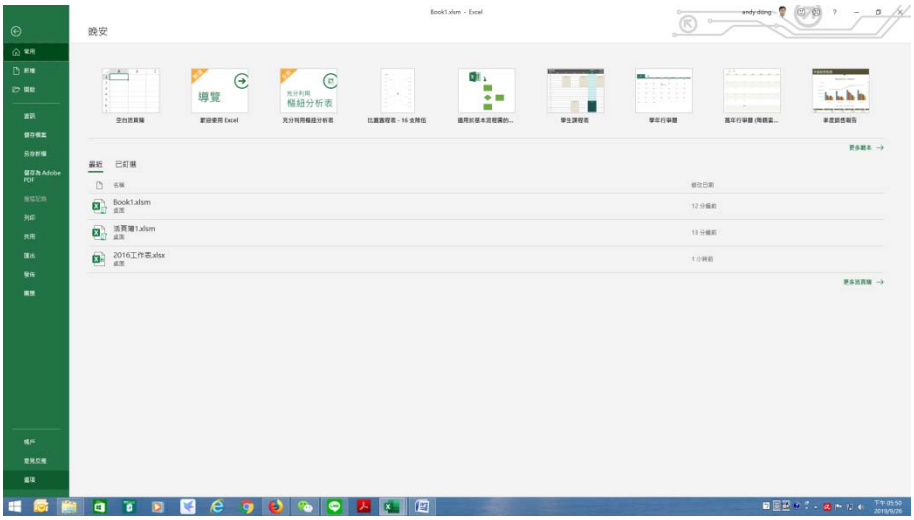
在上一小節中已經利用編輯巨集的方式進入到 Visual Basic 的編輯器中，而上一小節所產生的程式碼基本上是由 Excel 來產生的，若是您對 VBA 的程式語言夠了解，也可以直接來進行編修，「VBA 編輯器」(Visual Basic 編輯器視窗)是在獨立的應用程式視窗中所執行的，因此和 Excel 的應用程式是分開執行的。

啟動 VBA 編輯器有多種方法，說明如下列所示：

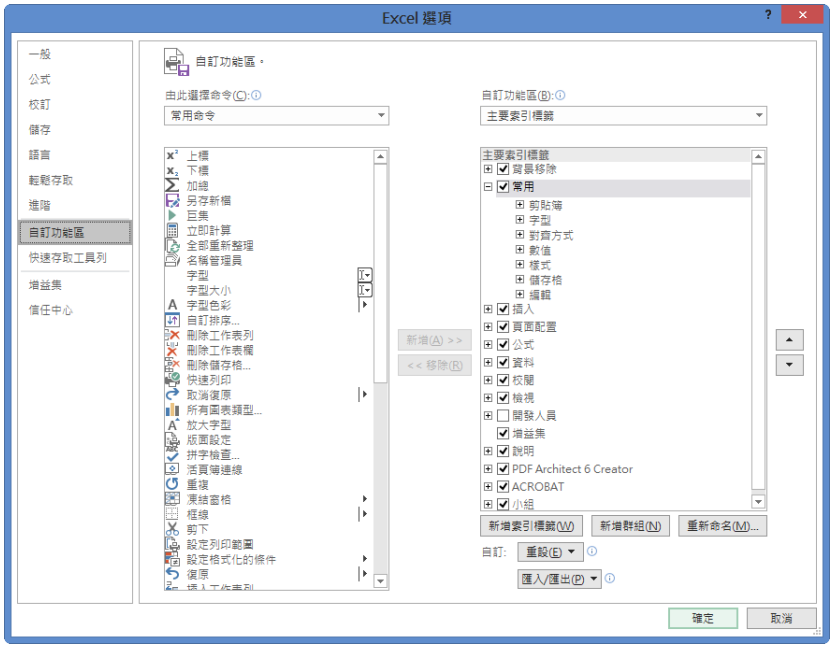
- 按快速鍵 Alt + F11 。

■ 由功能表頁次【開發人員】的方式來進入【VBA 編輯器】。如果在功能區中沒有看到【開發人員】頁次，需如下處理。

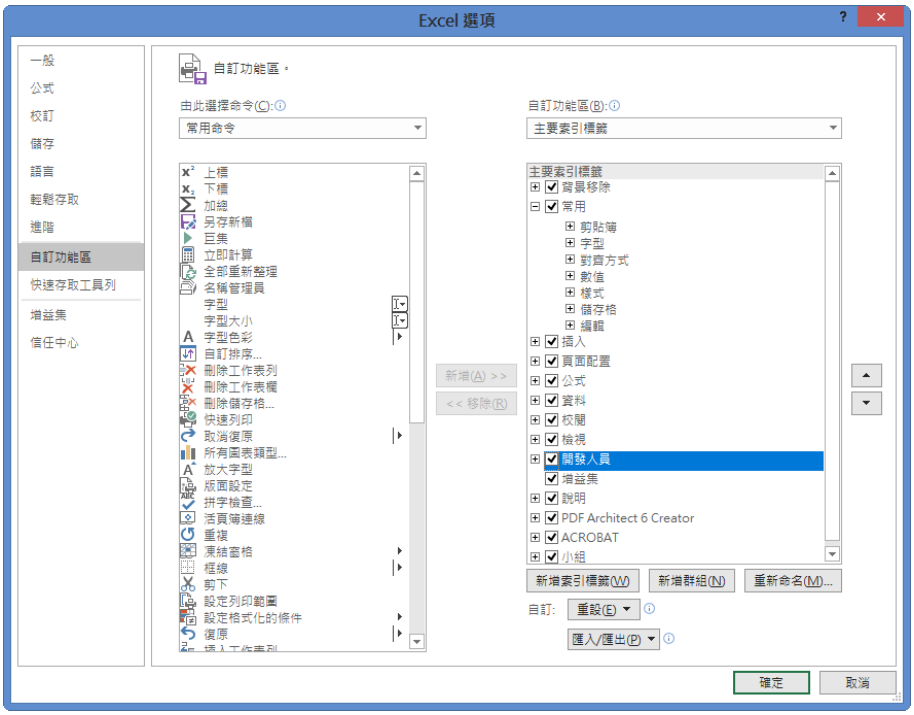
首先，由點選【檔案】頁次如下，選取左下方【選項】。



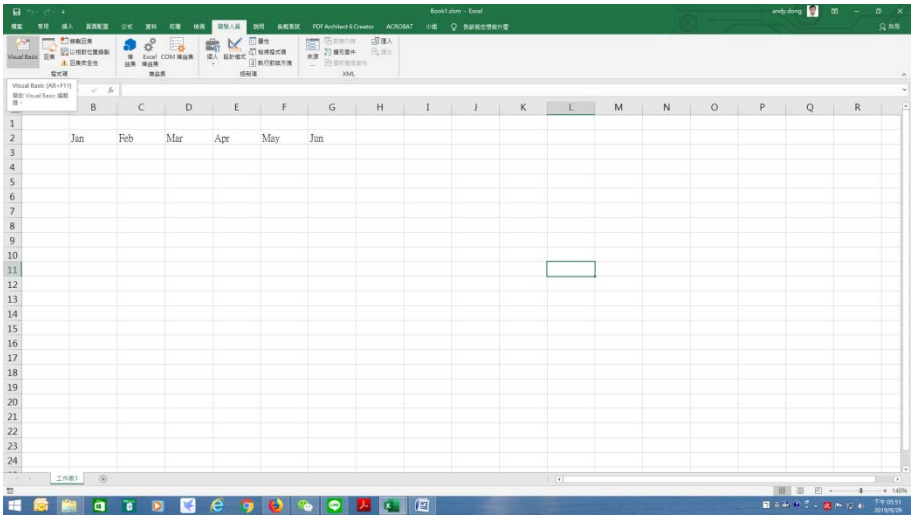
其次，在右下方主要索引標籤內，點選開發人員項次。



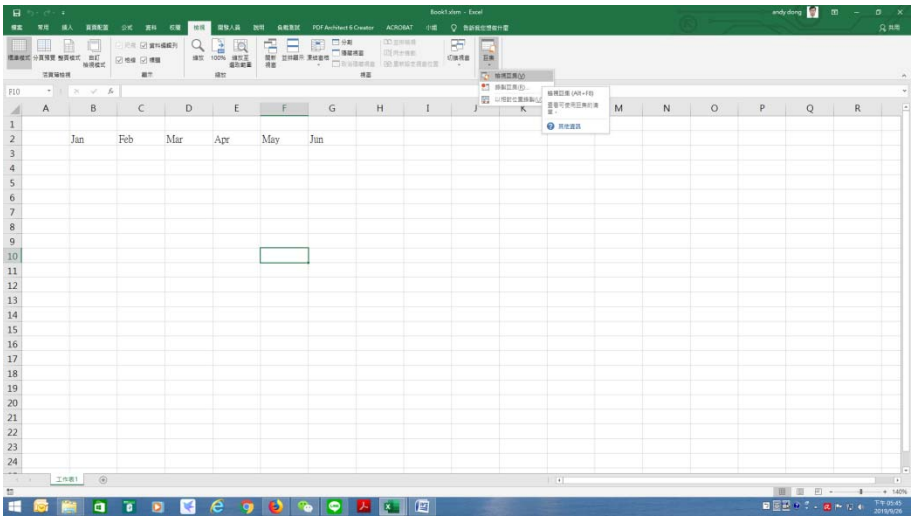
點選開發人員項次後如下。



便可以看到【開發人員】頁次。

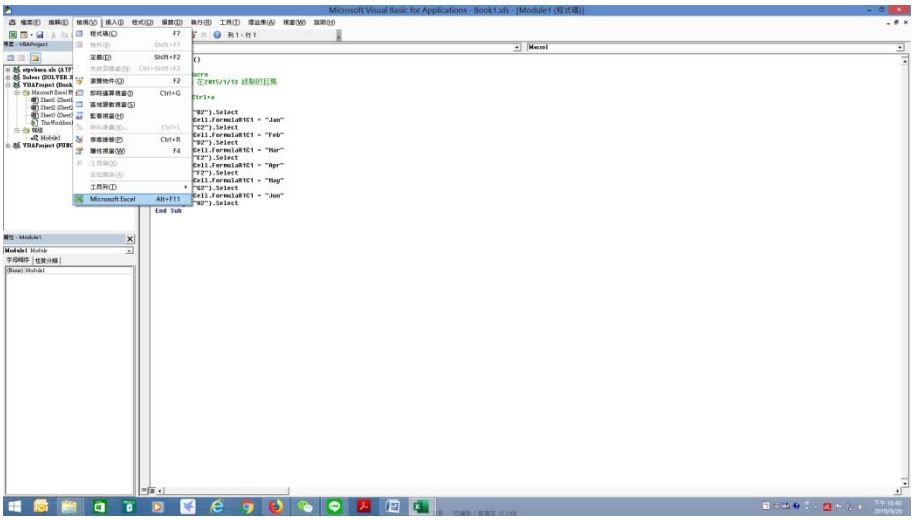


- 執行功能表指令【檢視／巨集／檢視巨集…】，再利用編輯巨集的方式來進入【VBA 編輯器】。

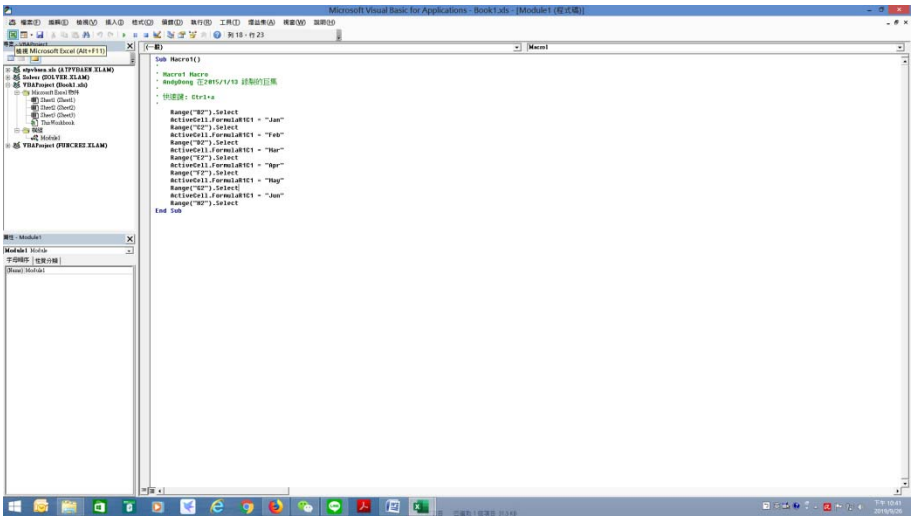


至於由 VBA 編輯器回到試算表，也有下面許多的方式：

- 按快速鍵 Alt + F11 。
- 點選功能表指令【檢視／Microsoft Excel】。



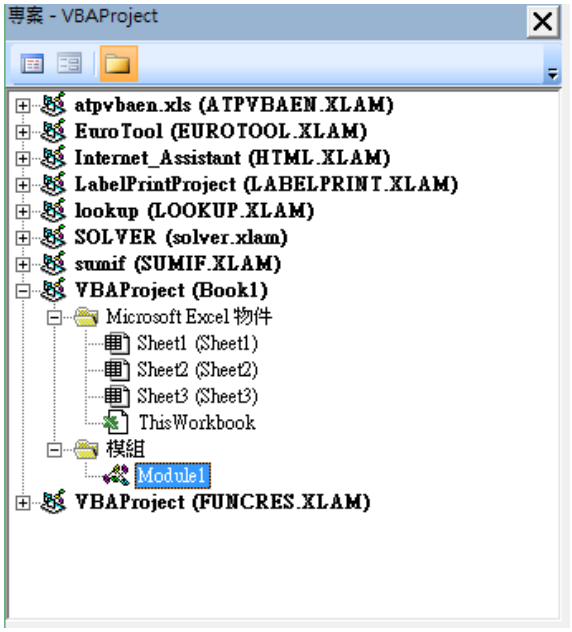
- 可直接按工具列左上方有 Excel 圖示的【檢視 Microsoft Excel】按鈕。



錄製的「巨集」都是放置在模組的程式碼中，也就是屬於《模組》的分類，這可由圖 1.6.4 左上方的視窗中看出。左上方的視窗稱之為《專案總管》，左下方稱之為《屬性視窗》。右方的視窗就是放置錄製的程式碼的《程式碼視窗》。下面逐一說明。

三、專案總管

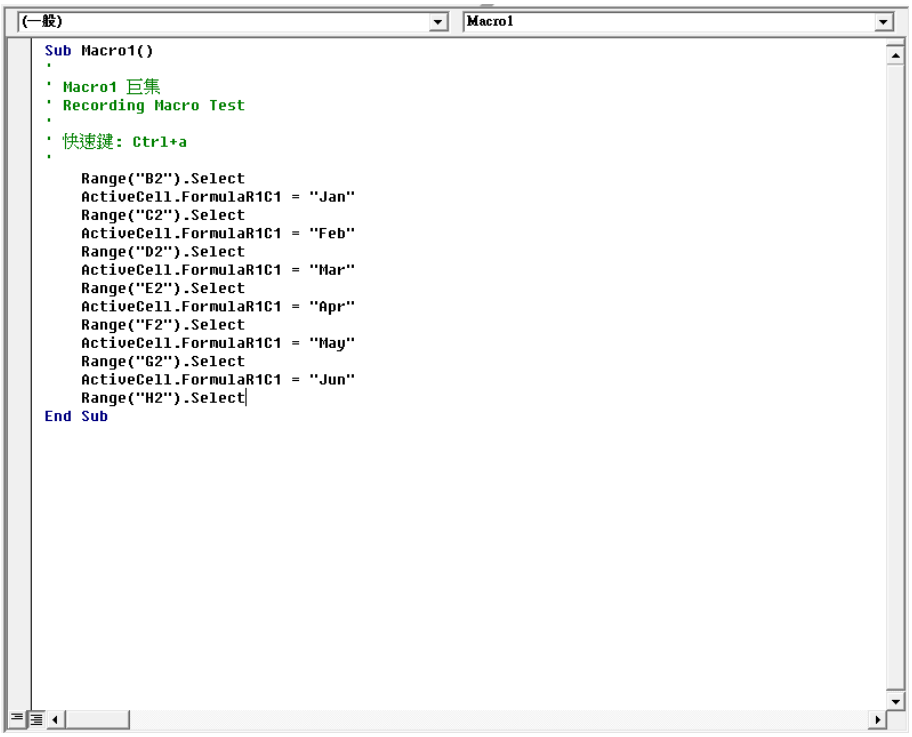
專案總管視窗中放置了我們工作的活頁簿，Book1.xlsm。以 Excel 的稱呼而言，一個活頁簿就是一個專案《VBAProject(Book1.xlsm)》，下面包含了一些成員，以上圖而言有《Microsoft Excel 物件》與《模組》兩大類。《Microsoft Excel 物件》又包含《Sheet1(Sheet1)》、《Sheet2(Sheet2)》、《Sheet3(Sheet3)》以及《ThisWorkbook》四個成員。事實上 Sheet1、Sheet2、Sheet3 這三個成員就是我們一開起 Excel 就自動產生的試算表。至於 ThisWorkbook 則不是一個我們直接可以看到的成員。



至於《模組》下則有《Module1》一個成員，和《ThisWorkbook》一樣，它不是一個我們直接可以看到的成員。但是，在 VBA 編輯環境下，我們是可以看到它的存在。上圖中《Module1》的反白，顯示它被我們點選到，因此程式碼視窗中顯視的就是《Module1》的程式碼，如下圖。讀者可以點選其他《Microsoft Excel 物件》下的成員，一樣會出現它們的程式碼視窗，但是應該都是空白的。

我們也可以自行產生模組下的成員，以便放置自行輸入的程式碼。作法為點選功能表【插入／模組】，此時《模組》下便會產生《Module2》新成員。另外也可用滑鼠先選取《VBAProject(Book1.xlsm)》或《模組》，按滑鼠右鍵，選取跳出的功能表之【插入／模組】，一樣可以達成目的。

四、程式碼視窗

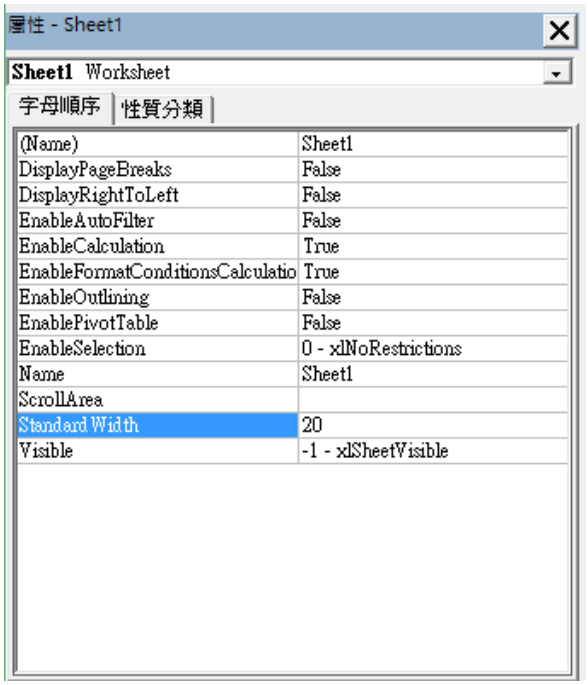


程式碼視窗中可放置活頁簿中各相關成員自己的程式碼。利用錄製方式產生的程式碼會放在模組中。如上圖視窗上的標題顯示 Book1.xlsm – Module1(程式碼)。至於我們自行撰寫的程式碼，則可以按需要放在各個成員中。不同版本的 Excel 可能產生不同的程式碼，但是執行的效果是一樣的。

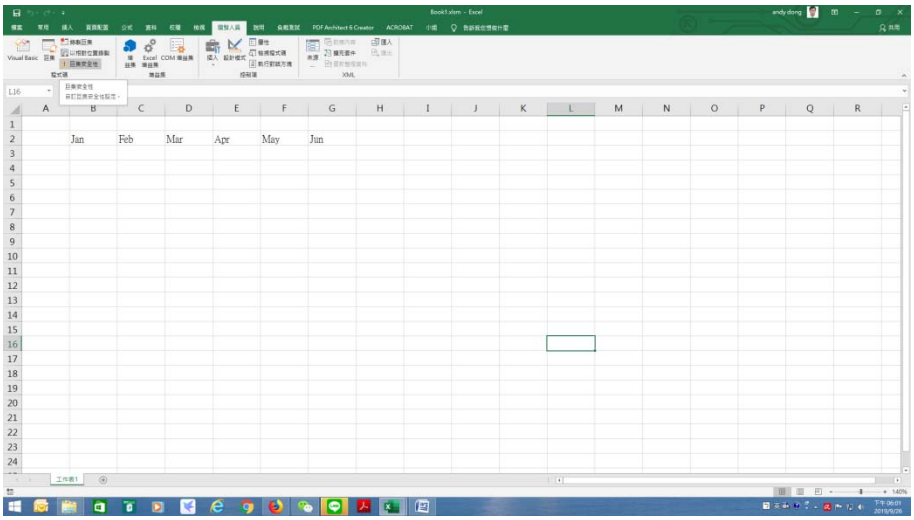
五、屬性視窗

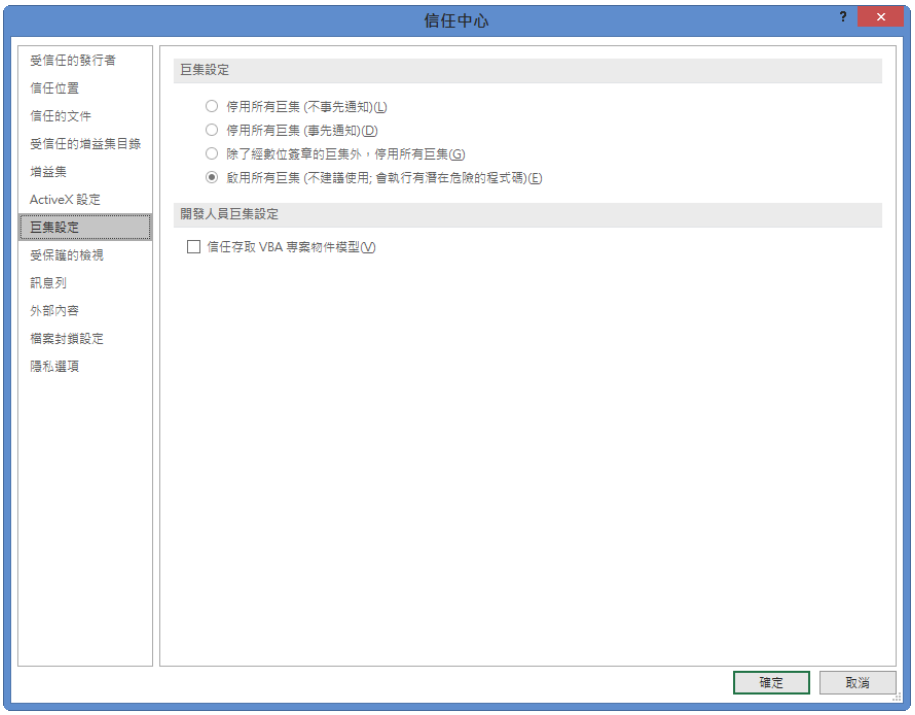
至於屬性視窗則紀錄各個活頁簿成員的一些特徵，以上圖為例，由於《Module1》成員並不是一個可見的成員，因此它的特徵只有一個，就是它的名字。讀者可以試著改變其名字，看看專案總管中的名稱是否有相對的改變。

如果我們在專案總管中點選《Sheet1(Sheet1)》，則屬性視窗中便會顯示 Sheet1 的屬性，如下圖所示。讀者可試著改變其中的一些特徵，如 Standard Width，將 8.38 改為 20，然後切換到試算表的 Sheet1 去看看，儲存格的欄寬是否全都變寬了。



錄製完成後，不要忘記將檔案儲存起來，我們可以將之存為 Book1.xlsm，讀者可在所附的光碟中的目錄 Ch01 下找到此檔案。當讀者要打開此檔時，可能會有警示訊息出現而無法開啟，這是因為 Excel 的安全設定所致。讀者可以選取【工具／巨集／安全性…】，便會出現下面對話視窗。





選取較低等級的安全性，按【確定】鈕後關閉 Excel 再開啟之，此時便可選擇開啟巨集，看到錄製的程式碼了。