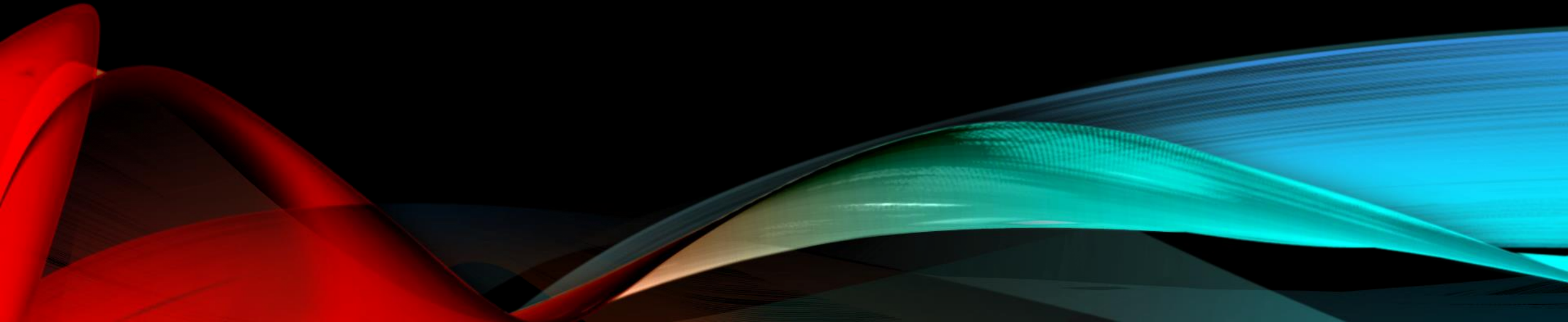


EXCEL VBA 進階班

Lecture 3

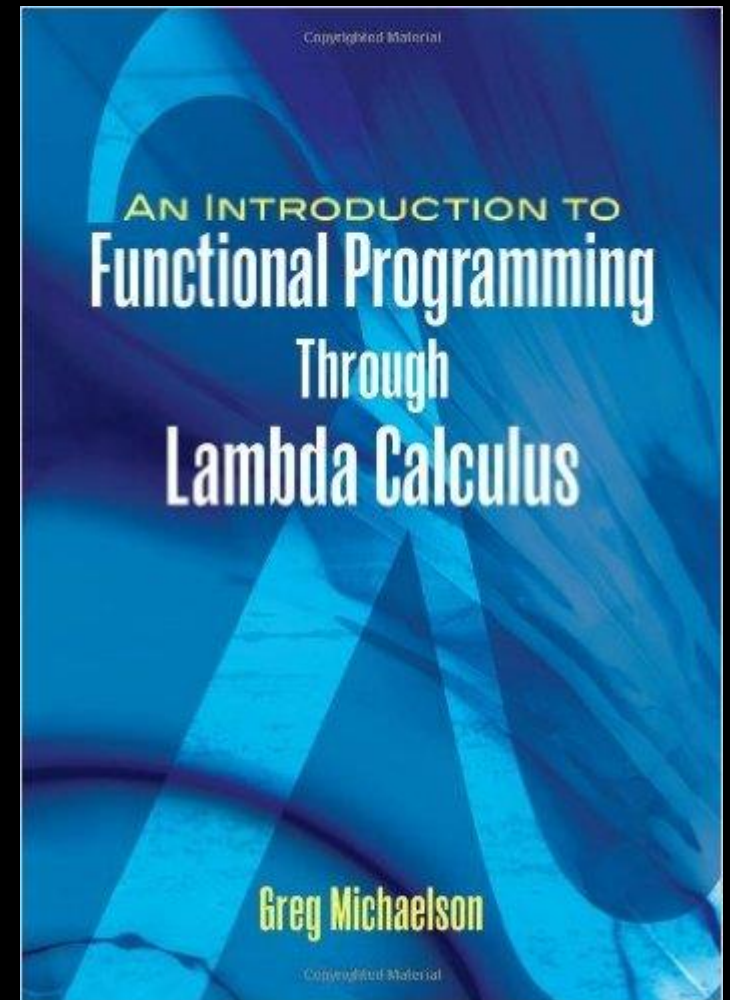
SECTION 4. FUNCTIONAL PROGRAMMING (2)

Evaluate



FUNCTIONAL PROGRAMMING: EXPRESSION

- 函數式語言另一特性為表達式 (Expression) 。
- 表達式代表的是函數一連串運算的過程。
- 我們可以 **在需要的時候再去執行表達式中的運算過程** 。
- 有興趣的人可以參考 λ - Calculus 。



EVALUATE

- R 語言在這部分功能就很強，甚至能用字串即時組出一段之後需要的程式碼並執行。
- VBA 的 Evaluate 也可以做到類似的功能，但僅限於 Excel 公式。

EVALUATE

- 例如，我們想要讓兩串儲存格，X 與 Y 的每個元素比較大小留下大的，但必須在我們選定了 X 與 Y 儲存格的範圍後再開始執行並回傳值。
- 以前 VBA 中我們會使用以下函數：

```
Function pmax(x As Range, y As Range) As Double()
```

```
Dim nRow As Integer: nRow = x.Rows.Count
```

```
Dim nCol As Integer: nCol = x.Columns.Count
```

```
Dim xValue As Variant: xValue = x.Value
```

```
Dim yValue As Variant: yValue = y.Value
```

```
Dim i As Integer
```

```
Dim j As Integer
```

```
ReDim output(nRow, nCol) As Double
```

```
For i = 1 To nRow
```

```
    For j = 1 To nCol
```

```
        If xValue(i, j) >= yValue(i, j) Then output(i, j) = xValue(i, j) Else output(i, j) = yValue(i, j)
```

```
    Next j
```




```
Next i
```

```
pmax = output
```

```
End Function
```


EVALUATE

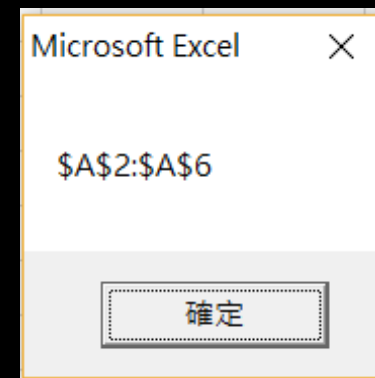
- 但其實在 Excel 上我們可以很簡單的使用 IF 函數達到該效果：

D2		:	  		{=IF(A2:A6>=B2:B6,A2:A6,B2:B6)}			
	A	B	C	D	E	F	G	H
1	X	Y	pmax VBA	pmax Excel				
2	1	2	2	2				
3	4	1	4	4				
4	2	5	5	5				
5	3	3	3	3				
6	8	7	8	8				

EVALUATE

- 那我們能不能在 VBA 中使用 Excel 的寫法呢？Evaluate 提供一個方法，能將一串字串變成一個 Excel 公式的表達式：
- 首先，X 與 Y 兩部分儲存格的範圍我們可以用 .Address 取得絕對位置。

```
Function pmaxUsingEvaluate(x As Range, y As Range) As Variant  
  
    Dim xLocation As String: xLocation = x.Address  
    MsgBox xLocation  
  
End Function
```



EVALUATE

- 接著組合成我們想要的字串內容，注意，不用在最左邊加入等號：

```
Function pmaxUsingEvaluate(x As Range, y As Range) As Variant
```

```
Dim xLocation As String: xLocation = x.Address
```

```
Dim yLocation As String: yLocation = y.Address
```

```
Dim expression As String
```

```
' 組合出想要的字串 : IF(X>=Y,X,Y)
```

```
expression = "IF(" + xLocation + ">=" + yLocation + "," + xLocation + "," + yLocation + ")"
```

```
End Function
```

EVALUATE

- 最後利用 Application.Evaluate 執行該公式取得結果即可：

```
Function pmaxUsingEvaluate(x As Range, y As Range) As Variant
```

```
Dim xLocation As String: xLocation = x.Address
```

```
Dim yLocation As String: yLocation = y.Address
```

```
Dim expression As String
```

```
' 組合出想要的字串：IF(X>=Y,X,Y)
```

```
expression = "IF(" + xLocation + ">=" + yLocation + "," + xLocation + "," + yLocation + ")"
```

```
pmaxUsingEvaluate = Application.Evaluate(expression)
```

```
End Function
```

EVALUATE

- 可以看出三者都會得到相同的結果：

	A	B	C	D	E
1	X	Y	pmax VBA	pmax Excel	pmax Evaluate
2	1	2	2	2	2
3	4	1	4	4	4
4	2	5	5	5	5
5	3	3	3	3	3
6	8	7	8	8	8
7					

EVALUATE

- 使用 Evaluate 可以很直覺的在 VBA 中使用 Excel 函數，如同在 Excel 中輸入一樣。
- Evaluate 還有很多功用，可以幫助我們在 VBA 中快速產生想要的東西，以下舉幾個例子。

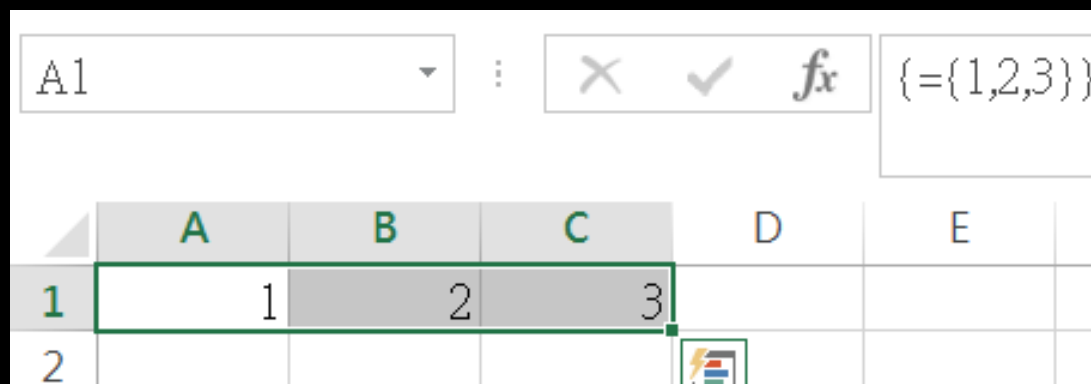
EVALUATE

- 1. 立刻創造出給定數字的二維陣列。
- 以前在一維陣列的部分，我們可以用 `Array` 函數快速創造，但無法變為二維：

```
Sub createArray()  
  
On Error GoTo isOneDim  
Dim x As Variant: x = Array(1, 3, 5, 7, 9) '二維陣列怎麼辦？  
  
MsgBox "第一維長度: " & UBound(x, 1)  
MsgBox "第二維長度: " & UBound(x, 2)  
Exit Sub  
  
isOneDim:  
MsgBox "非二維陣列"  
  
End Sub
```

EVALUATE

- 在 Excel 中可以利用花括號，{，快速創造陣列，例如創造一個一維陣列 {1, 2, 3} 放在 A1:A3，可以這樣做：



EVALUATE

- 而二維陣列可以使用分號，；，代表切換至下個 Row，注意每個 Row 的元素必須相同數量，否則 Excel 會提示錯誤。

A1

:

✕

✓

fx

{={1,2,3;4,5,6}}

	A	B	C	D	E	F
1	1	2	3			
2	4	5	6			
3						

EVALUATE

- 依照此方法使用 Evaluate，即可快速將一個二維陣列輸入至 Variant 中，注意此方法還是限 Variant。

```
Sub createArray()  
  
On Error GoTo isOneDim  
Dim x As Variant: x = Application.Evaluate("{1,2,3;4,5,6}")  
|  
MsgBox "第一維長度: " & UBound(x, 1)  
MsgBox "第二維長度: " & UBound(x, 2)  
Exit Sub  
  
isOneDim:  
MsgBox "非二維陣列"  
  
End Sub
```

EVALUATE

- 2. 把 Column A 的每個數字乘上相對應的 Row 之後放到 Column B

	A	B
1	把下面每個數字乘上其 Row 的數字並放到 Column B	結果
2	1	
3	4	
4	6	
5	1	
6	3	
7	65	
8	2	
9	4	
10	5	
11	41	

EVALUATE

- 以前在 VBA 我們會想到使用 For 迴圈：

```
Sub multipleExample()  
  
Dim endRow As Integer: endRow = Cells(1, 1).End(xlDown).Row  
Dim i As Integer  
  
For i = 2 To endRow  
    Cells(i, "B").Value = Cells(i, "A").Value * i  
Next i  
  
End Sub
```

EVALUATE

- 現在對於儲存格我們可以應用原來的 Excel 陣列計算了。

```
Sub multipleExample()  
  
Dim endRow As Integer: endRow = Cells(1, 1).End(xlDown).Row  
Dim inputRange As Range: Set inputRange = Range("A2", Cells(endRow, "A"))  
inputRange.Offset(, 1).Value = Application.Evaluate(inputRange.Address + " * Row(" + inputRange.Address + ")")  
|  
End Sub
```

EVALUATE：練習

- 下面有一組資料，有五萬筆商品的抽樣調查，GOOD 代表狀況好，BAD 代表不好，試著用 Evalute 寫一個函數或子程序計算良率。

抽樣編號	物品狀態
1	GOOD
2	BAD
3	BAD
4	BAD
5	BAD
6	BAD
7	BAD
8	GOOD

EVALUATE：另一種簡寫

- 除了用 `Application.Evaluate` 以外，還有另外一種更簡單的簡寫，`[]`。
- 例如我們要計算 A1:A5 的值之總合，可以使用：
 - `Application.Evaluate("SUM(A1:A5)")`
 - `[SUM(A1:A5)]`

EVALUATE：另一種簡寫

- 兩種方式都會得到相同的結果，但要注意，[] 裡並不是放字串，而是一段 Excel 公式，不用加 "" 。
- 但反過來說也成為 [] 的缺點，[] 並沒有辦法根據情況即時用字串組成想要的 Excel 公式字串，而是只能執行固定的內容。
- [] 最主要的用途可以用來代替原來之 Range，使程式碼變得更精簡。

SECTION 5. 進階變數型態介紹 (2)

Variant 詳述



VARIANT 變數型態詳述

- Variant 是一種不同於其他變數型態的獨特變數型態。
- 一般的變數型態如 Double、String、Boolean 等稱為強型別 (Strong Type)，賦值只能夠是相同型態的資料 (或是會被強制轉為符合的型態)。
- 例如：
 - 在 Double 型態的變數中輸入字串會因型態不同產生錯誤。
 - 在 Integer 型態的變數中輸入 Double 會去掉小數點被強制轉為 Double。

VARIANT 變數型態詳述

- 但 Variant 為弱型別，自動根據賦值的類型定義變數類型。
- 如此方便的變數型態，到底犧牲了什麼呢？
 - 第一，Variant 每次輸出輸入時需要花時間判斷資料型態，因此相對來說時間較慢，據 Lomax 書中所說，大約比強型別變數多了 30 % 的時間。
 - 第二，Variant 由於不會因型態錯誤而停止，因此程式可讀性會被降低，較容易出現錯誤。

VARIANT 變數型態詳述

- 我們利用一個例子來說明上述的缺點。
- 某一商品抽五萬筆資料來檢測良率：
 - 商品為良品顯示 “GOOD”
 - 不良品顯示 “BAD”

抽樣編號	物品狀態
1	GOOD
2	BAD
3	BAD
4	BAD
5	BAD
6	BAD
7	BAD
8	GOOD

VARIANT 變數型態詳述

- 用 Variant 與強行別變數皆可建立一個良率的函數，強行別變數的函數如下：

```
Private Function calculateGoodRatioViaString(samples() As String) As Double

    Dim nSample As Long: nSample = UBound(samples)
    Dim i As Long

    For i = 1 To nSample
        Select Case samples(i)
            Case "GOOD"
                calculateGoodRatioViaString = calculateGoodRatioViaString + 1
        End Select
    Next i

    calculateGoodRatioViaString = calculateGoodRatioViaString / nSample

End Function
```

VARIANT 變數型態詳述

- Variant 型態函數如下：

```
Private Function calculateGoodRatioViaVariant(samples)

Dim nSample: nSample = UBound(samples)
Dim i

For i = 1 To nSample
    Select Case samples(i)
        Case "GOOD"
            calculateGoodRatioViaVariant = calculateGoodRatioViaVariant + 1
    End Select
Next i

calculateGoodRatioViaVariant = calculateGoodRatioViaVariant / nSample

End Function
```

VARIANT 變數型態詳述

- 可以明顯看出，Variant 函數的可讀性明顯較低，各個變數的作用無法透過型態一目了然。
- 另外可以來比較兩者的速度。

VARIANT 變數型態詳述

- 關於計算時間，可以使用 VBA 內建的一個函數，Timer。
- Timer 會以秒為單位回傳目前系統之時間。
- 要計算一段程式的消耗時間可以用以下的方法：

時間計算：TIMER

'計算產生十萬個亂數至陣列的時間

```
Sub calculateSpendingTime()
```

```
Dim startTime As Double, _  
    i As Long, _  
    x(100000#) As Double
```

'開始時間

```
startTime = Timer
```

```
For i = 1 To 100000#
```

```
    x(i) = Rnd
```

```
Next i
```

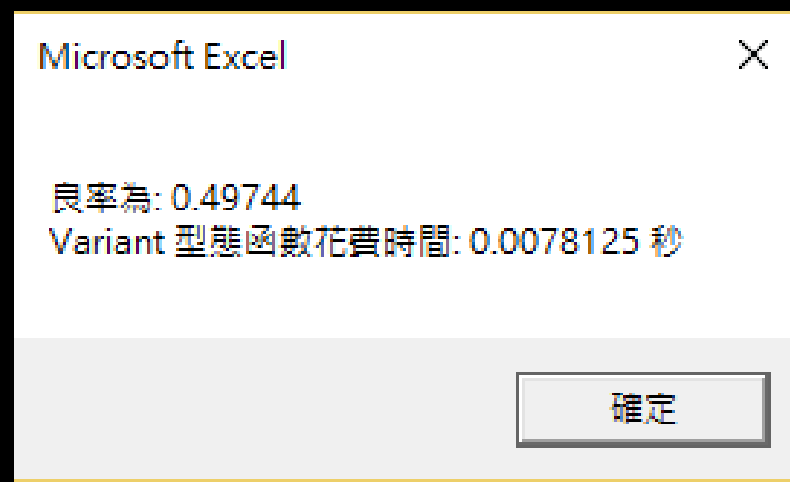
結束時間 - 開始時間 = 消耗的時間

```
MsgBox "花費 " & (Timer - startTime) & " 秒"
```

```
End Sub
```

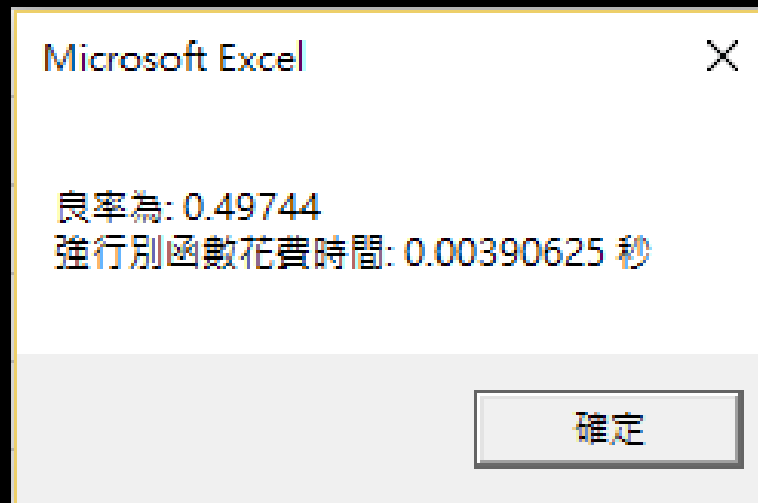
VARIANT 變數型態詳述

- Variant 型態五萬筆資料花費時間：



VARIANT 變數型態詳述

- 強行別五萬筆資料花費時間：



VARIANT 變數型態詳述

- 單筆有的時候結果會有落差，甚至出現強行別較慢。
- 不過如果是平均的情況呢？
- 大家可以練習計算平均花費時間。

VARIANT 變數型態詳述

- Variant 變數與其他變數另外不同之處，在於它提供了一些其他變數所沒有的特別狀態。
- 可以使用 VarType 函數觀察該 Variant 變數目前的變數型態。

```
Sub getVariableType()
```

```
Dim x As Variant
```

```
'輸入 x = 100
```

```
x = 100
```

```
MsgBox "x: " & x & Chr(10) & "Type of x: " & VarType(x)
```

```
'輸入 x = 500000
```

```
x = 500000
```

```
MsgBox "x: " & x & Chr(10) & "Type of x: " & VarType(x)
```

```
'輸入 x = 1E+5
```

```
x = 100000#
```

```
MsgBox "x: " & x & Chr(10) & "Type of x: " & VarType(x)
```

```
'輸入 x = 2.71828
```

```
x = 2.71828
```

```
MsgBox "x: " & x & Chr(10) & "Type of x: " & VarType(x)
```

```
End Sub
```

VARIANT 變數型態詳述

- 而各個數字代表甚麼意思呢？
- Type 2 代表的是 Integer，而當數字超過 Integer 的邊界（-32,768 ~ +32,768）時，整數會自動變為長整數 Long，也就是 Type 3。
- 而我們可以看到浮點數形態時都是 Type 5，也就是 Double，值得注意的是利用科學計算符號得到的數字皆為 Double，Variant 預設浮點數皆為 Double，除非利用轉換型別的函數。

VARIANT 變數型態詳述： 型別轉換函數

- VBA 裡提供一系列型別轉換函數，可以把某一型別的資料嘗試轉換為另一型別。
- 注意：其中有一型別為 Variant 之 subtype，一般宣告變數時並不存在該種型別。
 - CBool：轉為布林值
 - CByte：轉為 Byte
 - CDate：轉為日期
 - CCur：轉為 Currency
 - CDbI：轉為 Double

VARIANT 變數型態詳述： 型別轉換函數

- CInt：轉為整數
- CLng：轉為長整數
- CSng：轉為 Single
- CStr：轉為 String
- CVar：轉為 Variant
- CDec：轉為 Decimal，該種資料型別為 Variant 裡的一種 Type，無法直接宣告只能利用 CDec 函數輸入至 Variant 中。

VARIANT 變數型態詳述

- Decimal 為整數的一種，用來處理超過 Long 之大小的整數。
- Long 的範圍為 -2,147,483,648 ~ -2,147,483,647，Variant 在處理時，超過該數字則使用 Double 取代。
- 如想保留整數部位非浮點數，可以使用 Decimal，其範圍為 +/- 79,228,162,514,264,337,593,543,950,335。
- 整體對照表如下：

VARIANT 變數型態詳述

Value	Subtype
0	Empty
1	Null
2	Integer
3	Long
4	Single
5	Double
6	Currency
7	Date
8	String

VARIANT 變數型態詳述

Value	Subtype
9	OLE Automation Object (VBA 中的物件，例如 Worksheet 等)
10	Error
11	Boolean
12	Variant 陣列
13	Data Access Object (與資料庫有關的物件)
14	Decimal
17	Byte
36	User-defined type
8192	陣列

VARIANT 變數型態詳述：EMPTY

- 部分資料型態之後會再詳述，此處介紹 Null 與 Empty 兩種情況。
- Empty 也是 Variant 特有的 subtype，代表的是還沒被填入任何的值。
- 空白的儲存格之 Value 也是 Empty。
- 可以利用 IsEmpty 函數檢查某個儲存格或 Variant 變數是否為 Empty。

VARIANT 變數型態詳述：NULL

- Null 幾乎存在於每種程式語言中，在許多定義裡，Null 意指 “沒有值” 或是 “未知的值”。
- 在 VBA 中存取資料庫（如 SQL 等）出現缺失值時，有可能會回傳 Null。
- Null 有以下幾種特性需要注意：

VARIANT 變數型態詳述：NULL

- 1. Null 並不等於任何值，甚至是另外一個 Null。
- 2. 任何東西與 Null 做運算皆會變為 Null，我們可以用 IsNull 函數做確認：

```
Sub nullExample()  
  
Dim x As Variant: x = Null  
Dim y As Variant: y = 1  
Dim z As Variant  
z = x > y  
y = x + y  
  
MsgBox IsNull(y)  
MsgBox IsNull(z)  
End Sub
```

VARIANT 變數型態詳述：NULL

- 對 Null 進行錯誤處理可以使用 IsNull 函數，當發現 Null 時可以做例外處理。
- 比較新的版本有 Nz 與 IIF 函數，可以更方便進行錯誤處理：
- <https://support.office.com/zh-tw/article/Nz-%E5%87%BD%E6%95%B8-8ef85549-cc9c-438b-860a-7fd9f4c69b6c>