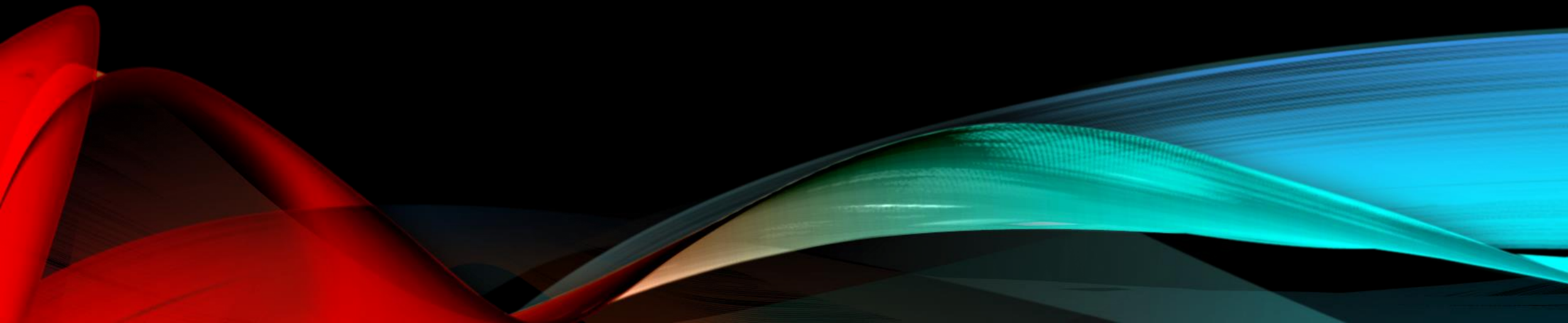


EXCEL VBA 進階班

SECTION 0. 課程簡介

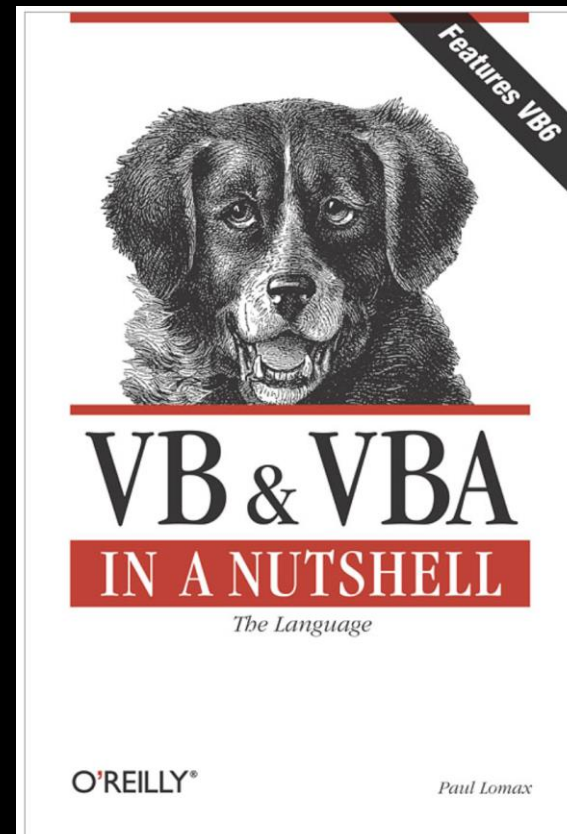
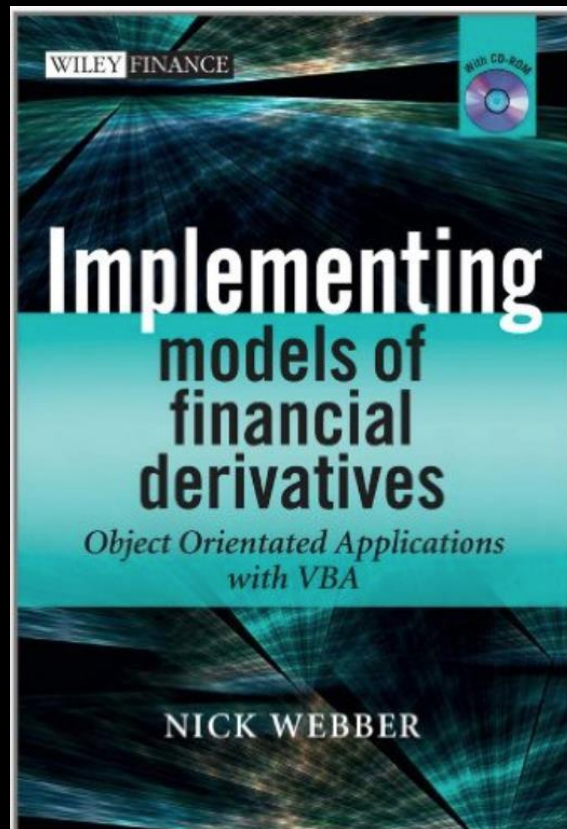


課程簡介

- 本課程為進階班，希望同學對基礎語法有一定程度的瞭解。
- 課程內容核心如下：
 - 進階VBA語法與效率改善
 - 物件導向程式設計
 - 利率衍生性金融商品評價與避險

課程簡介

- 課程內容無指定教課書，主要可以參考以下書籍：



上課資訊

- 請加入以下社團：
- facebook.com/groups/advancedVBV275

課程簡介

- 在第四堂課開始以前，如發現課程內容與預期有差距，可以按比例退費。
- 如需獲得課程證書，須滿足以下兩個條件：
 - 課堂會點名，點名低於五次者不發放證書。
 - 課程結束後一星期內須繳交作業。

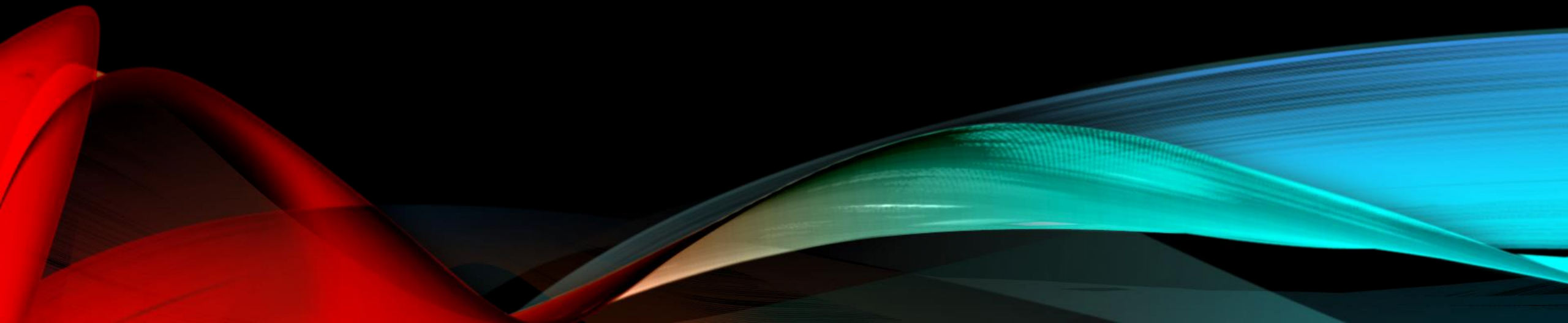
課程簡介：作業

- 作業內容：以物件導向方式寫出一個投資策略回測獲釋衍生性金融商品評價的程式
- 做不出來也會斟酌，但是請勿教空白檔案。

課程簡介：錄影

- 本課為方便同學複習程有進行錄影。
- 為維護上課同學權益，請勿將錄影散播。

SECTION 1. 變數進階修飾

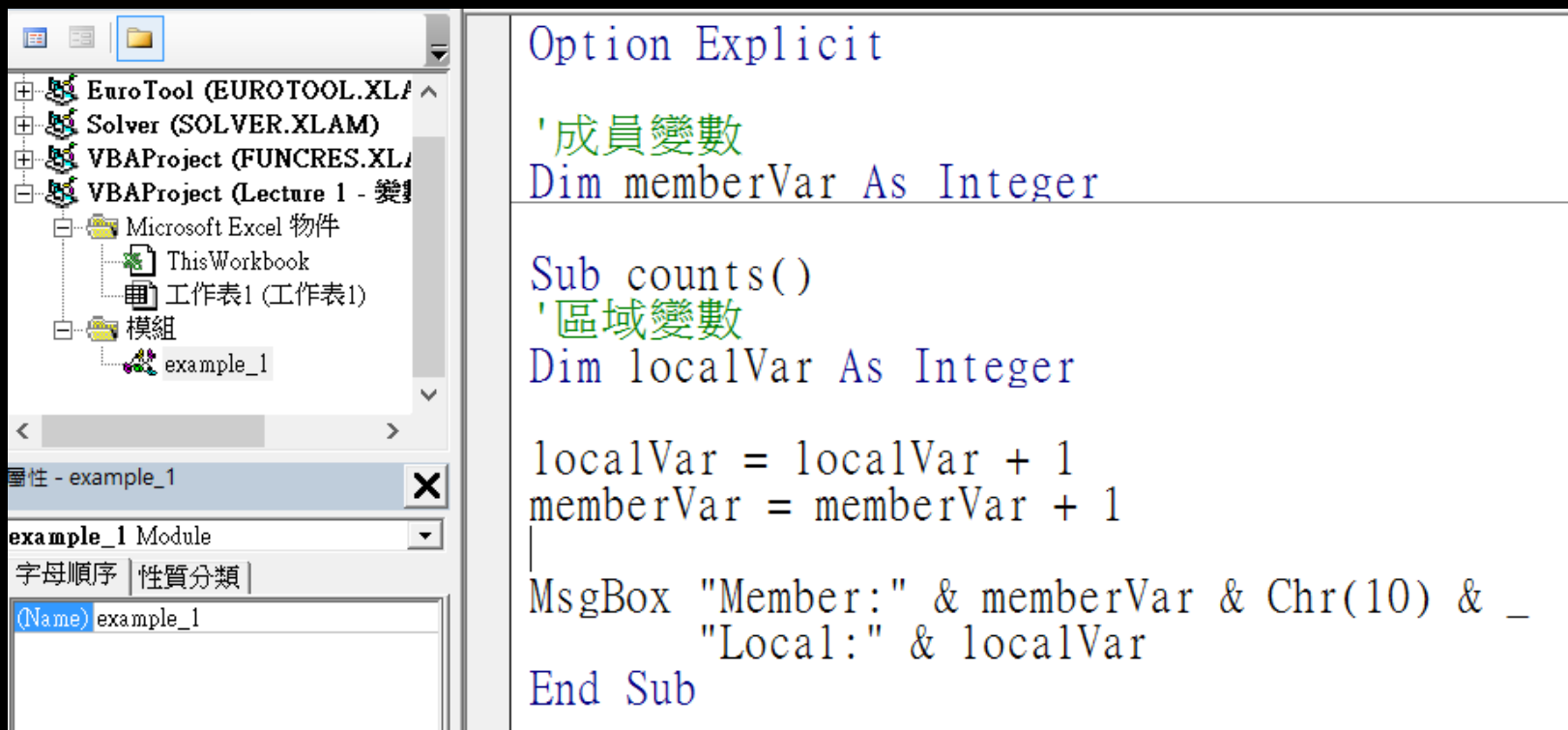


成員變數複習

- 初階班的時候有提過，變數影響範圍分為以下三種：
 - 全域變數 (Global Variable)
 - 成員變數 (Member Variable)
 - 區域變數 (Local Variable)
- 其中，成員變數可以讓該模組 (Module) 中的任何子程序與函數存取，不會在該子程序 (函數) 結束時消失，而是在該模組或該變數被改名或刪除時才會釋放記憶體。

成員變數複習

- 參考變數影響範圍範例的 example_1:



The screenshot displays the VBA Project Explorer on the left and the VBA Editor on the right. The Project Explorer shows a project named 'VBAProject (Lecture 1 - 變數)' containing a '模組' (Modules) folder with 'example_1' selected. The VBA Editor shows the code for 'example_1 Module'.

```
Option Explicit

'成員變數
Dim memberVar As Integer

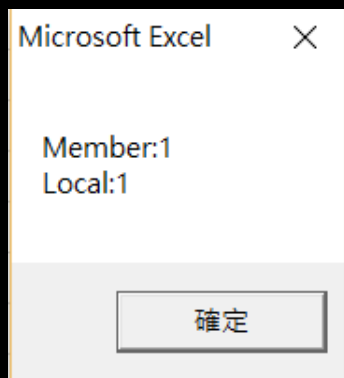
Sub counts()
    '區域變數
    Dim localVar As Integer

    localVar = localVar + 1
    memberVar = memberVar + 1

    MsgBox "Member:" & memberVar & Chr(10) & _
        "Local:" & localVar
End Sub
```

成員變數複習

- 第一次執行得到：

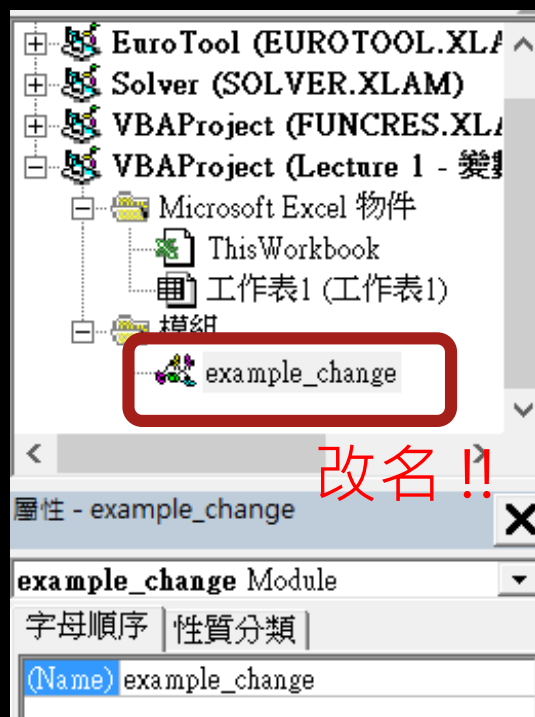


- 第二次得到：



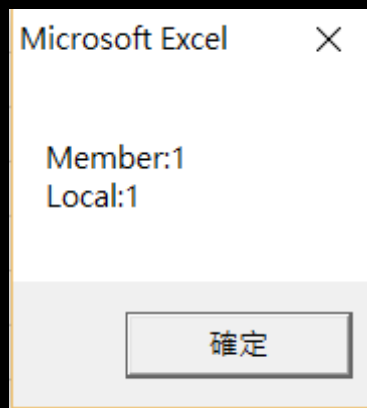
成員變數複習

- 可以看出成員變數有被記錄下來，但當我們將 example_1 改名為 example_change 後：



成員變數複習

- 由於原來的 example_1 模組消失了，因此 memberVar 的紀錄跟著消失被重置，在次執行 count，得到：



成員變數複習

- 成員變數因為只要模組存在就不會由記憶體中被釋放，因此可以記錄一些會被重複使用的變數。
- 回顧指數加權平均 (EMA)，我們需要一個 $\alpha \in (0,1)$ 來當作各天權重的基礎。

成員變數複習

- 以前的做法是將 α 也當作一個引數來進行計算

'Version 1

Function emaCalculator(priceArray As Range, alpha As Variant) As Variant

Dim dominator As Variant

'加權平均加總所除上的分母

Dim onePrice As Range

'某天價格的儲存格

Dim weight As Variant

'該價格之權重

Dim counts As Integer: counts = priceArray.Count

'現在權重的 alpha 之次方

For Each onePrice In priceArray

counts = counts - 1

weight = alpha ^ counts

emaCalculator = emaCalculator + onePrice.Value * weight

dominator = dominator + weight

Next onePrice

emaCalculator = emaCalculator / dominator

End Function

成員變數複習

- 但是我只要一不小心忘記在第二個引數的部份加上 \$ ，往下拉或點兩下就會造成計算錯誤：

SUM						=emaCalculator(B7:B11,F6)	
	A	B	C	D	E	F	
1	Date	Adjusted Close	EMA		Alpha	0.97	
2	2000/1/3	1455.22					
3	2000/1/4	1399.42					
4	2000/1/5	1402.11					
5	2000/1/6	1403.45					
6	2000/1/7	1441.47	1420.208				
7	2000/1/10	1457.6	1457.6				
8	2000/1/11	1438.56	1438.56				
9	2000/1/12	1432.25	1432.25				
10	2000/1/13	1449.68	1449.68				
11	2000/1/14	1465.15	or(B7:B11,F				
12	2000/1/18	1455.14					

成員變數複習

- 第一個想到的辦法是利用成員變數，我們把 α 移到函數外變為成員變數，如果要改變 α ，我們新增一個子程序 `setAlpha` 來賦予新的值。

成員變數複習

```
Dim alpha As Variant
```

```
Sub setAlpha()
```

```
alpha = Range("F1").Value
```

```
End Sub
```

'Version 2 : Member variable

```
Function emaCalculator(priceArray As Range) As Variant
```

```
Dim dominator As Variant
```

'加權平均加總所除上的分母

```
Dim onePrice As Range
```

'某天價格的儲存格

```
Dim weight As Variant
```

'該價格之權重

```
Dim counts As Integer: counts = priceArray.Count
```

'現在權重的 alpha 次方

```
For Each onePrice In priceArray
```

```
counts = counts - 1
```

```
weight = alpha ^ counts
```

```
emaCalculator = emaCalculator + onePrice.Value * weight
```

```
dominator = dominator + weight
```

```
Next onePrice
```

```
emaCalculator = emaCalculator / dominator
```

```
End Function
```

成員變數複習

- 執行 setAlpha 之後再計算 EMA 可以得到正確的值：

=emaCalculator(B2:B6)						
	A	B	C	D	E	F
1	Date	Adjusted Close	EMA		Alpha	0.97
2	2000/1/3	1455.22				
3	2000/1/4	1399.42				
4	2000/1/5	1402.11				
5	2000/1/6	1403.45				
6	2000/1/7	1441.47	1420.208			
7	2000/1/10	1457.6				
8	2000/1/11	1438.56				
9	2000/1/12	1432.25				
10	2000/1/13	1449.68				
11	2000/1/14	1465.15				

成員變數複習：練習

- 但是如果使用者忘記在開始計算前使用 `setAlpha` 呢？要如何改善這個問題？
- 使用者如何確認當下使用的 α 為多少？
- 試著練習改進該程式。

成員變數複習：缺點

- 成員變數有一個小缺點，當函數或子程序裡宣告一個同名的區域變數，等於是把該成員變數重新宣告一次。

成員變數複習：缺點

```
Option Explicit
```

```
'成員變數
```

```
Dim memberVar As Integer
```

```
Sub counts()
```

```
'區域變數
```

```
Dim localVar As Integer
```

```
localVar = localVar + 1
```

```
memberVar = memberVar + 1
```

```
MsgBox "Member:" & memberVar & Chr(10) & _
```

```
"Local:" & localVar
```

```
End Sub 執行 oopsCounts 會重新計算
```

```
' Oh !! 不小心同名了
```

```
Sub oopsCounts()
```

```
Dim memberVar As Integer
```

```
memberVar = memberVar + 1
```

```
MsgBox "Member:" & memberVar
```

```
End Sub
```

靜態變數

- 那除了成員變數外，有沒有其他方法可以達到同樣的效果呢？
- 如果有一個區域變數，它存在的時間能跟成員變數一樣久呢？
- 在這裡我們可以使用靜態變數 (static variable)。
- 通常程序停止後，區域變數就不復存在，而靜態變數會繼續存在並保留其最新值。
- 下次當程式碼呼叫程序時，不會重新初始化變數，而且它仍會保存指派給它的最新值。

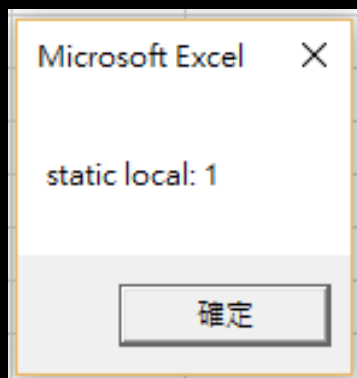
靜態變數

- 以前面 counts 的範例來說，我們嘗試以下範例：

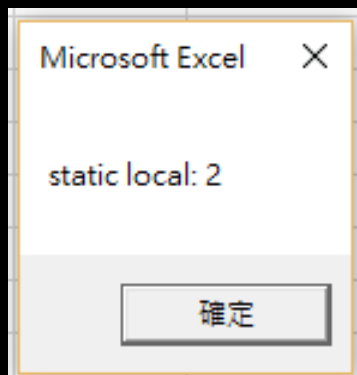
```
Sub newCounts()  
'宣告為靜態變數前面用 static  
Static localVar As Integer  
|  
localVar = localVar + 1  
MsgBox "static local: " & localVar  
  
End Sub
```

靜態變數

- 第一次執行：



- 第二次執行：



靜態變數

- 被宣告為 Static 的變數，只要該子程序（函數）不被更名，或是該變數一開始宣告的修飾與型態沒被更改，則該值將被繼續存放在記憶體中。

靜態變數

- 回到 EMA 計算的範例，如果不使用成員變數，要如何利用靜態變數給定 α 呢？
- 可以加入兩個靜態變數：
 - `isInitialized` : 確定有沒有初始化
 - `alpha` : 計算用的 α

```
Sub setAlpha(ByRef alpha As Variant)
alpha = Range("F1").Value
End Sub
```

'Version 3 : Static variable

```
Function emaCalculator(priceArray As Range) As Variant
```

```
Static alpha As Variant
```

' alpha 的值

```
Static isInitialized As Boolean
```

' 檢查看 alpha 有沒有被設定了

'設定初始化

```
If Not isInitialized Then: Call setAlpha(alpha)
```

'開始原本的計算

```
Dim dominator As Variant
```

'加權平均加總所除上的分母

```
Dim onePrice As Range
```

'某天價格的儲存格

```
Dim weight As Variant
```

'該價格之權重

```
Dim counts As Integer: counts = priceArray.Count
```

'現在權重的 alpha 次方

```
For Each onePrice In priceArray
```

```
counts = counts - 1
```

```
weight = alpha ^ counts
```

```
emaCalculator = emaCalculator + onePrice.Value * weight
```

```
dominator = dominator + weight
```

```
Next onePrice
```

```
emaCalculator = emaCalculator / dominator
```

```
End Function
```


靜態變數

- 由於布林值宣告後預設為 `False`，因此該函數一開始是預設沒有被初始化的。
- 因此當判斷 `Not isInitialized` 為 `True` 時，會執行 `setAlpha` 將 `F1` 儲存格的值給該函數。
- 但該函數有一個缺點，一旦經過初始化後， α 將不能再更動，要如何修改將函數更有彈性呢？

PARAMARRAY

- 回憶一個我們很常用的 Excel 函數：SUM。
- SUM 有一個很有趣的特性，如下圖所示，假如我們要加總 B1 到 B3 的值：

	A	B
1	Data 1	2
2	Data 2	3
3	Data 3	8
4	SUM	

計算加總到此處

PARAMARRAY




- 我們可以有以下的做法：
- 第一種：取聯集

B4	:	✕	✓	<i>fx</i>	=SUM(B1,B2,B3)
	A	B	C	D	
1	Data 1	2			
2	Data 2	3			
3	Data 3	8			
4	SUM	13			

=SUM(B1,B2,B3)




PARAMARRAY

- 第二種：利用範圍運算子

B4	:				=SUM(B1:B3)	=SUM(B1:B3)
	A	B	C	D		
1	Data 1	2				
2	Data 2	3				
3	Data 3	8				
4	SUM	13				

PARAMARRAY

- 第三種：甚至都使用

B4	:				=SUM(B1:B2,B3)
	A	B	C	D	
1	Data 1	2			
2	Data 2	3			
3	Data 3	8			
4	SUM	13			

=SUM(B1:B2,B3)




PARAMARRAY

- 但是依照我們以前對函數的了解，函數給定的引數數量是固定的，如我們自己寫一個加總函數如下：

```
Function mySum(inputData As Variant) As Double  
    Dim oneVal As Variant  
    mySum = 0  
    For Each oneVal In inputData  
        mySum = mySum + oneVal  
    Next oneVal  
End Function
```





PARAMARRAY

- 該函數可以利用範圍運算子得出我們想要的結果：

B4	:				=mySum(B1:B3)
	A	B	C	D	
1	Data 1	2			
2	Data 2	3			
3	Data 3	8			
4	SUM	13			

PARAMARRAY

- 但從程式碼可以看出，這個自訂函數只有一個引數，inputData !!
- 我們嘗試用聯集會得到以下結果：

B4		:				=mySum(B1,B2,B3)
	A	B	C	D		
1	Data 1	2				
2	Data 2	3				
3	Data 3	8				
4	SUM 	#VALUE!				

B1,B2,B3 有了三個引數

PARAMARRAY

- 那為何 Excel 的 SUM 可以辦得到呢？原因是 SUM 函數把它裡面所有的 input 都視作一個引數。
- 相同的我們可以利用 ParamArray 辦到。

PARAMARRAY

- 試試看用以下方法

```
'ParamArray Version
Function mySum_2(ParamArray inputData()) As Variant

Dim oneElem As Variant

mySum_2 = 0

For Each oneElem In inputData
    mySum_2 = mySum_2 + oneElem
Next oneElem

End Function
```

PARAMARRAY

- 這時候試試看使用聯集的方式，該函數又順利執行了。

B4				
	A	B	C	D
1	Data 1	2		
2	Data 2	3		
3	Data 3	8		
4	SUM	13		

PARAMARRAY

- 但是想換回範圍運算子或是並行使用會發生甚麼事呢？
- 很遺憾又不能使用了，ParamArray 到底是一個怎麼樣的修飾詞呢？

B4				
	A	B	C	D
1	Data 1	2		
2	Data 2	3		
3	Data 3	8		
4	SUM	#VALUE!		

B1:B2,B3 失敗了，為什麼？

PARAMARRAY

- ParamArray 為不定量引數，可以將所有各種不同的資料型態變為一個陣列。
- ParamArray 允許為 **empty** 也允許元素為陣列。
- 回到剛才的 mySum_2，B1:B2 為一個 Variant 之陣列，因此 **mySum_2** 是無法與一個陣列相加的。
- 那要如何克服該問題呢？

PARAMARRAY

- 當元素不為陣列時，該元素的值可以直接加入加總之中，因此，我們如果確定該元素不為陣列，可直接進行累加。

```
'ParamArray Version
Function mySum_2(ParamArray inputData()) As Variant
    Dim oneElem As Variant
    mySum_2 = 0
    For Each oneElem In inputData
        '不是陣列就直接累加
        If Not IsArray(oneElem) Then
            mySum_2 = mySum_2 + oneElem
        End If
    Next oneElem
End Function
```

PARAMARRAY

- 那面對是陣列的元素，我們可以把裡面每個值取出來進行加總，因此可以再用一層的 For Each。

PARAMARRAY

```
'ParamArray Version
Function mySum_2(ParamArray inputData()) As Variant

Dim oneElem As Variant, _
    elemInElem As Variant

mySum_2 = 0

For Each oneElem In inputData

    '不是陣列就直接累加
    If Not IsArray(oneElem) Then
        mySum_2 = mySum_2 + oneElem

    '是陣列就再用一層For Each 取值累加
    Else
        For Each elemInElem In oneElem
            mySum_2 = mySum_2 + elemInElem
        Next elemInElem
    End If

Next oneElem

End Function
```

PARAMARRAY

- 注意：一個函數只能有一個 ParamArray，而且如果還有其他引數，ParamArray 必須放在最後面。
- 原因很好理解，在 ParamArray 後面的引數，無法判斷是否為 ParamArray 的一份子。
- 同理，兩個 ParamArray 無法判別第一個結束處與第二個開始處。

PARAMARRAY

- 可以執行：

```
'檢查看看 targetValue 有沒有在選擇起來的範圍中
Function hasValue(targetVal As Variant, ParamArray checkRange()) As Integer
hasValue = False
Dim oneElem As Variant, elemInElem As Variant

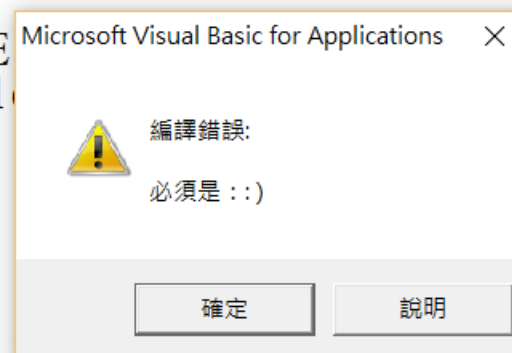
For Each oneElem In checkRange
    If Not IsArray(oneElem) Then
        If targetVal = oneElem Then
            hasValue = True
            Exit Function
        End If
    Else
        For Each elemInElem In oneElem
            If targetVal = elemInElem Then
                hasValue = True
                Exit Function
            End If
        Next elemInElem
    End If
Next oneElem
End Function
```

PARAMARRAY

- ParamArray 在前面：

```
' 檢查看 targetValue 有沒有在選擇起來的範圍中
Function hasValue(ParamArray checkRange(), targetVal As Variant) As Integer
hasValue = False
Dim oneElem As Variant, elemInElem As Variant

For Each oneElem In checkRange
    If Not IsArray(oneElem) Then
        If targetVal = oneElem Then
            hasValue = True
            Exit Function
        End If
    Else
        For Each elemInElem In oneElem
            If targetVal = elemInElem Then
                hasValue = True
                Exit Function
            End If
        Next elemInElem
    End If
Next oneElem
End Function
```

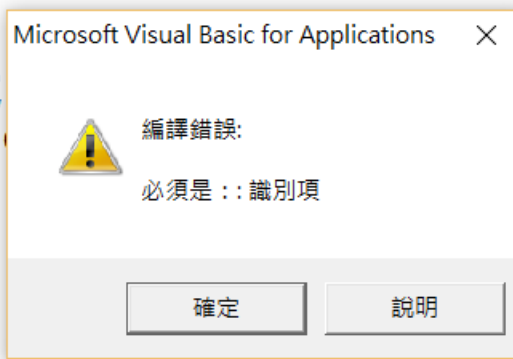


PARAMARRAY

- ParamArray 前面不可加 Optional 修飾，因為其本身一定是 Optional。

```
'檢查看看 targetValue 有沒有在選擇起來的範圍中
Function hasValue(targetVal As Variant, optional ParamArray checkRange()) As Integer
hasValue = False
Dim oneElem As Variant, elemInElem As Variant

For Each oneElem In checkRange
    If Not IsArray(oneElem) Then
        If targetVal = oneElem Then
            hasValue = True
            Exit Function
        End If
    Else
        For Each elemInElem In oneElem
            If targetVal = elemInElem Then
                hasValue = True
                Exit Function
            End If
        Next elemInElem
    End If
End If
```



PARAMARRAY

- 練習：嘗試將前述指數加權平均函數寫為 ParamArray 版本。

CONST 修飾

- 我們的程式有些時候會時常需要用一個 (或一些) 常數。
- 例如估計股價之波動度。
- 我們假設股票的價格服從幾何布朗運動 (Geometric Brownian Motion)

$$dS(t) = \mu S(t)dt + \sigma S(t)dW(t)$$

- 或可表示為：

$$d \log S(t) = \left(\mu - \frac{1}{2} \sigma^2 \right) dt + \sigma dW(t)$$

CONST 修飾

- μ : 股價之年化預期收益率。
 - 實際之預期收益率我們並不知道，但可以透過歷史資料估計。
- σ : 年化波動度，波動度越高時股價之不確定性越大。
 - 可以由歷史資料估計。
 - 或由市場上選擇權報價反推隱含波動度。
- dt : 瞬間變動的年化時間。

CONST 修飾

- $W(t)$ ：布朗運動，為一隨機過程。
 - $W(t) \sim \text{normal}(0, t)$
 - $W(t) - W(s) \sim \text{normal}(0, t - s), \text{ for } t > s$

CONST 修飾

- 假設每經過一天，年化時間的變動都為一個常數： $\frac{1}{365}$
- 在幾何布朗運動下， t 天後之價格為一隨機變數如下：

$$S\left(\frac{t}{365}\right) = S(0)e^{\left(\mu - \frac{\sigma^2}{2}\right)\frac{t}{365} + \sigma W\left(\frac{t}{365}\right)} \sim \text{lognormal}\left(\mu \frac{t}{365}, \sigma^2 \frac{t}{365}\right)$$

- 不論 t 為多少，將 t 天後與 $t - 1$ 天後之價格相除，可以得一天之變動率為下列之隨機變數：

$$S\left(\frac{t}{365}\right)/S\left(\frac{t-1}{365}\right) = e^{\left(\mu - \frac{\sigma^2}{2}\right)\frac{1}{365} + \sigma[W\left(\frac{t}{365}\right) - W\left(\frac{t-1}{365}\right)]} \sim \text{lognormal}\left(\frac{\mu}{365}, \frac{\sigma^2}{365}\right)$$

CONST 修飾

- 假設資產以連續複利的年報酬率為 r ， r 與日報酬率的關係如下：

$$e^{\left(\mu - \frac{\sigma^2}{2}\right) \frac{1}{365} + \sigma \left[W\left(\frac{t}{365}\right) - W\left(\frac{t-1}{365}\right)\right]} = e^{r_t \times \frac{1}{365}}$$

- 可以得到 r_t 為一個期望值為 $\mu - \frac{\sigma^2}{2}$ ，變異數為 $\sigma^2 / \left(\frac{1}{365}\right)$ 之常態分配

CONST 修飾

- 但真正需估計的為 σ ，因此令隨機變數 Y_t 如下：

$$Y_t = r_t \sqrt{\frac{1}{365}}$$

- 估計 Y_t 之標準差 $\hat{\sigma}$ 即為對 σ 之估計。

CONST 修飾

- 而利用估計出之標準差 $\hat{\sigma}$ ，可以得到估計之平均如下：

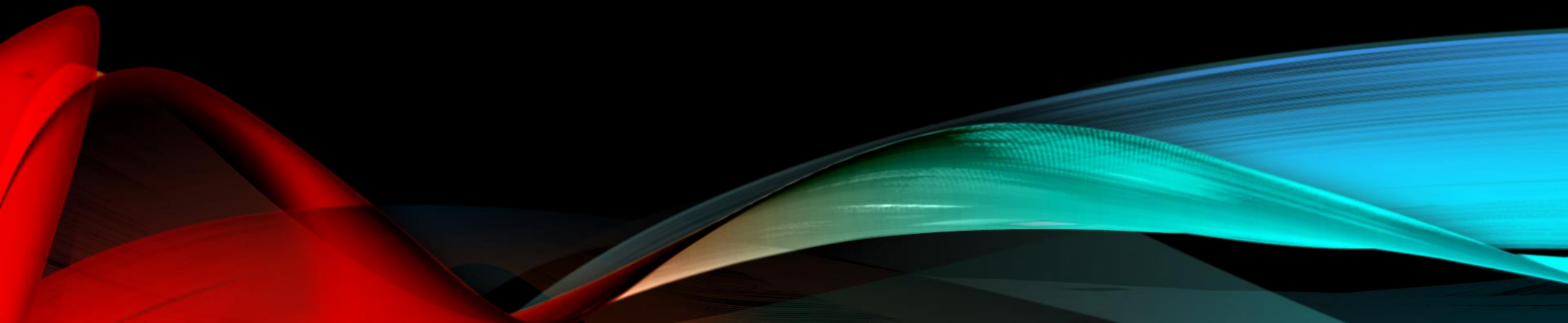
$$\hat{\mu} = \frac{\bar{Y}}{\sqrt{1/365}} + \frac{\hat{\sigma}^2}{2} = \bar{r} + \frac{\hat{\sigma}^2}{2}$$

CONST 修飾

- 程式終會不斷使用到一天之時間差 $\frac{1}{365}$ 。
- 重複輸入下可能容易出錯，因此可以使用一變數 $\text{deltaT} = 1/365$ 。
- 參考 Const 修飾範例一步一步計算歷史波動度。

SECTION 2. 進階語法介紹 (1)

GoTo 與 On Error



GOTO

- GoTo 是一個其他程式語言沒有的有趣功能。
- 當我的程式發生某個狀況時，我會希望他跳到該 script 的某一段。
- 例如，我們想要取前十筆利率大於 6% 之資料，一般想到會利用 for 迴圈。

GOTO

	A	B
1	2000/1/3	5.27
2	2000/1/4	5.27
3	2000/1/5	5.28
4	2000/1/6	5.25
5	2000/1/7	5.22
6	2000/1/10	5.24
7	2000/1/11	5.27
8	2000/1/12	5.29
9	2000/1/13	5.25
10	2000/1/14	5.25
11	2000/1/18	5.24
12	2000/1/19	5.35
13	2000/1/20	5.32
14	2000/1/21	5.31
15	2000/1/24	5.32
16	2000/1/25	5.40
17	2000/1/26	5.41
18	2000/1/27	5.42

GOTO

```
Dim lastRow As Integer: lastRow = Cells(1, 1).End(xlDown).Row
Dim rateValues As Variant: rateValues = Range("B2", Cells(lastRow, "B")).Value
Dim counts As Integer: counts = 0
Dim firstTenGreaterThanOrEqualToSix(10) As Double
Dim i As Integer

For i = 2 To lastRow
    If rateValues(i, 1) > 6 Then
        counts = counts + 1
        firstTenGreaterThanOrEqualToSix(counts) = rateValues(i, 1)
    End If

    If counts = 10 Then GoTo getTenElements
Next i
```

GOTO

- 我們希望：
 - 當資料確實有蒐集到十個時，會回報收集成功。
 - 沒有時會據實報告蒐集個數。
- 因此我們結束後會要期再多做一個判斷。

```
If counts = 10 Then  
    MsgBox "確實蒐集到十筆"  
Else  
    MsgBox "蒐集到 " & counts & " 筆資料"  
End If
```

GOTO

- VBA 的 GoTo 語法提供了另外的方法可以讓程式有不同的寫法。
- 我們希望：
 - 蒐集滿十個是已經確定的事，不用再做一次判斷，因此可以建立一個完成標籤，
`getTenElements`，讓程式直接跳到確定完成的部分。

GOTO

```
For i = 2 To lastRow  
    If rateValues(i, 1) > 6 Then  
        counts = counts + 1  
        firstTenGreaterThanSix(counts) = rateValues(nrow, 1)  
    End If  
  
    If counts = 10 Then GoTo getTenElements  
Next i
```

'確實蒐集十個的標籤:
getTenElements:
 MsgBox "確實蒐集到十筆"

GOTO

- 但這時候會發現一個問題：
- 使用 “data 2” 工作表其實並沒有十筆大於 6% 之資料，但是一樣會跳出確實蒐集完十筆資料之結果，為什麼？
- 標籤下的程式一樣會繼續執行，GoTo 只是幫助跳過中間某些不須執行的段落！！
- 所以個標籤下如果要在該結果停下來，須在該結果結束的過程後加上 Exit Sub，離開該子程序。

```
For i = 2 To lastRow
    If rateValues(i, 1) > 6 Then
        counts = counts + 1
        firstTenGreaterThanSix(counts) = rateValues(i, 1)
    End If

    If counts = 10 Then GoTo getTenElements
Next i
'沒有十個的話:
MsgBox "蒐集到 " & counts & " 筆資料"
Exit Sub
'確實蒐集十個的標籤:
getTenElements:
    MsgBox "確實蒐集到十筆"
```

GOTO

- 許多書籍其實並不建議使用 GoTo，因為容易因程式的更動產生錯誤。
- 只有一種情況是例外，在程式錯誤時進行錯誤處理，如基礎班時說過的取得陣列維度函數即為一應用。