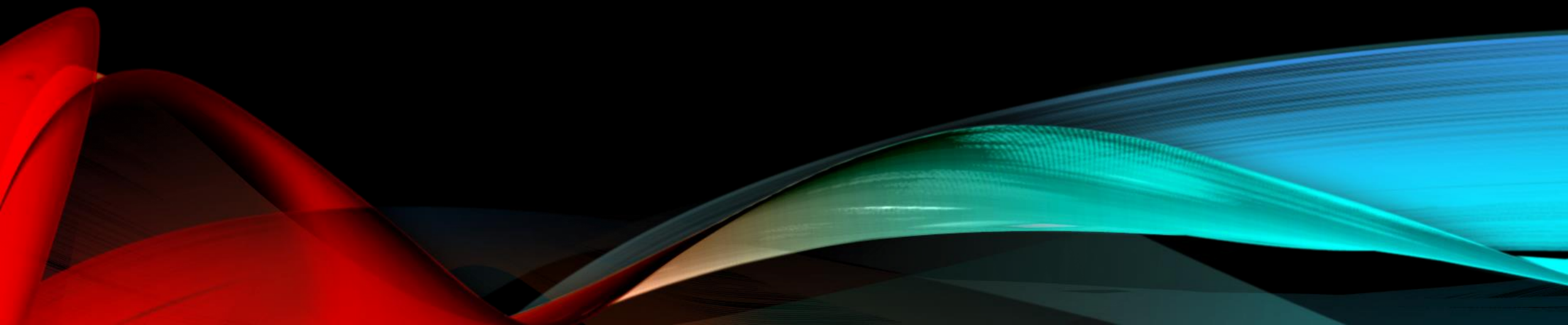
EXCEL VBA 進階班

Lecture 4

SECTION 6. USER-DEFINED TYPE

timeSeries 資料 (1)



USER-DEFINED TYPE

- VBA 本身存在很多種資料型態，但有時我們需要一些更有意義的資料型態，幫助我們更方便處理想要做的事。
- 如現在我們有很多個股的 CSV 檔資料如下：

	A	B	C
1		2330.TW.Adjusted	
2	2010/1/1	50.8372	
3	2010/1/4	51.1525	
4	2010/1/5	50.8372	
5	2010/1/6	51.1525	
6	2010/1/7	50.6007	
7	2010/1/8	50.4431	
8	2010/1/11	50.8372	

USER-DEFINED TYPE

- 可以看到每個個股分別有時間與調整後收盤價兩個資料。
- 我們想要利於利用時間軸來整理資料，例如：
  - 找出位於某段時間之中的股價
  - 找出股價位於某個區間的時間點

USER-DEFINED TYPE

- 之前講過的 Collection 雖然可以做到命名上的便利，但使用增減資料上還是不易。
- 因此我們想要創造一種本身就擁有兩個相對應的資料型態 ：
  - 一組日期的陣列
  - 一組與個日期相對應的價格陣列
- 此時就可以使用 Type 定義自己想要的資料型態。

USER-DEFINED TYPE

- 利用 Type 可以宣告一種自訂的資料型態：

```
Public Type timeSeries
    dates() As Date
    values() As Double
End Type
```

USER-DEFINED TYPE

- 接下來我們可以對該資料型態稍微做一個測試，讀取一個 CSV 檔並把他的資料內容組成一個 timeSeries。

```
Sub createAndPrintTimeSeries()

Dim filePath As String: filePath = Application.GetOpenFilename()
Dim dataFile As Workbook: Set dataFile = Workbooks.Open(filePath)
Dim i As Integer
Dim lastRow As Integer
Dim nData As Integer

dataFile.Worksheets(1).Activate
lastRow = Cells(2, "A").End(xlDown).Row
nData = lastRow - 1

Dim inputDates As Variant: inputDates = Range("A2", Cells(lastRow, "A")).Value
Dim inputAdjClose As Variant: inputAdjClose = Range("B2", Cells(lastRow, "B")).Value

ReDim dates(nData) As Date
ReDim adjClose(nData) As Double

For i = 1 To nData
    dates(i) = inputDates(i, 1)
    adjClose(i) = inputAdjClose(i, 1)
Next i

Dim adjColsedTimeSeries As timeSeries
adjColsedTimeSeries.dates = dates
adjColsedTimeSeries.values = adjClose

dataFile.Close
End Sub
```

USER-DEFINED TYPE

- 在上述程式碼執行完後，我們已經成功建立好一組時間序列資料，不過我們想要對他進行我們想要的作業要怎麼做呢？
- 首先，我們需要一個函數，幫助我們快速的把時間序列資料放回儲存格，以利我們之後觀察想要的結果。
- 現在我們函數與子程序的輸入參數可以使用 timeSeries 了。
- 寫一個函數，把時間序列變回一個二維的 Variant ，以利直接放回工作表。

USER-DEFINED TYPE

```
Function toVariantArray(inputData As timeSeries) As Variant

Dim nData As Integer: nData = UBound(inputData.dates)
Dim i As Integer
ReDim outputArray(nData + 1, 2) As Variant

outputArray(1, 1) = "Date"
outputArray(1, 2) = "Value"

For i = 1 To nData
    outputArray(i + 1, 1) = inputData.dates(i)
    outputArray(i + 1, 2) = inputData.values(i)
Next i

toVariantArray = outputArray

End Function
```

USER-DEFINED TYPE

- 現在我們來看看剛才的時間序列資料內容：

```
Dim timeSeriesData As Variant: timeSeriesData = toVariantArray(adjColsedTimeSeries)
Range("A1", Cells(lastRow, "B")).Value = timeSeriesData
```

- 可以得到結果的確是我們讀取進來的資料。

USER-DEFINED TYPE

- 確定是我們想要的之後，我們可以對時間序列資料做進一步的處理。
- 第一，我們想找位於某一日期時間之內的股價。
- 可以寫成下頁的函數：

USER-DEFINED TYPE

```
Function inPeriod(inputSeries As timeSeries, fromDate As Date, toDate As Date) As timeSeries

Dim originalDates() As Date: originalDates = inputSeries.dates
Dim originalValues() As Double: originalValues = inputSeries.values
Dim nData As Integer: nData = UBound(inputSeries.dates)
Dim counts As Integer
Dim i As Integer
ReDim periodDates(nData) As Date
ReDim periodValues(nData) As Double

For i = 1 To nData
    If originalDates(i) >= fromDate And originalDates(i) <= toDate Then
        counts = counts + 1
        periodDates(counts) = originalDates(i)
        periodValues(counts) = originalValues(i)
    End If
Next i

ReDim Preserve periodDates(counts)
ReDim Preserve periodValues(counts)

Dim outputSeries As timeSeries
outputSeries.dates = periodDates
outputSeries.values = periodValues

inPeriod = outputSeries
End Function
```

USER-DEFINED TYPE

- 可以來試試看，對某一股價只找位於 2015 年一月的資料並列印出來

```
Dim seriesInJan15 As timeSeries: seriesInJan15 = inPeriod(adjColsedTimeSeries, _
                                                          "2015-01-01", _
                                                          "2015-01-31")
Dim timeSeriesData As Variant: timeSeriesData = toVariantArray(seriesInJan15)
Range("A1", Cells(UBound(seriesInJan15.dates) + 1, "B")).Value = timeSeriesData
```

USER-DEFINED TYPE ：練習

- 上述的函數有一個問題，我們一定要給定上下區間才能找，但有時候我只是想找：
  - 位於某一個時間點後的所有資料
  - 位於某一日期前的所有資料
  - 給定一個日期陣列，找兩串日期交集之資料（稍後會更詳細講述）
- 要如何修改函數已達到這個效果？

USER-DEFINED TYPE ：練習

- 提示：當函數 Optional 的引數為 Variant 時，我們可以用 IsMissing 來確認是否有給予其值。
- 當然，使用 IsEmpty 檢查也可以達到相同的效果。
- 試試看下面函數 ：

```
Function f(x As Variant, Optional y As Variant) As Variant
MsgBox IsMissing(y)
f = x
End Function
```

USER-DEFINED TYPE

- 當然，我們現在可以反過來，找股價滿足某一條件的資料。
  - 股價大於某個值
  - 股價大於等於某個值
  - 股價等於某個值
  - 股價不等於某個值
  - 股價小於等於某個值
  - 股價小於某個值

USER-DEFINED TYPE

- 可以利用與前述函數類似的方法建構出一個函數，我們先想只有兩種可能：
  - 股價大於等於某個值
  - 股價小於等於某個值
- 一開始會想到使用 Select Case 與迴圈

```
Function findValue(inputSeries As timeSeries, _
                   comparisonOperator As String, _
                   targetValue As Double) As timeSeries

Dim originalDates() As Date: originalDates = inputSeries.dates
Dim originalValues() As Double: originalValues = inputSeries.values
Dim nData As Integer: nData = UBound(inputSeries.dates)
Dim counts As Integer
Dim i As Integer
ReDim findDates(nData) As Date
ReDim findValues(nData) As Double

Select Case comparisonOperator
    Case ">="
        For i = 1 To nData
            If originalValues(i) >= targetValue Then
                counts = counts + 1
                findDates(counts) = originalDates(i)
                findValues(counts) = originalValues(i)
            End If
        Next i
    Case "<="
         For i = 1 To nData
            If originalValues(i) <= targetValue Then
                counts = counts + 1
                findDates(counts) = originalDates(i)
                findValues(counts) = originalValues(i)
            End If
        Next i
End Select

ReDim Preserve findDates(counts)
ReDim Preserve findValues(counts)

Dim outputSeries As timeSeries
outputSeries.dates = findDates
outputSeries.values = findValues

findValue = outputSeries
End Function
```

USER-DEFINED TYPE

- 我們以日月光的股票測試，找股價大於等於 30 之資料，可以發現確實可行。
- 但這個函數存在幾個問題：
  - 迴圈部分其實是重複的，只是需要的判斷不同
  - 字串判斷會因為不小心打錯字而發生錯誤
- 要如何改善這些問題呢？

USER-DEFINED TYPE

- 首先，避免因字串輸入錯誤造成程式錯誤，我們可以使用 enumerator。

```
Enum comparisonOperator
    less = 0
    lessEqual = 1
    unequal = 2
    greaterEqual = 3
    greater = 4
End Enum
```

USER-DEFINED TYPE

- 再來，各個比較運算其實可以想像成是一個函數，以大於等於為例：
  - 一個函數有 x 與 y 兩個引數
  - 當 x >= y ➔ 回傳 True
  - 反之回傳 False
- 我們可以利用這個想法建立各個比較運算子函數：

USER-DEFINED TYPE

```
Function isLess(elem1 As Double, elem2 As Double) As Boolean
isLess = elem1 < elem2
End Function

Function isLessEqual(elem1 As Double, elem2 As Double) As Boolean
isLessEqual = elem1 <= elem2
End Function

Function isEqual(elem1 As Double, elem2 As Double) As Boolean
isEqual = elem1 = elem2
End Function

Function isUnequal(elem1 As Double, elem2 As Double) As Boolean
isUnequal = elem1 <> elem2
End Function

Function isGreatrEqual(elem1 As Double, elem2 As Double) As Boolean
isGreatrEqual = elem1 >= elem2
End Function

Function isGreatr(elem1 As Double, elem2 As Double) As Boolean
isGreatr = elem1 > elem2
End Function
```

USER-DEFINED TYPE

- 利用 enumerator 與 Run，我們可以避免寫過多重複的迴圈：

```
Function findValue(inputSeries As timeSeries, _
                   operator As comparisonOperator, _
                   targetValue As Double) As timeSeries

Dim originalDates() As Date: originalDates = inputSeries.dates
Dim originalValues() As Double: originalValues = inputSeries.values
Dim nData As Integer: nData = UBound(inputSeries.dates)
Dim counts As Integer
Dim i As Integer
ReDim findDates(nData) As Date
ReDim findValues(nData) As Double
Dim operatorFun As String
Dim x
Select Case operator
    Case less
        operatorFun = "isLess"
    Case lessEqual
        operatorFun = "isLessEqual"
    Case equal
        operatorFun = "isEqual"
    Case unequal
        operatorFun = "isUnequal"
    Case greaterEqual
        operatorFun = "isGreaterEqual"
    Case greater
        operatorFun = "isGreater"
End Select

For i = 1 To nData
    If Application.Run(operatorFun, originalValues(i), targetValue) Then
        counts = counts + 1
        findDates(counts) = originalDates(i)
        findValues(counts) = originalValues(i)
    End If
Next i
```

USER-DEFINED TYPE

- 不過這邊也存在一個抵換關係。
- 如果我們之後有更多地方會用到這種比較運算的函數，其實可以再新增一個函數名為 compare 如下頁所示。
- 原來的函數可以大幅簡化，且以後各個函數都可以輕鬆使用比較運算。

USER-DEFINED TYPE

```
Function compare(elem1 As Double, operator As comparisonOperator, elem2 As Double) As Boolean

dim operatorFun as string

Select Case operator
    Case less
        operatorFun = "isLess"
    Case lessEqual
        operatorFun = "isLessEqual"
    Case equal
        operatorFun = "isEqual"
    Case unequal
        operatorFun = "isUnequal"
    Case greaterEqual
        operatorFun = "isGreaterEqual"
    Case greater
        operatorFun = "isGreater"
End Select

compare = Application.Run(operatorFun, elem1, elem2)

End Function
```

```
Function findValue(inputSeries As timeSeries, _
                   operator As comparisonOperator, _
                   targetValue As Double) As timeSeries

Dim originalDates() As Date: originalDates = inputSeries.dates
Dim originalValues() As Double: originalValues = inputSeries.values
Dim nData As Integer: nData = UBound(inputSeries.dates)
Dim counts As Integer
Dim i As Integer
ReDim findDates(nData) As Date
ReDim findValues(nData) As Double

For i = 1 To nData
    If compare(originalValues(i), operator, targetValue) Then
        counts = counts + 1
        findDates(counts) = originalDates(i)
        findValues(counts) = originalValues(i)
    End If
Next i

ReDim Preserve findDates(counts)
ReDim Preserve findValues(counts)

Dim outputSeries As timeSeries
outputSeries.dates = findDates
outputSeries.values = findValues

findValue = outputSeries
End Function
```

USER-DEFINED TYPE

- 但這裡存在一個問題，每次比較時都必須要重新用 Select Case 找尋要使用的函數，相對來說會使得迴圈應用上重複很多次搜尋的工作，執行時可以明顯感覺速度上的差異。

USER-DEFINED TYPE

- 在沒有選好區間時，結果很容易變為找不到任何符合的值，這時候如果不想程式發生錯誤可以如何處理呢？
- 方法 1. 補 0，把日期與價格皆補上一個 0。

USER-DEFINED TYPE

```
For i = 1 To nData
    If compare(originalValues(i), operator, targetValue) Then
        counts = counts + 1
        findDates(counts) = originalDates(i)
        findValues(counts) = originalValues(i)
    End If
Next i

If counts <> 0 Then
    ReDim Preserve findDates(counts)
    ReDim Preserve findValues(counts)
Else
    ReDim Preserve findDates(1): findDates(1) = 0
    ReDim Preserve findValues(1):  findValues(1) = 0
End If
```

USER-DEFINED TYPE

- 但此處問題在於，有些時候我們可能要處理多筆時間序列，其中可能是有一筆找不到資料的，我們要做其他處理。
- 但該資料與只找到一筆資料的時間虛列其實很難錯區分，要經過很多判斷。

USER-DEFINED TYPE

- 方法 2. 將 dates 與 values 兩組資料改為使用 Variant，當沒資料的時候填入 Null。
- 優點 : Null 本身就是用來表示缺失值的一個方式，容易與陣列型態區分。
- 缺點 : 程式速度會稍微受到影響，而且內部資料因 Variant 型態變得可以傳入不和我們預想資料型態的資料。

USER-DEFINED TYPE

```
Dim adjColsedTimeSeries As timeSeries
adjColsedTimeSeries.dates = dates
adjColsedTimeSeries.values = adjClose
adjColsedTimeSeries.dates = array("A","D","G","C") '出現這樣的情況也不會錯
dataFile.Close
```

USER-DEFINED TYPE：練習

- 上述是 user-defined type 的一大缺點，我們不能防止不小心傳入不符我們設想的資料，因為並沒有一個防範機制。
- 因此強烈建議不要在執行的巨集中直接更改 user-defined type 資料的值，而是透過寫好的函數來變動，每個函數中保有錯誤檢查或防止的機制。
- 因此要件時間序列資料，最好也是靠函數。
- 練習寫一個建構時間序列資料之函數。

USER-DEFINED TYPE：練習

- 時間序列資料現在都假設建構時我們假設日期都已經是由前到後排序完畢，如果不是呢？
- 在建構函數中加入排序功能。

USER-DEFINED TYPE：練習

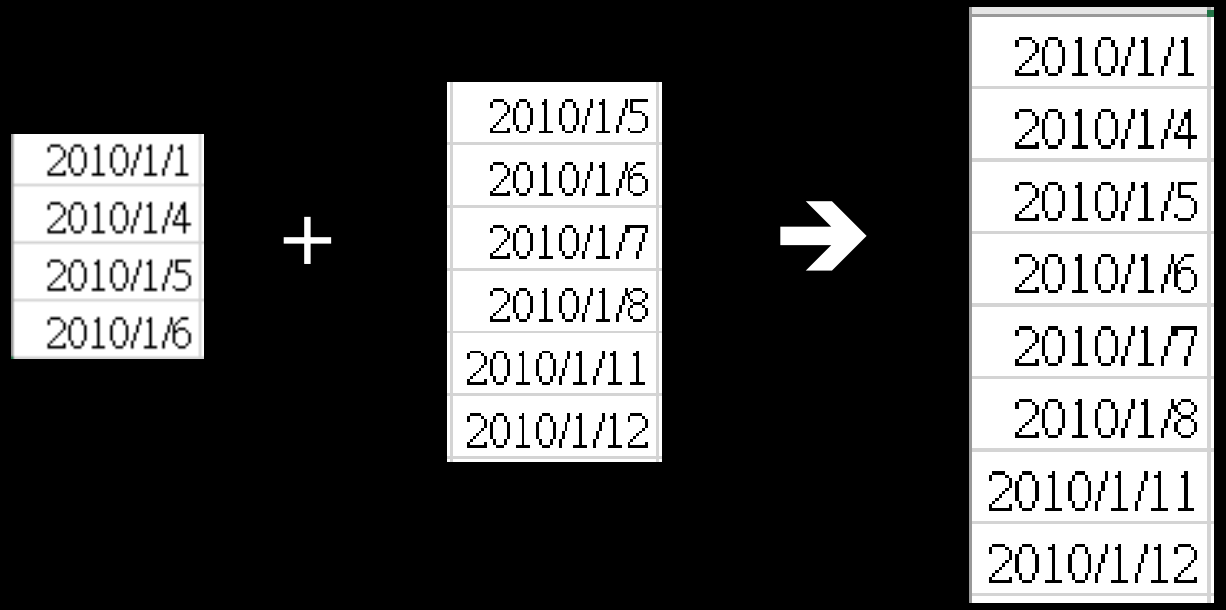
- 我們的 findValue 函數只能找超過某一值的，而沒有落在某區間中的選項。
- 自行寫一個 inRange 函數達到此效果。

USER-DEFINED TYPE：多筆資料整合

- 現在我們想要整合各個時間序列資料，將多筆時間序列在相同時間軸上的合為同一個時間序列資料。
- 但每筆資料可能擁有的時間是不一樣的，我們怎麼把他們整理至同一個時間軸上呢？
- 在這裡介紹 merge algorithm。

USER-DEFINED TYPE：多筆資料整合

- merge algorithm 首先需要的是各個時間序列的日期已經由小到大排序完畢。
- 我們要將兩串日期資料整理成一串由小到大的時間投，且元素不重複。



USER-DEFINED TYPE：多筆資料整合

- 演算法步驟如下：
  - 我們要將 A 與 B 兩個陣列整合為一個且元素不重複，首先將兩陣列由小到大排序（此處可參閱以前過的 bubble sort）。
  - 接下來我們先創造一個空的陣列 C。
  - 當 A 陣列與 B 陣列兩者都還有元素時，我們做以下的動作
    - 如果 A 的第一個元素小於等於 B 的第一個元素 ➔ 將該元素變為 C 的下一個元素 ➔ 清除 A 的第一個元素
    - 反之則將 B 的第一個變為 C 的下一個元素 ➔ 清除 B 的第一個元素
  - 接著將尚未清空的陣列剩下的元素放入 C 陣列即可。

USER-DEFINED TYPE：多筆資料整合

- 將兩者日期處理完之後，各個股票的資料即可按照日期填入。
- 對於有缺失的地方有兩種處理方式：
  - 有任一者有缺失值則刪除該天資料。
  - 將該天資料缺失者補上 Null。

USER-DEFINED TYPE：練習

- 可以整合多筆資料後會出現一個問題，原來的區間函數都沒有辦法用了，那我們要怎麼處理這個問題？