COMP1022Q
Introduction to Computing with Excel VBA

Arrays

David Rossiter, Eddie Chan and Oz Lam

This Presentation

- In this presentation we will look at:
  - What is an array?
  - Lower bound and upper bound
  - Using Option Base 1
  - With…End With
  - Two dimensional arrays
  - Array vs. Range

What is an Array?

- Perhaps the easiest way to think of an array is a row of boxes, into which you can put things
- For example, you can create an integer array, like this:

```
Dim NumberArray(0 To 2) As Integer
```

Index	0	1	2
Value			

- You can put values into the array like this:

```
NumberArray(0) = 10
NumberArray(1) = 14
NumberArray(2) = 3
```

Index	0	1	2
Value	10	14	3

Example of Creating an Array

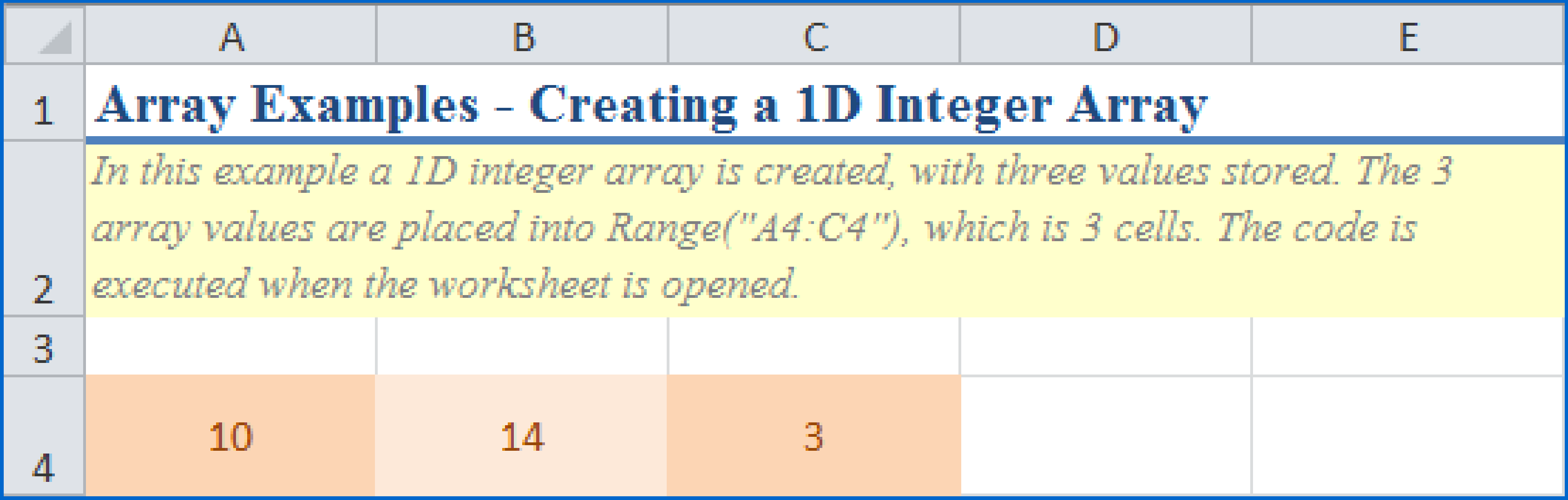
```
Sub Workbook_Open()
    Dim NumberArray(0 To 2) As Integer

    NumberArray(0) = 10
    NumberArray(1) = 14
    NumberArray(2) = 3

    Range("A4:C4").Value = NumberArray
End Sub
```

The index starts from 0

Index	**0**	**1**	**2**
Value	10	14	3

- *You can put multiple values into multiple cells using an array*
- *In this example, each of the cells A4, B4 and C4 stores a value from the `NumberArray`*

Running the Example



	A	B	C	D	E
1	**Array Examples - Creating a 1D Integer Array**				
2	*In this example a 1D integer array is created, with three values stored. The 3 array values are placed into Range("A4:C4"), which is 3 cells. The code is executed when the worksheet is opened.*				
3					
4	10	14	3		

- When the code is executed, an array is created, values are placed in each item of the array, then all three item values are placed in three Excel cells

Array Data Type

- You can have an array of any type that Excel VBA knows about
- For example, you can create the arrays shown below:

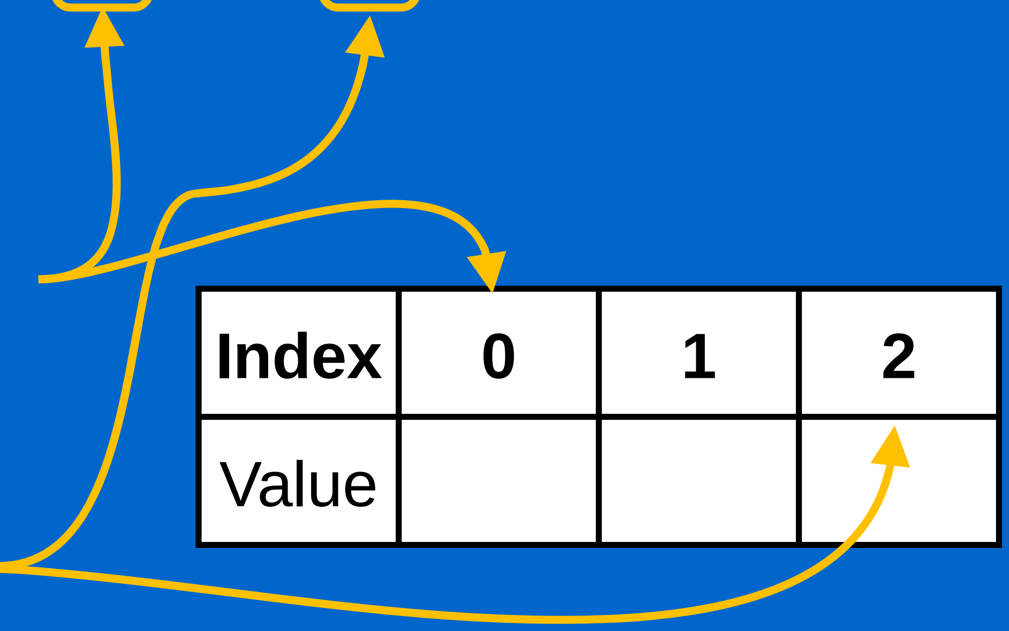
```
Dim MyArray(100) As Long
Dim MyArray(100) As Double
Dim MyArray(100) As Worksheet
Dim MyArray(100) As Range
Dim MyArray(100) As Shape
```

Lower and Upper Bound of an Array

- Previously, we used this code to create an integer array:

```
Dim NumberArray(0 To 2) As Integer
```

- The *lower bound* of an array is the smallest index of the array
- The *upper bound* of an array is the largest index of the array



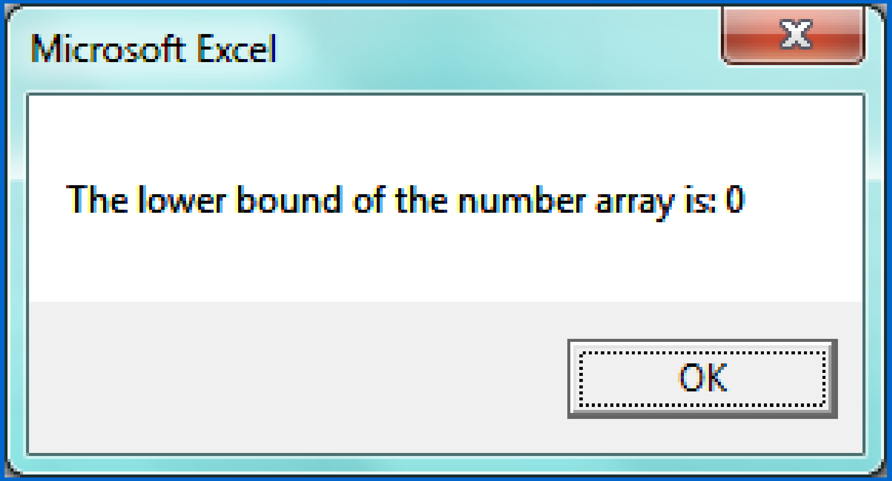
Index	0	1	2
Value			

Lower and Upper Bound of an Array

- `LBound()` returns the lower bound (smallest index) of the array whereas `UBound()` returns the upper bound (largest index) of the array
- The following example shows the lower and upper bound of an array:

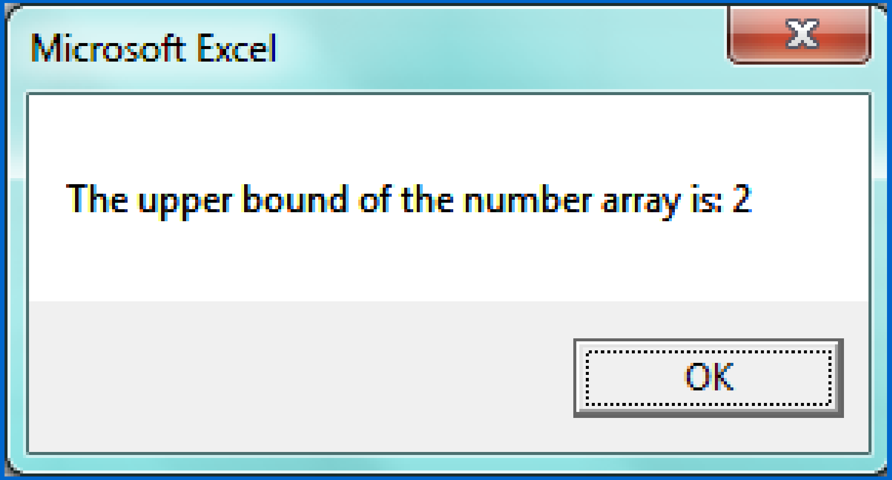
```
Dim NumberArray(0 To 2) As Integer

MsgBox "The lower bound of the number array is: " & _
        LBound(NumberArray)
MsgBox "The upper bound of the number array is: " & _
        UBound(NumberArray)
```

Showing the Lower and Upper Bound



Microsoft Excel

The lower bound of the number array is: 0



Microsoft Excel

The upper bound of the number array is: 2

Default Starting Array Index

- If you do not give a starting index, an array will start the index from 0, for example:

```
Dim NumberArray(2) As Integer
```

Index	0	1	2
Value			

0 is the default starting index

This creates three items in the array, not two. This is because the default starting index is zero and the highest index is 2.

- However, some programmers like to start the index at 1

Using Option Base 1

- You can select the default starting index of arrays to be either `0` or `1` using the `Option Base` command
- Here is an example:

```
Option Base 1
Dim NumberArray(5) As Integer
```

This specifies the default starting index of arrays to be 1

Index	1	2	3	4	5
Value					

The starting index of the array is 1

Example of Using Option Base 1

```
Option Base 1

Sub Workbook_Open()
    Dim NumberArray(3) As Integer

    NumberArray(1) = 10
    NumberArray(2) = 14
    NumberArray(3) = 3

    Range("A4:C4").Value = NumberArray
End Sub
```

3 is the upper bound of the array

The index starts from 1

Index	**1**	**2**	**3**
Value	10	14	3

Creating a Shape Array

- So far, we have learnt how to create and use an array which stores integer values
- A VBA array can be used to store lots of different things, not just numbers
- In the following example, we create an array which stores three triangle shapes using the `Shape` object
- A loop is used to go through the array and draw each triangle onto the worksheet

Example of Shape Array 1/2

Index	1	2	3
Value			

```
Option Base 1

Sub Workbook_Open()
    Dim TriangleArray(3) As Shape
    Dim StartX As Integer, StartY As Integer
    Dim LengthOfSide As Integer, _
        Counter As Integer

    LengthOfSide = 50
    StartX = 100
    StartY = 100
```

Example of Shape Array 2/2

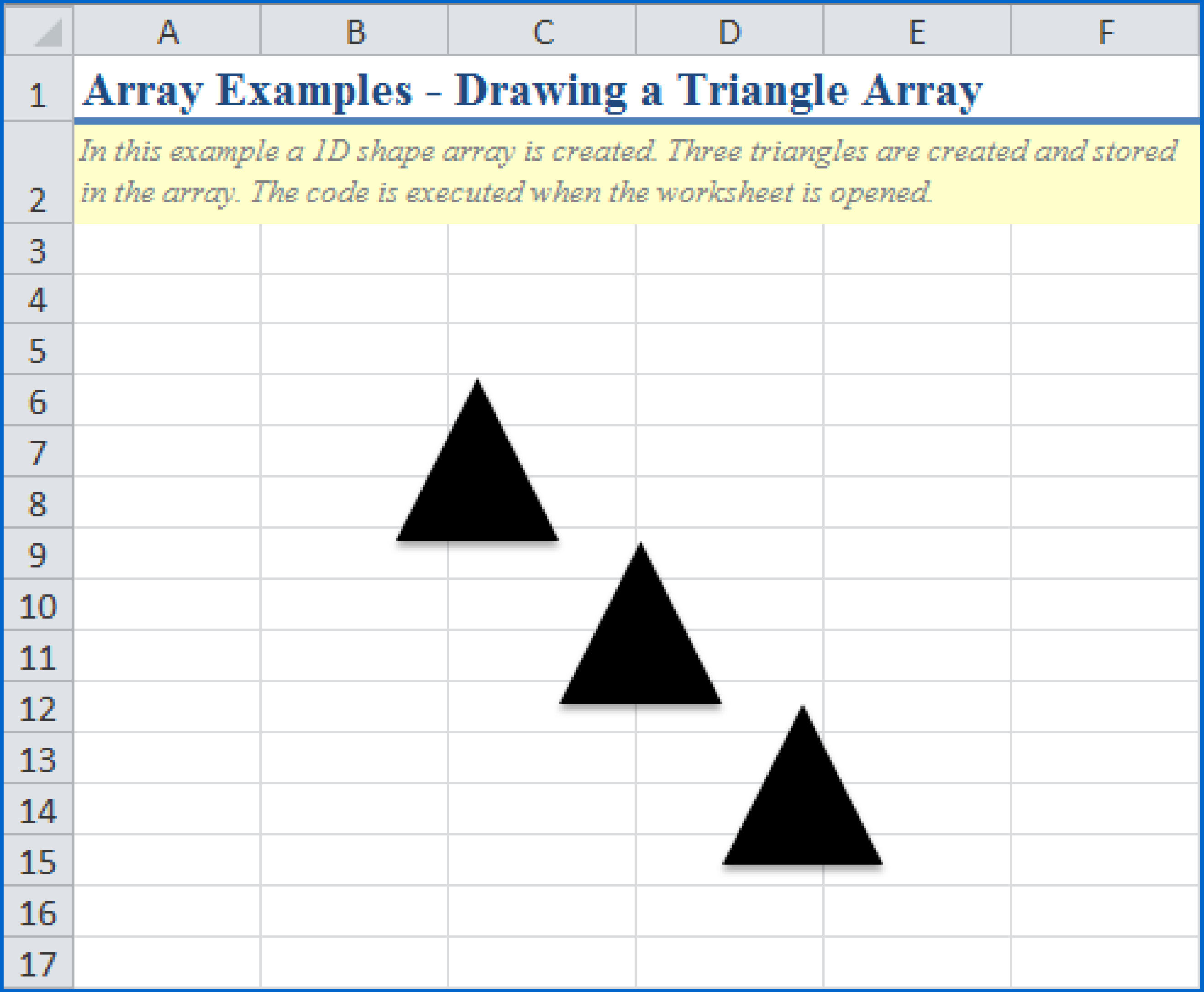
```
    For Counter = LBound(TriangleArray) To _
                  UBound(TriangleArray)
        Set TriangleArray(Counter) = _
            ActiveSheet.Shapes.AddShape( _
            msoShapeIsoscelesTriangle, StartX, _
            StartY, LengthOfSide, LengthOfSide)
        With TriangleArray(Counter)
            .Line.Visible = msoFalse
            .Fill.ForeColor.RGB = vbBlack
            .Fill.Solid
        End With
        StartX = StartX + 50
        StartY = StartY + 50
    Next Counter
End Sub
```

Return 1

Return 3

Set the appearance of the triangle

Example of Using a Shape Array



Array Examples - Drawing a Triangle Array

In this example a 1D shape array is created. Three triangles are created and stored in the array. The code is executed when the worksheet is opened.

With…End With

- In the previous example, the following code is used to change the appearance of a triangle:

```
With TriangleArray(Counter)
    .Line.Visible = msoFalse
    .Fill.ForeColor.RGB = vbBlack
    .Fill.Solid
End With
```

- In the above code `With` … `End With` encloses three lines of code - we saw the use of `With` … `End With` before, when we looked at recorded macros
- Let's reminder ourselves what it does on the next slide

Using With…End With

- `With … End With` is an efficient way to write multiple lines of code which involve the same object
- For example, the following code changes some properties of the same cell B5:

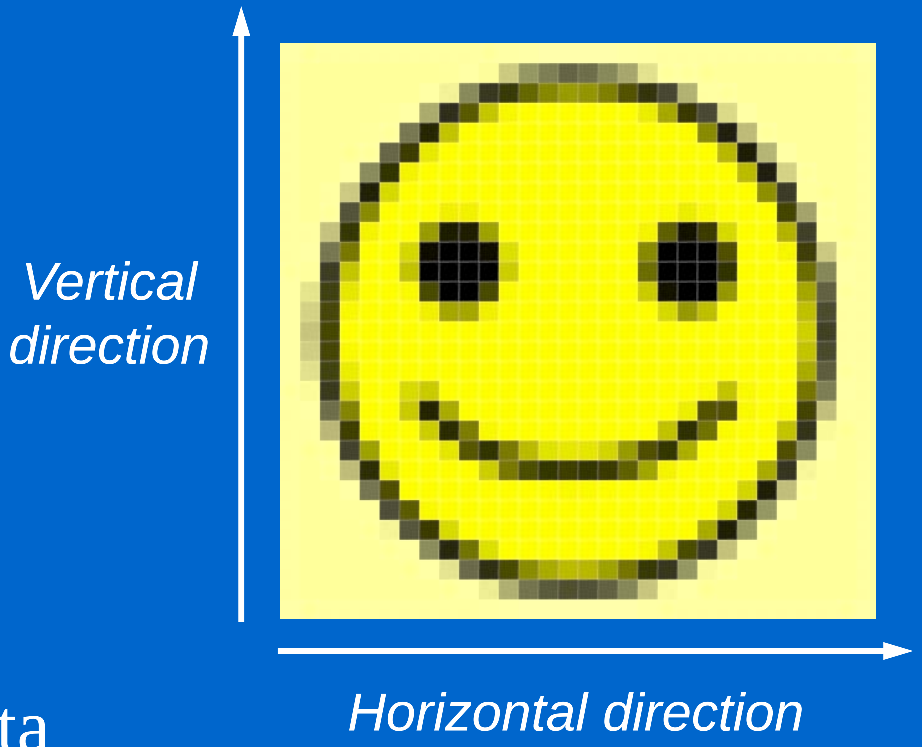
```
Range("B5").Font.Bold = True
Range("B5").Font.Size = 30
Range("B5").Color = vbRed
```

- Instead of writing `Range("B5")` three times, you only need to write it once, like this:

```
With Range("B5")
    .Font.Bold = True
    .Font.Size = 30
    .Color = vbRed
End With
```

Creating a Two Dimensional Array

- Sometimes you need to use a two dimensional (2D) array
- For example, a digital camera image is a 2D structure
- To create a 2D array you can use the `Variant` data type, like this:



```
Dim Number2DArray As Variant
```

Using a Two Dimensional Array

- You can then put values into a 2D array, like this:

```
Number2DArray = _
        [ {10,20,30,40; 50,60,70,80} ]
```

- The above code creates a 2D array with two rows, each row has four items

10	20	30	40
50	60	70	80

- The semi-colon `';'` is used to separate rows
- The comma `','` is used to separate items

Currency Calculator

- Here is a more advanced example of using a 2D array
- Typically, after you go on holiday to another country you have some leftover money in that country's currency
- This example calculates the worth of foreign currency somebody has, in Hong Kong Dollars

Running the Example

Currency Calculator

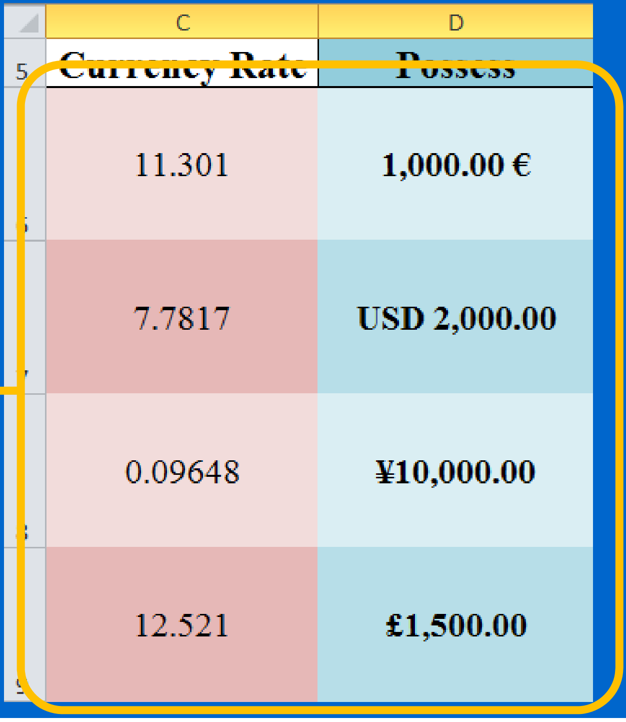
Currency	Image	Currency Rate	Possess
EUR		11.301	1,000.00 €
USD		7.7817	USD 2,000.00
JPY		0.09648	¥10,000.00
GBP		12.521	£1,500.00
		You have:	HK$46,610.70

These values are used in our 2D array

Currency Calculator Example 1/2

```
Option Base 1

Sub Workbook_SheetChange(ByVal Sheet As Object, _
                         ByVal Target As Range)
    Dim CurrencyArray As Variant, _
        SubTotal As Double, Total As Double, _
        Counter As Integer

    CurrencyArray = _
        Range("C6:D9").Value

    Total = 0
```

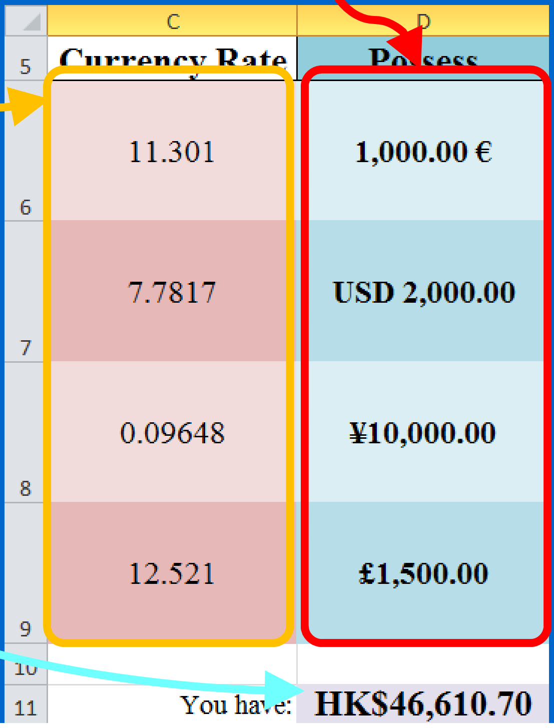
Put the values from the range C6:D9 to the `CurrencyArray` variable, for example, `CurrencyArray(2,1)` will be 7.7817



	C	D
5	Currency Rate	Possess
6	11.301	1,000.00 €
7	7.7817	USD 2,000.00
8	0.09648	¥10,000.00
9	12.521	£1,500.00

Currency Calculator Example 2/2

```
For Counter = LBound(CurrencyArray) To _
              UBound(CurrencyArray)
    ' Calculate the value of a foreign
    ' currency in HK dollars
    SubTotal = CurrencyArray(Counter, 1) * _
               CurrencyArray(Counter, 2)
    Total = Total + SubTotal
Next Counter

' Return the result
Range("D11").Value = Total

End Sub
```



	C	D
5	Currency Rate	Possess
6	11.301	1,000.00 €
7	7.7817	USD 2,000.00
8	0.09648	¥10,000.00
9	12.521	£1,500.00
10		
11	You have:	HK$46,610.70

Using Cells for Storage is Often Better Than Using Arrays for Storage

- A VBA array is not used very often in Excel
- Instead of creating an array to store three numbers, we can simply store the values in cells in a worksheet
- We can easily use worksheets to store a group of 1D or 2D things in cells
- However, most other computer languages are not ‘stuck together’ with worksheets like Excel VBA
- For these other languages, arrays are a lot more useful

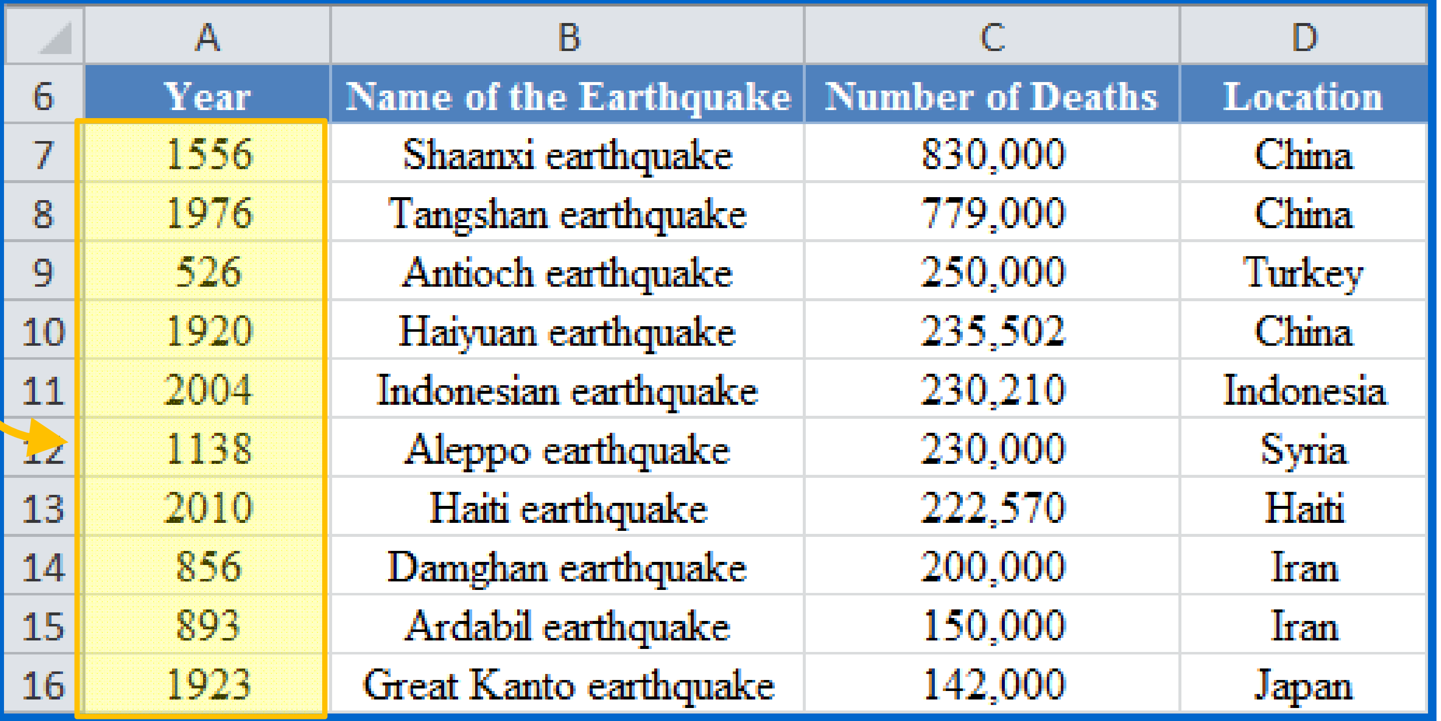
Using Range Methods is Often Better Than Writing Code for Arrays

- Another reason that arrays are not so common in Excel VBA is because of the Range object, which is extremely useful when you are programming VBA
- If you have some data in a Range object, you can use one line of VBA code to sort the data, find something in the data, combine several sets of data, and lots of other useful things
- But if you are using arrays it may take many lines of code to do the same thing

An Example of Using Arrays Compared to Range

- In the next few slides we show a comparison between using arrays and using Range to do the same thing
- First we show the code using Range, then we show the equivalent code using arrays

We want to sort all the data in chronological order of this column



	A	B	C	D
6	Year	Name of the Earthquake	Number of Deaths	Location
7	1556	Shaanxi earthquake	830,000	China
8	1976	Tangshan earthquake	779,000	China
9	526	Antioch earthquake	250,000	Turkey
10	1920	Haiyuan earthquake	235,502	China
11	2004	Indonesian earthquake	230,210	Indonesia
12	1138	Aleppo earthquake	230,000	Syria
13	2010	Haiti earthquake	222,570	Haiti
14	856	Damghan earthquake	200,000	Iran
15	893	Ardabil earthquake	150,000	Iran
16	1923	Great Kanto earthquake	142,000	Japan

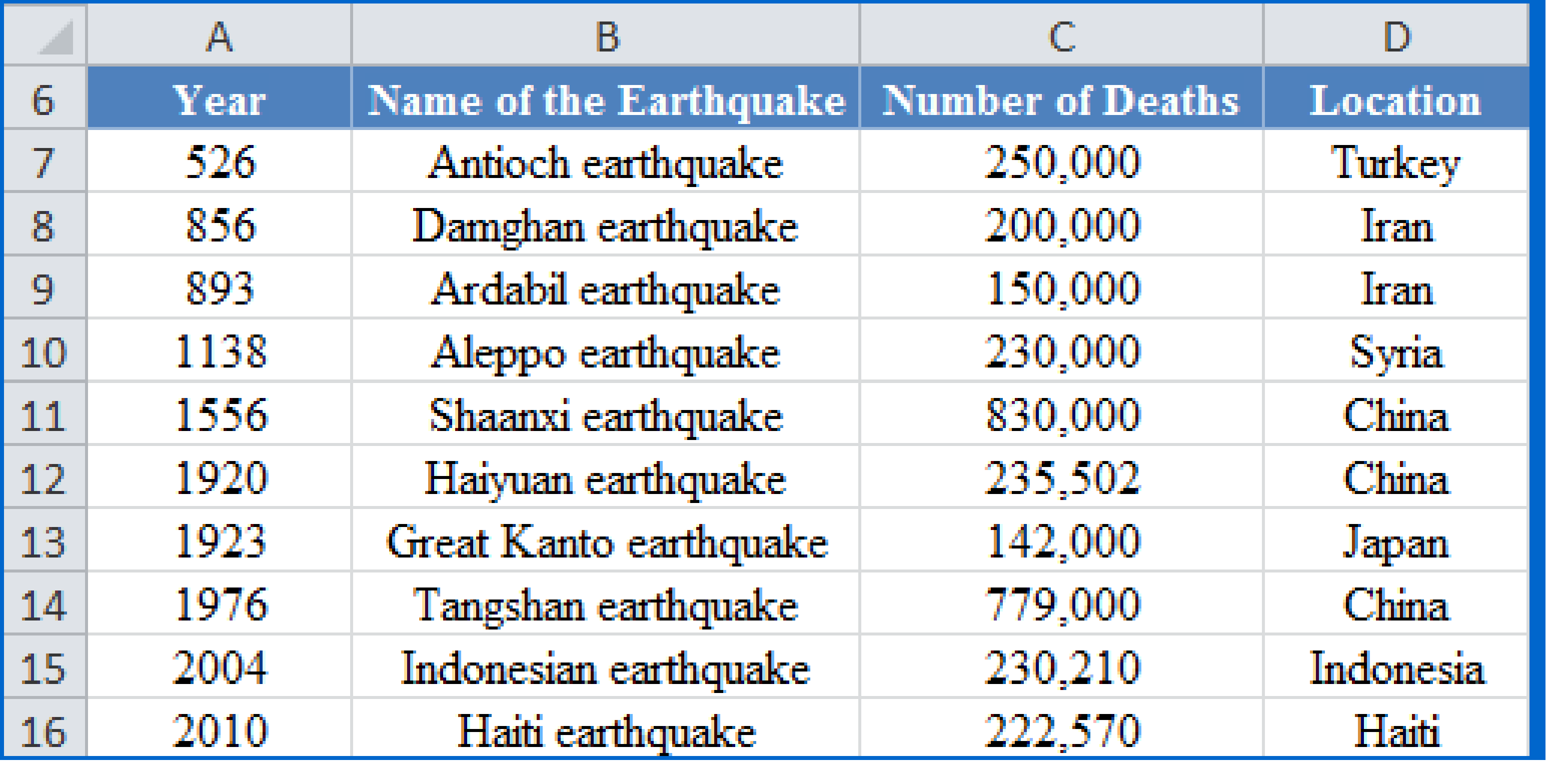
VBA Code for Sorting Using Range

- This code will do what we want:

```
Range("A6:D16").Sort _
                Key1:=Range("A6:A16"), _
                Order1:=xlAscending, _
                Header:=xlYes
```

- Here's the result:

The data is now sorted in ascending time order

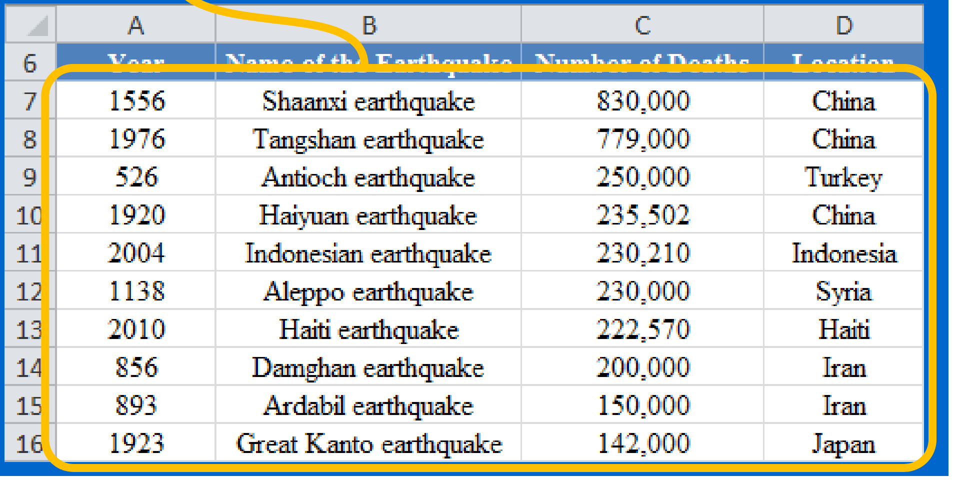


	A	B	C	D
6	Year	Name of the Earthquake	Number of Deaths	Location
7	526	Antioch earthquake	250,000	Turkey
8	856	Damghan earthquake	200,000	Iran
9	893	Ardabil earthquake	150,000	Iran
10	1138	Aleppo earthquake	230,000	Syria
11	1556	Shaanxi earthquake	830,000	China
12	1920	Haiyuan earthquake	235,502	China
13	1923	Great Kanto earthquake	142,000	Japan
14	1976	Tangshan earthquake	779,000	China
15	2004	Indonesian earthquake	230,210	Indonesia
16	2010	Haiti earthquake	222,570	Haiti

VBA Code for Sorting Using Arrays 1/3

- Now let’s look at code which does exactly the same thing, but using arrays only (no Range)
- The code is much larger and more complex

```
Dim EarthquakeArray As Variant
Dim EndRow As Integer, Row As Integer
Dim Column As Integer, SortColumn As Integer
Dim Temp As Variant

EarthquakeArray = Range("A7:D16").Value

' We want to sort the
' year column which is
' the first Column
SortColumn = 1
```



	A	B	C	D
6	Year	Name of the Earthquake	Number of Deaths	Location
7	1556	Shaanxi earthquake	830,000	China
8	1976	Tangshan earthquake	779,000	China
9	526	Antioch earthquake	250,000	Turkey
10	1920	Haiyuan earthquake	235,502	China
11	2004	Indonesian earthquake	230,210	Indonesia
12	1138	Aleppo earthquake	230,000	Syria
13	2010	Haiti earthquake	222,570	Haiti
14	856	Damghan earthquake	200,000	Iran
15	893	Ardabil earthquake	150,000	Iran
16	1923	Great Kanto earthquake	142,000	Japan

VBA Code for Sorting Using Arrays 2/3

```
' Sort the years of earthquake in ascending order
' This code uses the bubblesort algorithm
For EndRow = 1 To UBound(EarthquakeArray, 1) - 1
  For Row = 1 To _
            UBound(EarthquakeArray, 1) - EndRow

    ' If this element is bigger than the one
    ' above it, swap them
    If EarthquakeArray(Row, SortColumn) > _
       EarthquakeArray(Row + 1, SortColumn) Then
```

See next slide

VBA Code for Sorting Using Arrays 3/3

```
        ' Swap the two rows
        For Column = 1 To UBound(EarthquakeArray, 2)
          Temp = EarthquakeArray(Row, Column)
          EarthquakeArray(Row, Column) = _
                  EarthquakeArray(Row + 1, Column)
          EarthquakeArray(Row + 1, Column) = Temp
        Next Column
    End If
  Next Row
Next EndRow

Range("A7:D16") = EarthquakeArray
```

When LBound or Ubound is used with 2D arrays, the second parameter indicates the dimension you want to get the index from