Excel Macros & VBA School

Learn how to automate repetitive or complex tasks using the power of Excel Macros & VBA

MODULE 6

The Big Three

(Workbooks, Worksheets, Ranges)

Module 6 – The Big Three (Workbooks, Worksheets, Ranges)

We cover the basics of workbooks:

- Workbook commands (with no user input)
- Workbook commands (with user input)
- Workbook information (properties)

We learn how to work with worksheets:

- Worksheet information (types and naming)
- Worksheet commands

We discover the realities of ranges:

- Range basics (cells vs ranges, select, write, read, clear)
- Range borders and interiors
- Cell comments
- Range Cut Copy Paste
- Range Insert and Delete
- Range Hide and Unhide
- Rows
- Columns
- Resizing Ranges
- Variable Size Ranges

Make sure you watch the video tutorial lessons for step-by-step instruction on how to use these building blocks.

And check out the projects for some hands on fun.

Workbook Commands (Without User Input)

Create New Workbook

To add a new workbook to the Workbooks collection, use the ".Add" method

```
Workbooks.Add

Or

Dim Wbk_New As Workbook
Set Wbk_New = Workbooks.Add
```

Close Workbook

To close a workbook use the ".Close" method to a workbook object

```
Wbk_New.Close
```

Open Workbook

To open an existing use the ".Open" method and specify Filename (including the Path)

```
Workbooks.Open "C:\Temp\Demo-Workbook.xlsx"

Workbooks.Open Filename:="C:\Temp\Demo-Workbook.xlsx"
```

Save Workbook

To save an open workbook use the ".Save" method

```
Wbk_New.Save
```

Save Workbook As

To save an open workbook as a different filename use the ".SaveAs" method

```
Wbk_New.SaveAs Filename:="C:\Temp\Demo-Workbook-2", _
    FileFormat:=xlOpenXMLWorkbook
```

Official Microsoft documentation here:
https://msdn.microsoft.com/en-us/vba/excel-vba/articles/workbook-saveas-method-excel

Note on File Formats

When you use SaveAs in Excel 2007+ you need to specify the File Format.

You can either use the built-in constant name or the actual numerical value.

Here is a list of common file formats in Excel VBA:

Built-In Constant Name	**Equivalent Constant Value**	**Type of File**
xlOpenXMLWorkbook	51	.xlsx Excel 2007+ (No macros)
xlOpenXMLWorkbookMacroEnabled	52	.xlsm Excel 2007+ (Macro-enabled)
xlExcel12	50	.xlsb Excel 2007+ (Binary workbook)

You can find a comprehensive list on Microsoft's website here:

https://msdn.microsoft.com/en-us/vba/excel-vba/articles/xlfileformat-enumeration-excel

You can read up on the Binary workbook file format here:

http://www.spreadsheet1.com/how-to-save-as-binary-excel-workbook.html

Workbook Commands (With User Input)

Open Workbook chosen by User

To open a workbook with a dialog box use the "GetOpenFilename" method on the Excel Application object

Official Microsoft documentation here:
https://msdn.microsoft.com/VBA/Excel-VBA/articles/application-getopenfilename-method-excel

Save Workbook As File chosen by User

To open a workbook with a dialog box use the "GetSaveAsFilename" method on the Excel Application object

Official Microsoft documentation here:
https://msdn.microsoft.com/en-us/vba/excel-vba/articles/application-getsaveasfilename-method-excel

VBA Sample Code

```
Sub Workbook_Commands_With_User_Input()

Dim FName As String
Dim Wbk As Workbook

'// OPEN WORKBOOK CHOSEN BY USER
FName = Application.GetOpenFilename(FileFilter:="Excel (*.xls*),*.xls*")

Workbooks.Open Filename:=FName

Set Wbk = ActiveWorkbook

'// PROMPT USER TO SAVE AS
FName = Application.GetSaveAsFilename(InitialFileName:="New-workbook", _
    FileFilter:="Excel (*.xlsx),*.xlsx", _
    Title:="Please save your workbook")

If FName <> "False" Then
    Wbk.SaveAs Filename:=FName
    Wbk.Close
End If

End Sub
```

Workbook Information (Properties)

Workbook Name

The property ".Name" is the workbook name

```
Debug.Print ActiveWorkBook.Name
```

Workbook Path

The property ".Name" is the workbook path (to the folder where the workbook is stored)

```
Debug.Print ActiveWorkBook.Path
```

Workbook Full Name

The property ".FullName" is the full name including the workbook path

```
Debug.Print ActiveWorkBook.FullName
```

Workbook Saved?

The property ".Saved" shows True if workbook has been saved since it was modified, and False if workbook has not been saved since it was modified

```
Debug.Print ActiveWorkBook.Saved
```

Workbook Password Protected?

The property ".HasPassword" shows whether the workbook is password protected

```
Debug.Print ActiveWorkBook.HasPassword
```

Worksheet Information (Types and Naming)

"Sheet" types

There are three "sheet" types we mention in this course.

- Sheets (all types of sheets, including worksheets and charts)
- Worksheets (sheets with grids of rows and columns)
- Charts (sheets with only charts on them)

We look at Worksheets in the most detail because they are most useful for data analysis.

Referring to Sheets

The term "Sheets" is used for the collection of all Sheets in a workbook, like this:

```
Thisworkbook.Sheets
```

You can refer to individual sheets in the collection like this:

```
Thisworkbook.Sheets(1)
```

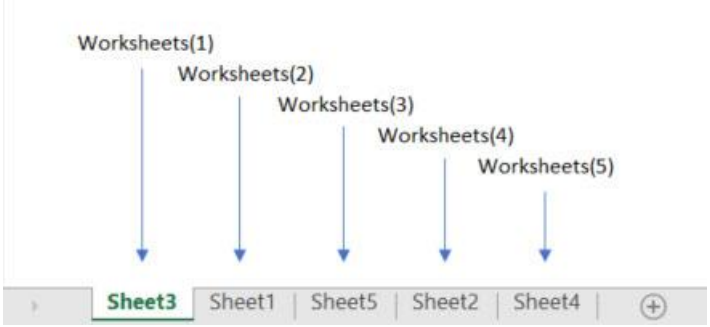
Similarly you can refer to individual worksheets in the collection of worksheets like this:

```
Thisworkbook.Worksheets(1)
```

Worksheets are numbered in the order they appear onscreen. This is different from what their actual names are. See further below for worksheet names.

Let's say you have 5 worksheets in a workbook. The following code picks the worksheet furthest to the right:

```
Thisworkbook.Worksheets(5)
```



As you can see from the screenshot, it doesn't matter what the names are only the position.

Which brings us onto worksheet names…

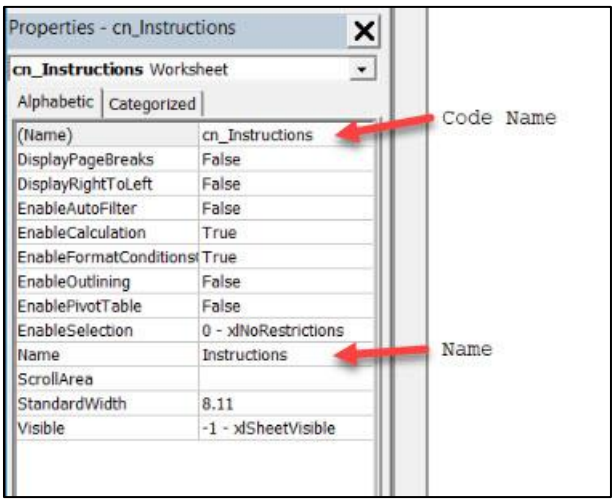
Worksheet names

There are two different name types.

- Codename – only visible in the VBA Editor
- Name – shows on the main Excel screen

They can be the same (e.g. "Sheet1" and "Sheet1") or you can choose different names (e.g. "cn_Instructions" for the codename and "Instructions" for the name).

Here is a screenshot showing the Properties window in the VBA Editor:



To change a worksheet's codename you click in the box next to (Name) in the Properties window.

Codenames are handy when you refer to worksheets in the same workbook as the VBA code.

You cannot refer to a worksheet in a different workbook by its codename.

Activating a worksheet by name

```
cn_Instructions.Activate                    '// Using the codename

Worksheets("Instructions").Activate         '// Using the public name
```

Finding out a worksheet's name with VBA

Here is code that prints the names of worksheet 1 to the Immediate Window:

```
'// WORKSHEET NAME vs CODENAME
With ThisWorkbook
    Debug.Print ""
    Debug.Print .Worksheets(1).Name          '// Public name
    Debug.Print .Worksheets(1).CodeName      '// Code name
End With
```

Worksheet Commands

Add Worksheet

To add a worksheet to the active workbook:

```
      Worksheets.Add
Or    Sheets.Add Type:=xlWorksheet
```

Remember you can also choose which workbook to add to like this:

```
      Workbooks("Module 6 Projects.xlsm").Worksheets.Add
```

Add Chart sheet

To add a chart sheet to the active workbook:

```
      Sheets.Add Type:=xlChart
```

Add Worksheet before

To add a worksheet before the second worksheet

```
      Worksheets.Add before:=.Worksheets(2)
```

Add Worksheet after

Add a Worksheet after the last worksheet

```
      Worksheets.Add after:=.Worksheets(.Worksheets.Count)
```

Add many Worksheets

To add three worksheets after the last worksheet

```
      Worksheets.Add after:=.Worksheets(.Worksheets.Count), Count:=3
```

Delete Sheet

To delete a sheet from the active workbook:

```
      Worksheets(1).Delete
Or    Worksheets("Sheet1").Delete
Or    Charts(1).Delete
```

Delete Last Sheet

To delete the last sheet from the active workbook:

```
      Sheets(.Sheets.Count).Delete
```

This is counting the number of sheets in the workbook to work out which to delete.

Display Alerts

To stop Excel warning you every time you delete sheets:

```
Application.DisplayAlerts = False
```

To turn the automatic warnings back on:

```
Application.DisplayAlerts = True
```

Move Sheet

To move a sheet you specify whether to move before or after, and then specify the sheet you want to move before or after.

Here is sample code to move the first sheet to the end of the workbook, then move the last sheet once space to the left.

```
Dim LastSheet As Long
LastSheet = Wbk.Sheets.Count

With Wbk
    .Sheets(1).Move after:=.Sheets(LastSheet)
    .Sheets(LastSheet).Move before:=.Sheets(LastSheet - 1)
End With
```

Copy Sheet

To copy a sheet you specify whether to move before or after, and then specify the sheet you want to copy before or after.

Here is sample code to copy the first sheet to the end of the workbook, then make a copy of the first sheet at the beginning.

```
Dim LastSheet As Long
LastSheet = Wbk.Sheets.Count

With Wbk
    .Sheets(1).Copy after:=.Sheets(LastSheet)
    .Sheets(1).Copy before:=.Sheets(1)
End With
```

Hide / Show Sheet

The sheet property ".Visible" allows you to specify three different states:

- xlSheetHidden – Can unhide using the Excel user interface
- xlSheetVeryHidden – Cannot unhide using the Excel user interface
- xlSheetVisible – Visible in the Excel user interface

To hide or show the sheet use the following VBA code:

```
Sheets(1).Visible = xlSheetHidden
Sheets(1).Visible = xlSheetVeryHidden
Sheets(1).Visible = xlSheetVisible
```

Protect Sheet

To protect a worksheet with a password:

```
Sheets(1).Protect Password:="ExcelVBA"
```

To unprotect a worksheet with a password:

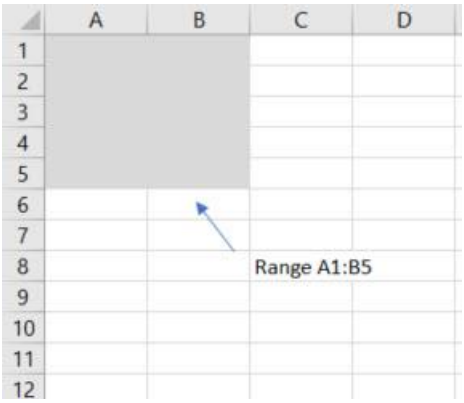
```
Sheets(1).Unprotect Password:="ExcelVBA"
```

Range Basics

Cells vs Ranges

A Range object represents a range of cells in a worksheet.

For example "A1:B5" is a range starting in cell A1, with two columns and five rows.



Officially there is no such thing as a "Cell" object in Excel!

It might seem sensible to expect that "Cells" are objects (or a collection of "Cell" objects) but VBA treats the "Cells" keyword as a property.

When you use the Cells keyword, VBA returns you a Range object.

Read on to understand more...

Select Cell

You can only select one cell at a time using the Cells keyword.

To select a cell you specify its Row and Column as numbers.

Cells (Row , Column)

I find it helps to remember the order of Row, Column by saying to myself the letters RC (which can stand for Radio Control or Rubik's Cube or Roman Catholic – pick one that helps you most)

Here are examples:

```
Cells(5, 1).Select              '// Select cell A5
Cells(1, 3).Select              '// Select cell C1
```

Select Range

You can select multiple cells in a continuous range using the .Select method

```
Range("A1:A10").Select
```

To select multiple ranges, separate them with a comma inside the quote marks:

```
Range("A1:A10, C1:C10").Select
```

When you select multiple ranges, the first cell becomes the active cell:

```
Range("C10, B9, A8, B7, C6, B5").Select
```

In the above example, C10 becomes the active cell.

Write to Range

To write to a range you can assign a value like this:

```
Range("C1") = "Knock knock"
```

You can also set the .value property like this:

```
Range("C2").Value = "Who's there?"
```

You can mix Range and Cells to write to many cells at once.

```
Range(Cells(1, 1), Cells(3, 3)) = "Hello"
```

Write to Range (using Cells)

To write values to multiple cells you can use a For loop.

This is where the R, C parameters of Cells (Row, Column) come in handy.

You can use a variable to loop through a counter.

```
Dim Counter As Long
For Counter = 1 To 10
    Cells(Counter, 2).Value = Counter
Next Counter
```

Here is the output of that code:

	A	B	C	D
1		1		
2		2		
3		3		
4		4		
5		5		
6		6		
7		7		
8		8		
9		9		
10		10		
11				
12				
13				

Read directly from Range

To read the value in a cell:

```
myValue1 = Rng.Value
```

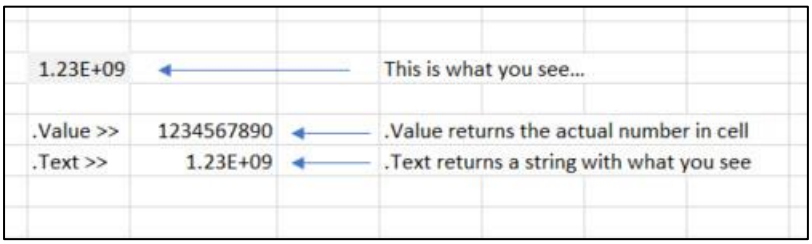
To read the text in a cell:

```
myValue2 = Rng.Text
```

The .Value represents the actual content of the cell (whether it's a number or text).

The .Text represents what is visible in the Excel window (which might be different).

Here is an example to illustrate:



Using "Cells" to Read from Range

Use a For loop to read from different cells:

```
For Counter = 1 To 10
    myValue1 = Cells(Counter, 3).Value
    Cells(Counter, 6).Value = myValue1
Next Counter
```

Clear Range

To clear contents and formatting:

```
Range("A1:C1").Clear
```

To clear contents only:

```
Range("A1:C1").ClearContents
```

To clear formatting only:

```
Range("A1:C1").ClearFormats
```

To clear contents and formatting for a whole worksheet:

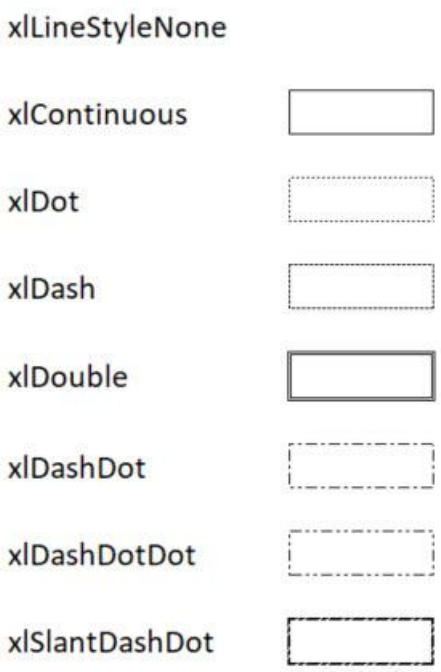
```
Worksheets(1).Cells.Clear
```

Range Borders and Interiors

Change Border Style

It can be very useful to change the border around the cells of a range to highlight that cell.

Here are different styles of border you can set:



To set the border change the .LineStyle property. Here is sample code:

```
Dim Rng As Range
Set Rng = cn_Ranges.Range("B2:E10")

With Rng.Borders
    .LineStyle = xlContinuous
    .LineStyle = xlDot
    .LineStyle = xlDash
    .LineStyle = xlDouble
    .LineStyle = xlDashDot
    .LineStyle = xlDashDotDot
    .LineStyle = xlSlantDashDot
End With
```

Clear Border

To clear the borders:

```
Rng.Borders.Linestyle = xlLineStyleNone
```

Change Border Line Thickness

To change the border line thickness change the .Weight property. Here is sample code:

```
Dim Rng As Range
Set Rng = cn_Ranges.Range("B2:E10")

With Rng.Borders
    .Weight = xlHairline
    .Weight = xlThin
    .Weight = xlMedium
    .Weight = xlThick
End With
```

Change Border Color

To change the border color change the .Color property.

You can use RGB(red, green, blue) to specify a color.

Here is sample code that changes line style, color and weight:

```
Dim Rng As Range
Set Rng = cn_Ranges.Range("B2:E10")

With Rng.Borders
    .LineStyle = xlContinuous
    .Color = RGB(40, 40, 180)
    .Weight = xlThin
End With
```

Set Border Around Outside

To set a border around only the outside of a range:

```
Range(B2:E10").BorderAround LineStyle:=xlContinuous, Weight:=xlThick
```

Set Range Background Color

To set the range background color change the .Color property of the .Interior property:

```
Range(B2:E10").Interior.Color = RGB(170, 215, 235)
```

Clear Formatting

Remember, to clear the formatting but not the contents of a range:

```
Range(B2:E10").ClearFormats
```

Cell Comments

Add Comment to Cell

To add a comment to a cell:

```
Dim Rng As Range
Set Rng = Range("C5")
Rng.AddComment ("I'm just a simple comment...")
```

Show / Hide Comment

To make a comment visible:

```
Rng.Comment.Visible = True
```

To hide a comment:

```
Rng.Comment.Visible = False
```

Read Comment Author

To show the comment author in a Message Box:

```
MsgBox prompt:="Comment author: " & Rng.Comment.Author
```

Read Comment Text

To show the comment text in a Message Box:

```
MsgBox prompt:="Comment text: " & Rng.Comment.Text
```

Add Many Comments

To add comments in cells C5:C15 using a For Loop:

```
Dim Counter As Long
For Counter = 5 To 15
    Cells(Counter, 3).AddComment ("I'm just a comment...")
    Cells(Counter, 3).Comment.Visible = True
Next Counter
```

To Delete a Comment

Check there is a comment in the range using a double negative. If there is no comment and you try to delete a comment then Excel reports an error:

```
Dim rng As Range
Set rng = ActiveSheet.Cells(4, 4)
If Not (rng.Comment Is Nothing) Then rng.Comment.Delete
```

Range Cut Copy Paste

Cut and Paste Range

To cut and paste a range you need to specify a paste destination:

```
Dim Rng As Range
Set Rng = Range("A1:A10")
Rng.Cut Destination:=Range("C1")
```

Copy and Paste Range

To copy and paste a range you need to specify a paste destination:

```
Set Rng = Range("C1:C10")
Rng.Copy Destination:=Range("E1")
```

Autofit Column Width

After you paste to the new range you can get Excel to autofit the Column Width to match the contents:

```
Range("E1").Columns.AutoFit
```

Paste Special

To paste a specific aspect of the original range, use the Paste Special method.

This is like using Paste Special in Excel.

First copy the range:

```
Rng.Copy
```

Then specify a destination range to paste:

```
Range("D2").PasteSpecial xlPasteAll
Range("D3").PasteSpecial xlPasteValues
Range("D4").PasteSpecial xlPasteFormats
Range("D5").PasteSpecial xlPasteComments
Range("D6").PasteSpecial xlPasteColumnWidths
Range("D7").PasteSpecial xlPasteAllExceptBorders
Range("D8").PasteSpecial xlPasteValuesAndNumberFormats
```

Clear Cut / Copy Mode

After you Cut or Copy a range, Excel remembers that range and displays it with an animated border that looks like "marching ants". To clear that range from selection:

```
Application.CutCopyMode = False     '// Clear Cut Copy Mode
```

Range Insert and Delete

Insert a Copied Range

To copy range A1:J2 and insert it into range A8:J8 (while shifting that range down):

```
Range("A1:J2").Copy
Range("A8:J8").Insert shift:=xlShiftDown
```

To copy range K1:K16 and insert it into range F1:F16 (while shifting that range right):

```
Range("K1:K16").Copy
Range("F1:F16").Insert shift:=xlShiftToRight
```

Insert Rows and Columns

To insert 3 rows at Row 6:

```
Range("6:9").Insert
```

To insert 2 columns at Column C:

```
Range("C:D").EntireColumn.Insert
```

Delete Range

To delete row 6:

```
Range("6:6").Delete
```

To delete range E4 and shift cells up:

```
Range("E4").Delete shift:=xlShiftUp
```

To delete range E5 and shift cells to the left:

```
Range("E5").Delete shift:=xlShiftToLeft
```

Range Hide and Unhide

Hide Row or Column

To hide row 5 and column 5:

```
Rows(5).Hidden = True
Columns(5).Hidden = True
```

Unhide Row or Column

To unhide row 5 and column 5:

```
Rows(5).Hidden = False
Columns(5).Hidden = False
```

Unhide all Rows or Columns

To unhide all rows and columns:

```
Rows.Hidden = False                 '// Unhide all rows
Columns.Hidden = False              '// Unhide all columns
Cells.Hidden = False                '// This unhides everything
```

Hide Many Rows or Columns

To hide rows 2 to 9 and columns 2 to 9 using a For loop:

```
For Counter = 2 To 9
    Rows(Counter).Hidden = True
    Columns(Counter).Hidden = True
Next Counter
```

Rows

Select Rows

To select a single row, specify the row number:

```
Rows(1).Select
```

To select an entire row from a single cell reference:

```
Range("A5").EntireRow.Select
```

To select rows 1 to 10 using quote marks:

```
Range("1:10").Select
```

To select rows 1, 3, 5, 7 and 9 using quote marks:

```
Range("1:1, 3:3, 5:5, 7:7, 9:9").Select
```

Set Row Height

To set row height to a specific value:

```
Rows(1).RowHeight = 40
```

Set Height of Many Rows

To set rows 1 to 10 to row height 40:

```
Dim Wks As Worksheet
Set Wks = Worksheets(1)

Dim Counter As Long
For Counter = 1 To 10
    Wks.Rows(Counter).RowHeight = 40
Next Counter
```

Set Row to Standard Height

To reset rows 1 to 10 to standard height:

```
Dim Wks As Worksheet
Set Wks = Worksheets(1)

Dim Counter As Long
For Counter = 1 To 10
    Wks.Rows(Counter).RowHeight = Wks.StandardHeight
Next Counter
```

Reset All Rows Heights

To reset all the row heights on a worksheet:

```
Wks.Rows.UseStandardHeight = True
```

Columns

Select Columns

To select a single column, specify the column number:

```
Columns(1).Select
```

To select an entire column from a single cell reference:

```
Range("C1").EntireColumn.Select
```

To select columns A to J using quote marks:

```
Range("A:J").Select
```

To select columns A, C, E, G, I using quote marks:

```
Range("A:A, C:C, E:E, G:G, I:I").Select
```

Set Column Width

To set column width to a specific value:

```
Columns(1).ColumnWidth = 2.5
```

Set Width of Many Columns

To set columns 1 to 10 to column width 2.5:

```
Dim Wks As Worksheet
Set Wks = Worksheets(1)

Dim Counter As Long
For Counter = 1 To 10
    Wks.Columns(Counter).ColumnWidth = 2.5
Next Counter
```

Set Column to Standard Width

To reset columns 1 to 10 to standard width:

```
Dim Wks As Worksheet
Set Wks = Worksheets(1)

Dim Counter As Long
For Counter = 1 To 10
    Wks.Columns(Counter).ColumnWidth = Wks.StandardWidth
Next Counter
```

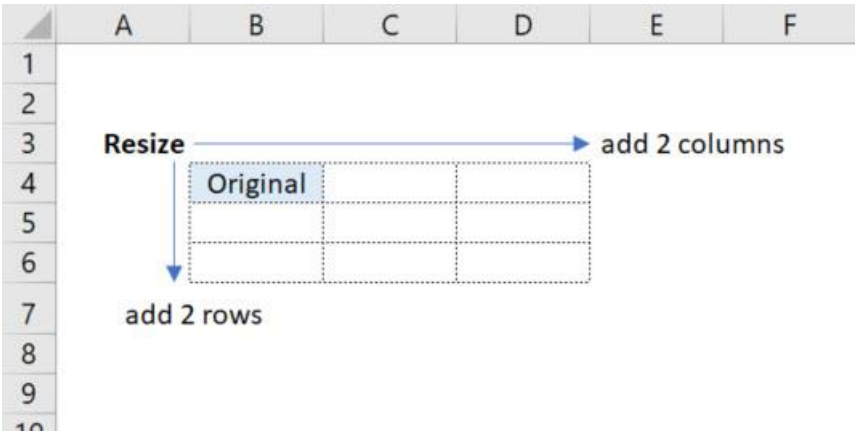
Reset All Column Widths

To reset all the row heights on a worksheet:

```
Wks.Columns.UseStandardWidth = True
```

Resizing Ranges

Range Resize



To resize a range specify the new rowsize and columnsize as whole numbers:

```
Set Rng = Rng.Resize(rowsize:=3, columnsize:=3)
```

Here is code that sets Rng to range "B4" then resizes it to range "B4:D6" by adding 2 rows and 2 columns:

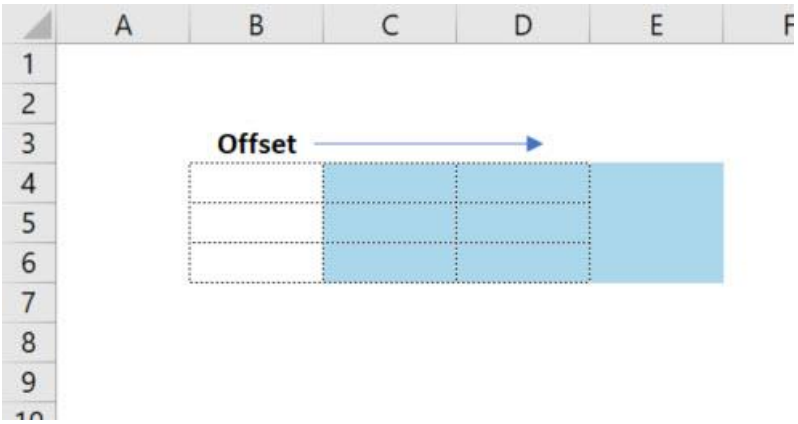
```
Dim nRows As Long
Dim nCols As Long
Dim Rng As Range

Set Rng = Range("B4")

nRows = Rng.Rows.Count + 2
nCols = Rng.Columns.Count + 2

Set Rng = Rng.Resize(rowsize:=nRows, columnsize:=nCols)
Rng.Borders.LineStyle = xlDot
```

Range Offset



To choose a new range that is offset from the current range specify the row offset and column offset as whole numbers:

```
Rng2 = Rng.Offset(row offset, column offset)
```

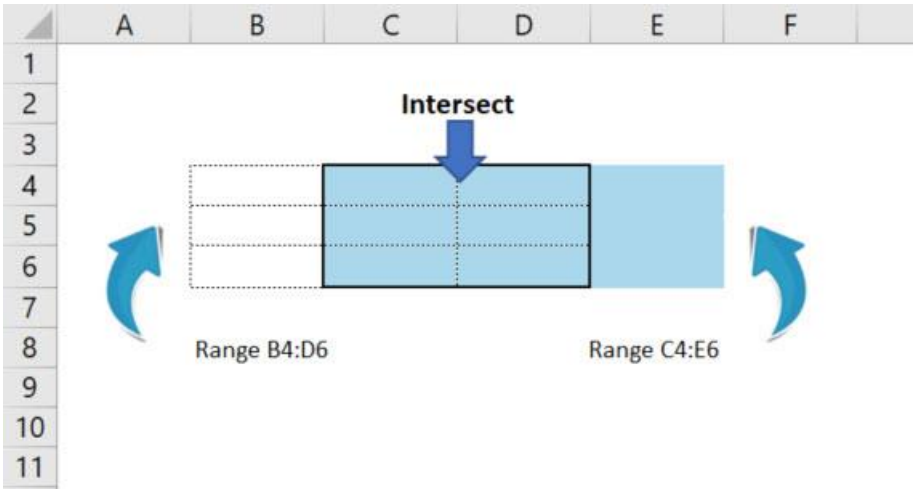
Here is code that sets Rng to range "B4:D6" then sets Rng2 to range "C4:E6" by offset 1 column to the right:

```
Dim Rng As Range
Set Rng = Range("B4:D6")

Dim Rng2 As Range
Set Rng2 = Rng.Offset(0, 1)

Rng2.Cells.Interior.Color = RGB(170, 215, 235)  '// Shade in blue
```

Range Intersect



To choose a new range that comes from the intersection of two ranges specify the first range and second range:

```
Set Rng3 = Intersect(Rng, Rng2)
```

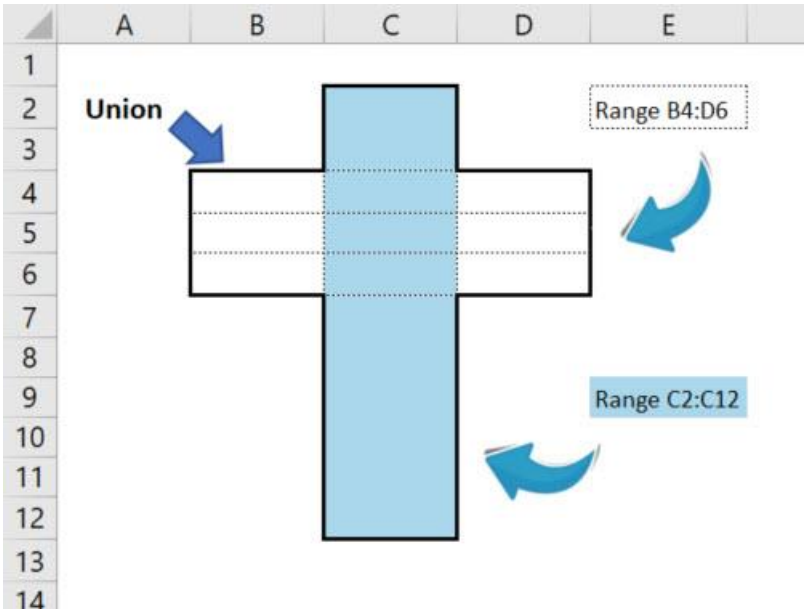
Here is code that sets Rng to range "B4:D6" then sets Rng2 to range "C4:E6" then sets Rng3 as the intersection of Rng and Rng2:

```
Dim Rng As Range
Set Rng = Range("B4:D6")

Dim Rng2 As Range
Set Rng2 = Range("C4:E6")

Dim Rng3 As Range
Set Rng3 = Intersect(Rng, Rng2)
```

Range Union



To choose a new range that comes from joining two ranges:

```
Set Rng3 = Union(Rng, Rng2)
```

Here is code that sets Rng to range "B4:D6" then sets Rng2 to range "C2:C12" then sets Rng3 as the joining of Rng and Rng2:

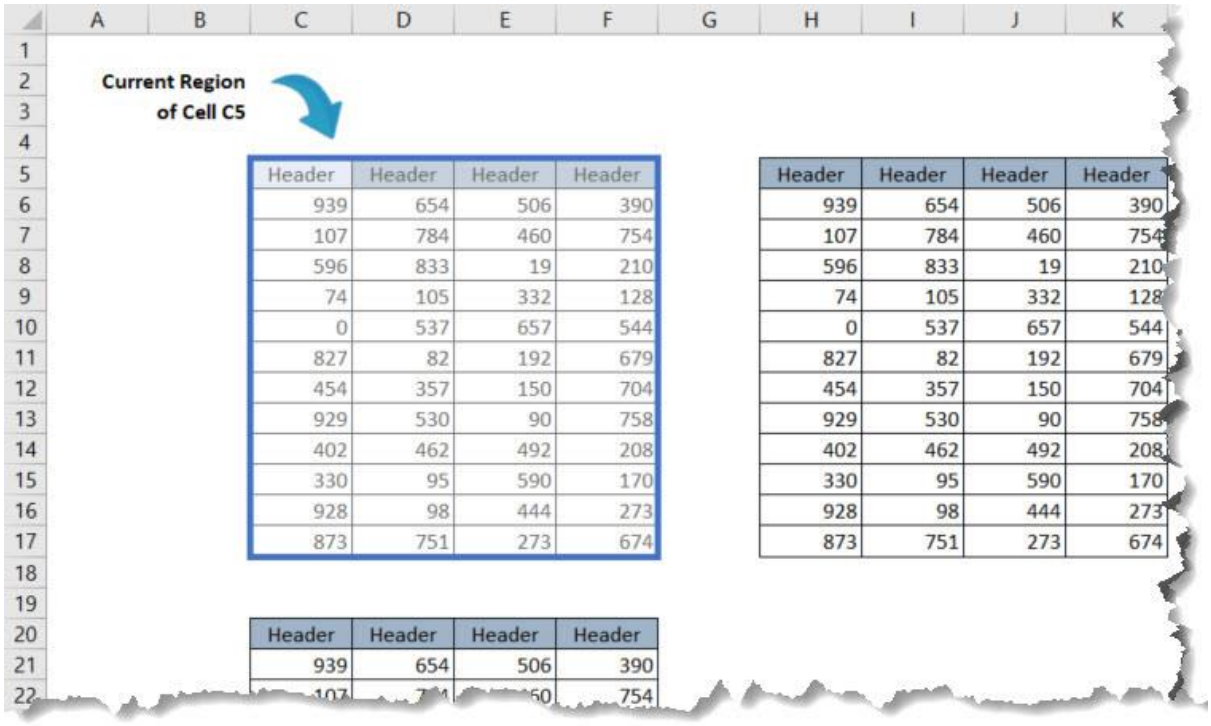
```
Dim Rng As Range
Set Rng = Range("B4:D6")

Dim Rng2 As Range
Set Rng2 = Range("C2:C12")

Dim Rng3 As Range
Set Rng3 = Union(Rng, Rng2)
```

Variable Size Ranges

Current Region



When you deal with ranges sometimes you want to select all the cells with data that are surrounding your current range.

To do this you use the **Current Region** property.

This selects the surrounding region that is bounded by blank rows and columns.

It has the same effect as choosing Home > Editing > Find & Select > GoTo Special and choosing the Current Region option.

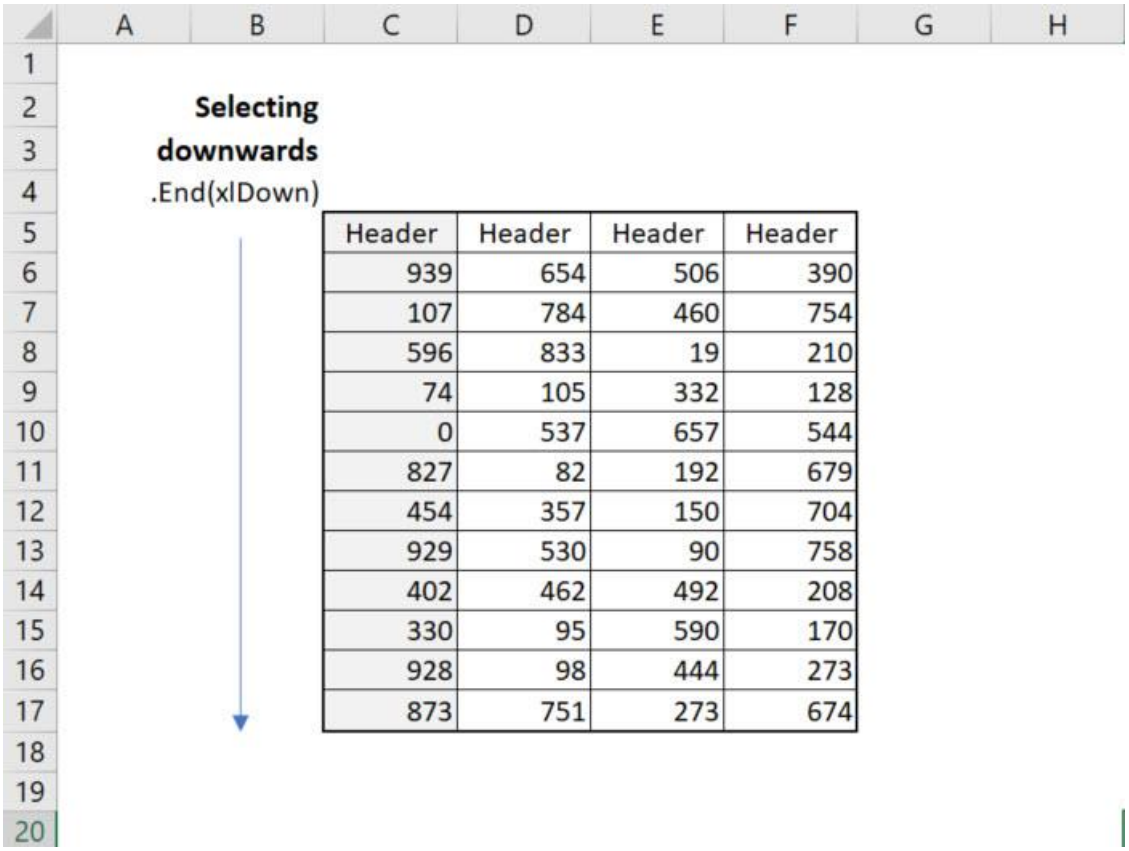
To expand your range to the Current Region surrounding your existing range:

```
Set Rng = Range("A1").CurrentRegion
```

Here is code that copies the current region around range "C5" to range "H5":

```
Dim Rng As Range
Set Rng = Range("C5")
Rng.CurrentRegion.Copy Range("H5")
```

Selecting to End of Column



Header	Header	Header	Header
939	654	506	390
107	784	460	754
596	833	19	210
74	105	332	128
0	537	657	544
827	82	192	679
454	357	150	704
929	530	90	758
402	462	492	208
330	95	590	170
928	98	444	273
873	751	273	674

If you want to select to the end of the current column of cells you can't use the Current Region property.

Instead use the **End(xlDown)** method. This is like pressing CTRL + down arrow in Excel.

To move down from the active cell to the last cell in a column of cells:

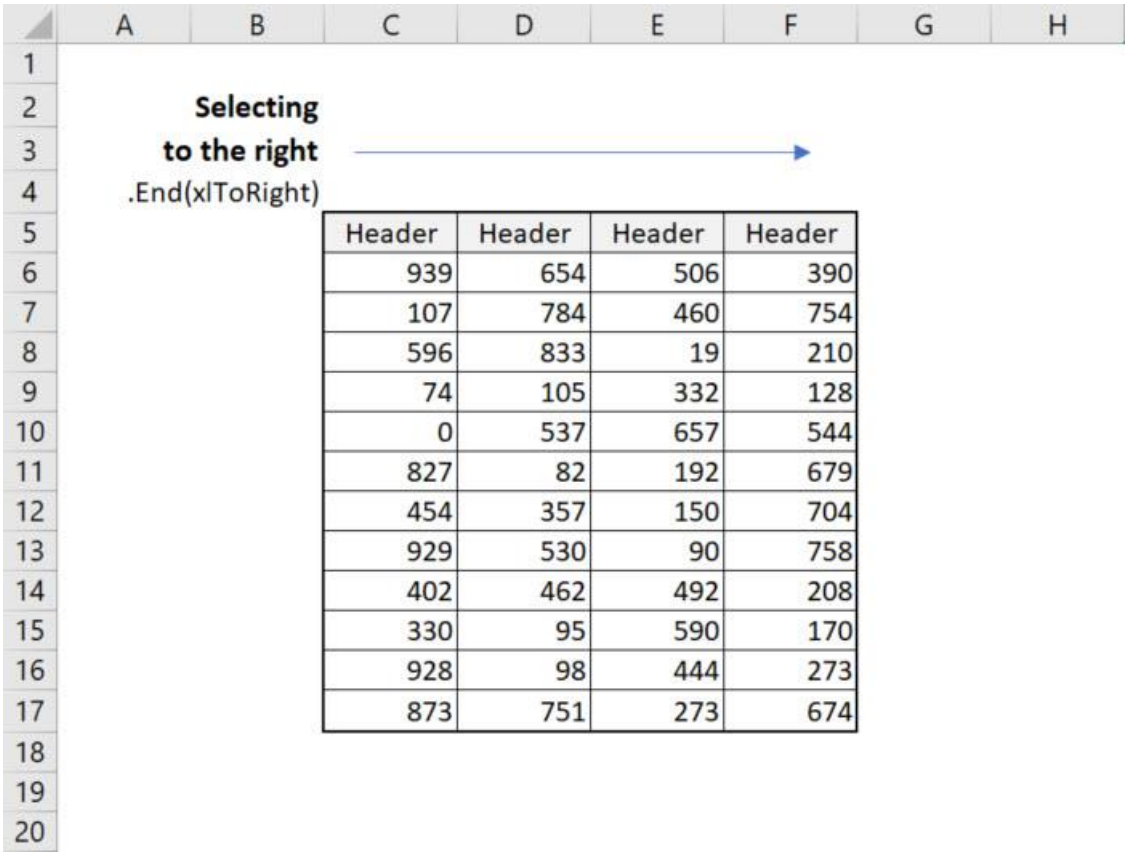
```
ActiveCell.End(xlDown).Select
```

To select a range combine Range with the **End** method. This is like pressing CTRL + SHIFT + down arrow in Excel.

To see what I mean when I say you combine Range with the End method, here is code that selects downwards from the active cell:

```
Cells(1, 1).Select
Range(ActiveCell, ActiveCell.End(xlDown)).Select
```

Selecting to End of Row



Header	Header	Header	Header
939	654	506	390
107	784	460	754
596	833	19	210
74	105	332	128
0	537	657	544
827	82	192	679
454	357	150	704
929	530	90	758
402	462	492	208
330	95	590	170
928	98	444	273
873	751	273	674

If you want to select to the end of the current row of cells you can't use the Current Region property.

Instead use the **End(xlToRight)** method. This is like pressing CTRL + right arrow in Excel.

To move right from the active cell to the last cell in a row of cells:

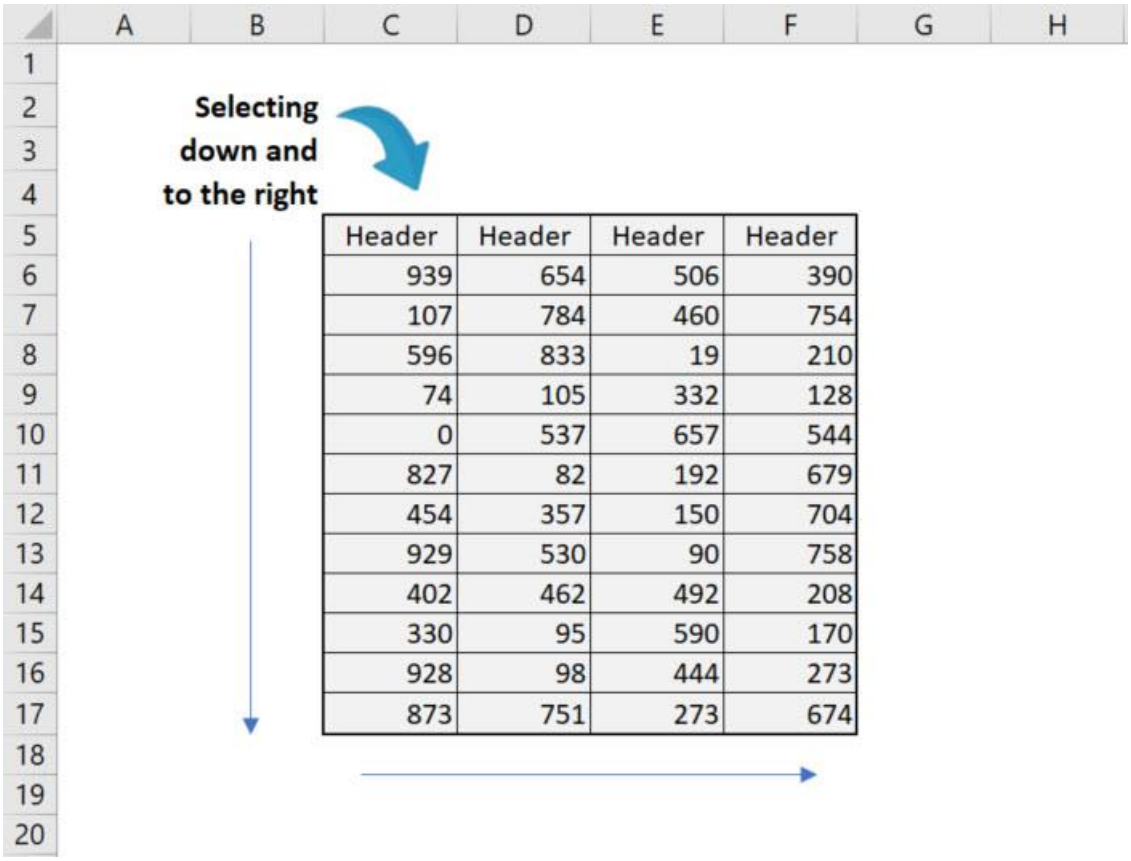
```
ActiveCell.End(xlToRight).Select
```

To select a range combine Range with the **End** method. This is like pressing CTRL + SHIFT + right arrow in Excel.

To see what I mean when I say you combine Range with the End method, here is code that selects right from the active cell:

```
Cells(1, 1).Select
Range(ActiveCell, ActiveCell.End(xlToRight)).Select
```

More Selections using "End"



Header	Header	Header	Header
939	654	506	390
107	784	460	754
596	833	19	210
74	105	332	128
0	537	657	544
827	82	192	679
454	357	150	704
929	530	90	758
402	462	492	208
330	95	590	170
928	98	444	273
873	751	273	674

You can combine more than one selection movement.

Here is an example that selects down, then selects to the right:

```
Range(ActiveCell, ActiveCell.End(xlDown).End(xlToRight)).Select
```

You have four options when using the End method:

- `xlUp`
- `xlDown`
- `xlToLeft`
- `xlToRight`

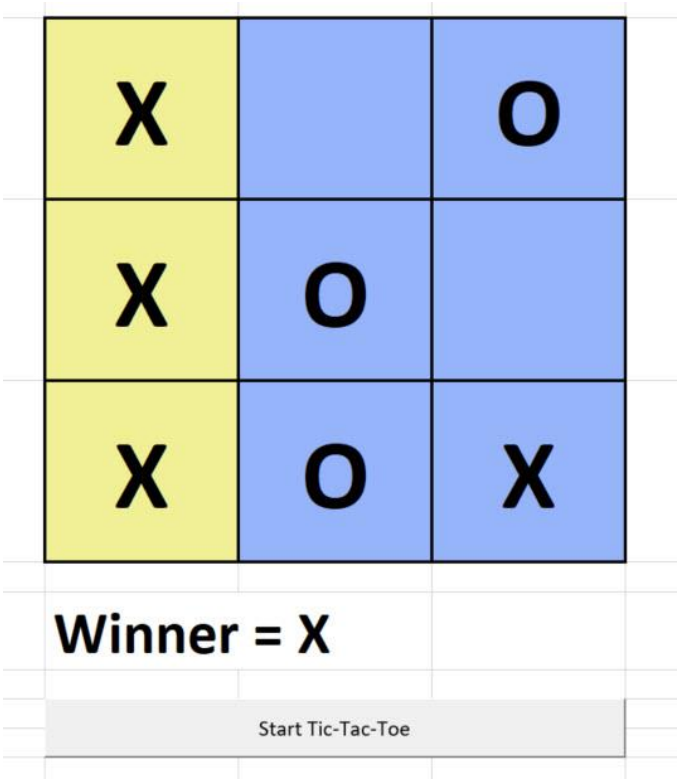
Mix and match these to your heart's content!

Hands On Fun with Projects

With Module 6 it is critical that you practice.

Being able to manipulate workbooks, worksheets and ranges is very useful. And it's a skill that only gets better when you practice.

To help you practice I created a number of projects. They range from simple to challenging:



I advise you to try the projects after you watch the video lessons. And use this handout as a handy reference guide for specific code.

Remember – practice lots and have fun!

Most good programmers do programming not because they expect to get paid or get adulation by the public, but because it is fun to program.

Linus Torvalds
