**Introduction**

A macro is a series of instructions which can be issued using a single command. The macro can be invoked in various different ways - from the keyboard (using a Control key combination), from a special icon on a toolbar or through the menu system. Ideally, you need to know a programming language (Microsoft Visual Basic) to create really useful macros; the examples below give you an introduction to macro writing.

**Absolute and Relative References**

Whenever you write a macro you have the option of using absolute or relative cell referencing. If you want to perform instructions on specific cells (eg move to cell A14, calculate the sum of the values above and format the result to appear as a currency in bold), then you use an *absolute reference*. If you want to move to a cell a certain number of rows/columns away from your present position (wherever that may be) then you use a *relative reference*. Hopefully the examples below will make this clear. First, open up the example file and set it up ready for use - users off campus can download the file by clicking on the filename in Step 1, below:

1. Load up Excel then press `<Ctrl o>` to [Open] the file called `phoenix.xlsx` in the D:\Training folder
2. Next, move to the File tab then click on Options
3. Click on **Customize Ribbon** and show the **Developer** tab in the **Main Tabs** – you’ll need the buttons on this tab later (though there is also a [Macros] button on the far right of the **View** tab on the **Ribbon**)

**Using Absolute References**

In this first example you are going to write a macro that adds up the values in columns A to E and applies a bold format to the totals.

1. Move to the **Developer** tab and click on the [Record Macro] button – a dialog box appears:

![Record Macro dialog box](image)

2. Set the **Macro name**: as **Sum** and the **Shortcut key**: to **Ctrl + S** (ie **<Shift> + s**)

   Note that case matters - avoid existing key combinations: **<Ctrl s>** is currently set to the **Save** command

3. **Store macro in**: **This Workbook** - the alternative is to store it in your **Personal Macro Workbook**, which would make the macro available every time you used Excel

4. Amend the **Description**: if you like (use this to say what the macro does & who recorded it)

5. Press **<Enter>** or click on [OK] to start recording

You now need to carry out the keystrokes required:

6. Press **<Ctrl Home>** (or click on A1) to make sure the macro starts in cell A1

7. Hold down **<Ctrl>** and press **<down arrow>** to move the active cell to the end of column A

8. Release **<Ctrl>** then press **<down arrow>** once more to move to cell A52

9. Press **<Alt =>** (or click on the [Sum] button on the **Home** tab) then press **<Ctrl Enter>** to confirm the formula =SUM(A2:A51) and stay in cell A52

10. Next, press **<Ctrl b>** (or click on the [Bold] button) to embolden the result

11. Using the cell handle, drag the formula in A52 across to E52 to get the totals for all the columns

12. Click on the **Developer** tab then on [Stop Recording] to end the macro

You should now have a macro which sums up the values in column A. Test out the macro as follows:

13. First, press **<Delete>** to empty the cells A52 to E52

14. Hold down the **<Ctrl>** and **<Shift>** keys together and type the letter **s** - the sums appear again in bold characters in cells A52 to E52

To demonstrate what is meant by **Absolute Referencing**:

15. Select rows 51 and 52 then press **<Ctrl c>** for [Copy]

16. Right click on cell A54 and under **Paste Options**... choose **Values** (the second icon shown as 123)

17. Select rows 51 and 52 again and **<Delete>** the contents – click on any cell to release the selection

18. Run the macro again (**<Ctrl Shift s>**) - you’ll find that the totals appear in the same cells based on the same data range (including the empty row 51), with the new total differing from that in row 55
19. Press <Delete> to remove the incorrect values
20. Select row 54 and drag the border of the selection to row 51 to reset the data for sample 50

This is a clear example of absolute references - the macro only works on the cells for which it was recorded. Even though a control key combination was used to move to the foot of the column, the next move (to cell A52 in step 9) was recorded as a move to a fixed cell.

In this second example, absolute references are left turned on but this does not affect the result. The object of this macro is to search for Blue-Green eggs and highlight the cells in an appropriate colour.

1. Press <Ctrl Home> to move to cell A1
2. On the Developer tab click on the [Record Macro] button
3. Set the Macro name: as Colour and the Shortcut key: to Ctrl+ C (ie <Shift> + c - <Ctrl c> is for copy)
4. Store Macro in: This Workbook and amend the Description: if you want
5. Press <Enter> or click on [OK] to start recording
6. Press <Ctrl f> (or move to the Home tab and click on [Find & Replace] then choose Find…)
7. In the Find what: box type Blue-Green
8. Press <Enter> or click on [Find Next]
9. Press <Esc> or click on the [Close] button to close the Find window
10. Right click on the cell found then use the list arrow attached to the [Fill Colour] button
11. Choose a suitable colour from the palette - eg Green
12. Click on the [Stop Recording] button on the Developer tab to end the macro

You should now have a macro which colours Blue-Green cells a suitable colour. Test out the macro by pressing the appropriate key combination:

13. Hold down the <Ctrl> and <Shift> keys and type the letter c - repeat this a couple of times

The macro works from the active cell downwards, moving to the next blue-green cell rather than to a fixed cell. To demonstrate this further:

14. Move the active cell down several rows
15. Hold down <Ctrl> and <Shift> and type the letter c

You will see later how to edit the macro to colour all the Blue-Green cells, including those which were skipped.

Using Relative References

You saw earlier how the Sum macro failed to work properly because it was using absolute references. Let’s repeat that here, to correct it:

1. First, click in cell A1 to make it the active cell then click on [Record Macro] on the Developer tab
2. Set the Macro name: as NewSum and the Shortcut key: to Ctrl+ t (this isn’t currently used)
3. Store Macro in: This Workbook and amend the Description: if you want
4. Press <Enter> or click on [OK] to start recording
5. Click on the [Use Relative References] button below [Stop Recording] on the Developer tab
6. Hold down <Ctrl> and press <up arrow> to make sure the active cell is at the top of the column
7. Next, hold down <Ctrl> and press <down arrow> to move the active cell to the end of the column
8. Release <Ctrl> then press <down arrow> once more to move to cell A52
9. Press <Alt =>> (or click on the [Sum] button) then press <Ctrl Enter> to confirm the formula
   =SUM(A2:A51) and stay in cell A52
10. Click on the [Stop Recording] button on the Developer tab to end the macro
11. Now, press <Ctrl Home> then move to cell D1 and press <Ctrl t> to run the macro again – it should work for column D (or, indeed, any column)

12. Press <Ctrl t> again for a second total (including the first)

You’ll find the new total isn’t what you might expect (it isn’t quite double the original total). If you look at the formula on the Formula Bar, you’ll see it says SUM(A3:A52) – it’s still summing 50 values, not the 51 you now have. You’ll see how to correct this later (but it needs a little programming).

13. <Delete> the value in cell D53

14. Click on the [Use Relative References] button to turn it off

This next example is designed to copy every fifth row of data to a new sheet (Sheet1). Here, both absolute and relative references are used. Relative references are required because each successive source and target area is a known distance apart (5 rows down and 1 row down, respectively):

1. Click on the [Insert Worksheet] button to the right of the PHOENIX tab to create a new Sheet1
2. On the Developer tab and click on [Record Macro]
3. Set the Macro name: as Every5 and the Shortcut key: to Ctrl+ e (this isn’t currently used)
4. Store Macro in: This Workbook and amend the Description: if you want
5. Press <Enter> or click on [OK] to start recording

Take great care to get the following instructions correct - if you make a mistake it’s probably best to abandon the recording and start again (you may have to use a different shortcut key).

6. First, check relative references are off (the button should not be highlighted in orange-brown)
7. Click on the PHOENIX tab then press <Ctrl Home> to make A1 the active cell
8. Now click on [Use Relative References] to turn relative references on
9. Hold down <Ctrl> and <Shift> and press the <right arrow> key to select row 1
10. Press <Ctrl c> (or right click and choose Copy)
11. Press <Ctrl Page Down> (or click on the Sheet1 tab) to move to the empty sheet
12. Press <Ctrl Home> to make sure you start in cell A1 then press <Enter> for Paste
13. Press the <down arrow> key to move the active cell on Sheet1 to A2
14. Press <Ctrl Page Up> (or click on the PHOENIX tab) to move back to the original data
15. Press the <down arrow> key FIVE times to move the active cell to A6 (or just click on it)
16. Hold down <Ctrl> and <Shift> and press the <right arrow> key to select the row
17. Press <Ctrl c> (or right click and choose Copy)
18. Press <Ctrl Page Down> (or click on the Sheet1 tab) then press <Enter> for Paste
19. Click on [Stop Recording] to end the macro and turn off [Use Relative References]

Test out the macro, if you like (clear the data in the cells on Sheet1 then press <Ctrl e>). All that happens is that the same data is pasted into the same cells on Sheet1. To complete the macro you now need to edit it.

**Editing a Macro**

Though recording macros is relatively simple, changing them is by no means easy. When you edit a macro you are taken into the Visual Basic Editor, and you need to understand exactly what the programming code is doing. Even if you can’t program you can make minor changes and, if you add the lines of programming detailed below, turn simple recorded macros into very useful ones.

1. On the Developer tab, click on [Macros]
2. Select Every5 then click on [Edit]

You are now placed in the Visual Basic Editor, with the following code displayed:
Sub Every5()
    ' Every5 Macro
    ' if you have added a description it appears here
    ' Keyboard Shortcut: Ctrl+e
    Sheets("PHOENIX").Select
    Range("A1").Select
    Range(Selection, Selection.End(xlToRight)).Select
    Selection.Copy
    ActiveSheet.Next.Select  ' or Sheets("Sheet1").Select if you clicked on it
    Range("A1").Select
    ActiveSheet.Paste
    Application.CutCopyMode = False
    ActiveCell.Offset(1, 0).Range("A1").Select
    ActiveSheet.Previous.Select  ' or Sheets("PHOENIX").Select, as above
    ActiveCell.Offset(5, 0).Range("A1").Select
    Range(Selection, Selection.End(xlToRight)).Select
    Selection.Copy
    ActiveSheet.Next.Select  ' or Sheets("Sheet1").Select, as above
    ActiveSheet.Paste
    Application.CutCopyMode = False
End Sub

Don't worry if you don't understand what appears on the screen. If you compare it carefully with the original instructions then you might get some idea of what is happening. Note the line which says:

    ActiveCell.Offset(1,0).Range("A1").Select

This is a relative reference - ie move the active cell down 1 row and across 0 columns from the active cell. Had you been using absolute references this line would have just read

    Range("A2").Select  - ie move to and select cell A2. You might like to scroll up to see the code recorded in the other macros.

**Tip:** You can use <Alt F11> to open up the Editor direct from Excel (or switch between it and Excel). Once you have finished making your changes, <Alt q> can be used to close down the Editor.

### Repeating an Action

Now try making some changes to the macro. The original objective of the macro was to select EVERY fifth row - the instructions which have been recorded need to be repeated 10 times (as there are 50 rows in the data set). To do this, you need to add extra lines to the middle and end of the program:

1. Move the typing position to the start of the code to be repeated - `ActiveCell.Offset(1,0)` ...
2. Add the following line of code `for k=1 to 10` then press <Enter>
3. Click at the end of the last-but-one line - `Application.CutCopyMode = False`
4. Press <Enter> and type the following line of code: `next k`

These extra two lines of code create a loop that is repeated 10 times. The loop counter, k, starts at 1 and increases by 1 each time the code is run. When it gets to 10, the loop ends.

5. Close the Editor window by clicking on the [Close] window button (or press <Alt q>) - the changes to the code are saved automatically
6. Select and <Delete> the values stored on Sheet1 then press <Ctrl e> to run the macro

You should find that 10 rows (every 5th one) are copied across to Sheet1.
7. See if you can perform a similar edit on the Colour macro to colour all the Blue-Green eggs.
Note that you will need to increase the final value of the loop to at least 21 (as there are 21 blue-green eggs).

Using Variables

The counter, k, in the example above is known as a variable in programming. It varies between a starting value of 1 up to an ending value of 10. You can use this variable in your module code:

1. On the Developer tab, click on [Macros] then select Every5 and [Edit] it.
2. Change the line reading ActiveCell.Offset(5, 0).Range("A1").Select to
   ActiveCell.Offset(k, 0).Range("A1").Select - ie change 5 to k.
3. Close the MS Visual Basic window by clicking on the close window button - or press <Alt q>.
4. Select and <Delete> the values stored on Sheet1 then press <Ctrl e> to run the macro.

You will find records 1, 3, 6, 10, 15, 21, 28, 36 and 45 are copied across - the offset (k) increases by 1 each time. When k=10, an empty record - sample number 55 in row 56 - is copied across.

For this final example, you need to understand a little more about programming. The instruction ActiveCell.Offset has numeric parameters - real numbers like 1, 5 or 0 or numeric variables like k. On the other hand, the instruction Range ("A1") has a string parameter, A1 (here enclosed in quotation marks). If you want to use the numeric variable k within the Range instruction, you must first convert it to a string of characters.

5. On the Developer tab, click on [Macros].
6. Select Every5 then click on [Edit].
7. Click at the end of the line for k=1 to 10 and press <Enter>.
8. Type in an extra line reading m$ = "A" & CStr(k).

The Cstr function converts the number stored in k into a string. CInt converts a string to a whole number.

9. Change the line reading ActiveCell.Offset(1, 0).Range("A1").Select to
   ActiveCell.Offset(1, 0).Range(m$).Select - ie change "A1" to m$.
10. Close the Editor window by clicking on the [Close] window button - or press <Alt q>.
11. Select and <Delete> the values stored on Sheet1 then press <Ctrl e> to run the macro.

This time the macro copies the same rows from phoenix as before, but pastes them in the corresponding rows on Sheet1. This is because the paste is carried out not 1 cell down from the active cell (in column A) but m$ (ie k) cells down.

Tip: You can easily supply the value for a parameter either by picking it up from a cell in the spreadsheet or via a dialog box. The Visual Basic you need for these are:

From a selected cell: k = Selection
From a dialog box: k = InputBox("Message giving instructions")

Obviously there's a lot more to macro writing than this simple introduction, but at least you should have some idea of how macros work and how you can change them.

Tip: If you click on the [Step Into] button instead of [Edit], you can use <F8> to step through the code one instruction at a time. You can even flip between the workbook and editor to see the effect of each line of code. This is invaluable in spotting where a macro is going wrong.

To correct the NewSum macro, you would need to make the following changes:

Add: Row = ActiveCell.Row - 1 after ActiveCell.Offset(1, 0).Range("A1").Select.
To work out how many rows there are, then edit the next line to:

Selection.FormulaR1C1 = "=SUM(R[-" & CStr(Row) & "]C:R[-1]C)"
Running a Macro from a Toolbar Button or Control

An alternative to running a macro using a control key combination would be to run it from a toolbar button.

1. Click on the down arrow on the right of the Quick Access Toolbar and choose More Commands…
2. Set Choose commands from: to Macros
3. Select any of your macros then click on [Add>>] to move them onto the toolbar
4. Next, click on [Modify…] and select a symbol for your button
5. Click on [OK] twice to close the dialog boxes
6. Finally, test out your new toolbar button – it should run the macro

To remove a button from the toolbar:
7. Repeat step 1 then select the button and [Remove] it – click on [OK] to close the dialog box

Controls

You can also run a macro from a control on a spreadsheet:

1. Click on the [Insert] button in the Controls group on the Developer tab
2. Choose [Button (Form Control)] – the first in this group
3. Click on the spreadsheet where you would like the control to appear – the Assign Macro dialog box appears
4. Select the required macro then press <Enter> for [OK]
5. Next, right click on the button and choose Edit Text
6. <Delete> the existing words then type in the name of the macro
7. Finally, click on the [Button] – the macro should run

Changing a Shortcut Key

If you want to change the shortcut key which runs a particular macro:

1. On the Developer tab, click on [Macros]
2. Select the macro whose shortcut key you wish to change then click on the [Options...] button
3. Set the new Shortcut key: then press <Enter> for [OK] and again to close the Macro dialog box

Deleting a Macro

To delete a macro you no longer want:

1. On the Developer tab, click on [Macros] then select the macro to be deleted and click on [Delete]
2. You are asked to confirm the deletion - press <Enter> for [Yes] (here press <Esc> or click on [No])
Saving a Macro

When you want to save a file with a macro in the new Excel format, you’ll find you have to save it with a special file extension (.xlsm) otherwise the macros won’t work:

1. Press <Ctrl s> or click on the [Save] button to save the file
2. Change Save as type: to Excel Macro-Enabled Workbook
3. Fill in the File name: then press <Enter> for [Save]

Setting Macro Security Levels

Because macros have been used to spread viruses between computers, Microsoft introduced extra security settings into Office. Currently, on the PCs in the IT Services labs, this is set to the lowest level in Excel, as we have extensive anti-virus security on our system. On your own PC, it’s advisable to set this to the medium level, giving you the opportunity to enable macros if you so choose.

To set the security level:

1. On the Developer tab, click on [Macro Security] - the following dialog box appears:

![Macro Security Dialog Box]

2. Select the level of security required (here, set it to Disable all macros with notification as above)
3. Press <Enter> or click on [OK]
4. Press <Ctrl F4> (or move to the File tab and choose Close) to close the file, leaving Excel running
5. Now click on the [Office Button] again and select phoenix.xls from the list of Recent Documents

With this security setting, whenever you open a file with a macro, it is automatically disabled and you have to take positive action to enable it. You should find a warning message immediately below the Ribbon:

![Security Warning]

6. Try running one of your macros – a warning message appears:

![Macro Error]

7. Click on [Enable Content] and try the macro again – it should now work
8. Finally, close the file again – there’s no need to save the changes (unless you want to)