Focused Seminar: Automating Tasks (Using Macros)

This 3 hour focused course will introduce the learner to running, creating, and editing macros in Excel and Word 2010. (Macros are a set of commands that can be triggered by clicking a button or using a keyboard shortcut instead of multiple clicks for repetitive tasks).

The first thing we need to do is to show the Developer tab in our Ribbon. To do this: Click on File, then Options. Choose Customize Ribbon. In the right-side column, put a check mark in the box in front of 'Developer'. Then click OK. There should be a Developer tab on the end of the Ribbon now (after the View tab). (NOTE: The Macros button can also be found on the View Tab, so you do NOT have to show the Developer tab if you choose not to)

You can also access the macro button from the view tab, but in order to get to the Visual Basic Editor, you must have the Developer tab enabled. (Either that, or add the Visual Basic button to the Quick Access Bar or to a custom group on the Ribbon).

When you open a file that contains macros, you will get a security warning. This is set to protect you from unknowingly downloading a virus that may be embedded in macros. (So only use files with macros if you know where they've originated). You will get a security warning above the formula bar that tells you that Macros have been disabled. Click the 'Enable Content' button in order to enable the macros in the workbook.

If you would like to disable the notifications (not recommended), go to the File tab, then Options, then Trust Center, then over on the right, click on Trust Center Settings and choose Macro Settings.

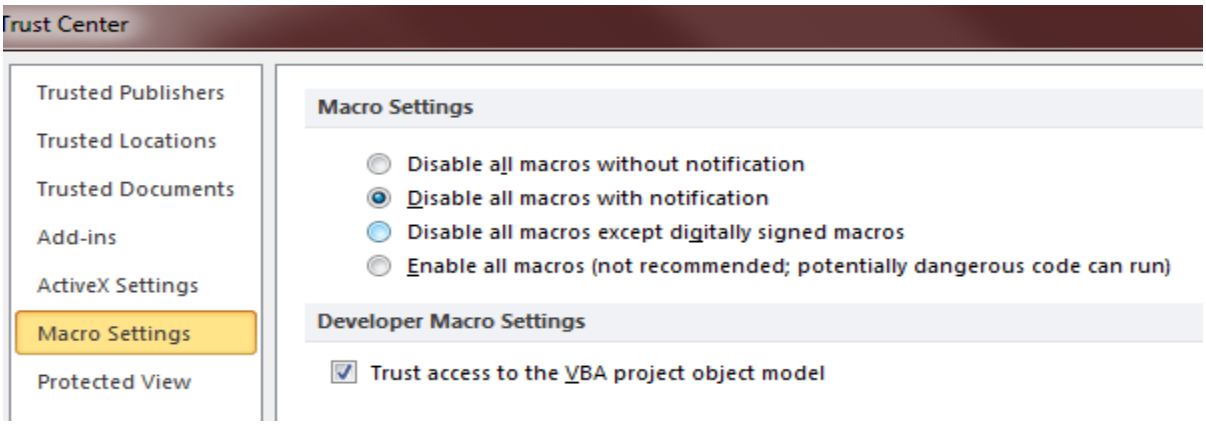

Figure 1: Trust Center Settings

In the newer versions of Excel (2007 & 2010) the file must be saved as a macro enabled file in order for macros to work. To save the file as macro-enabled, go to File, Save As. Choose Excel (or Word, etc) macro-enabled file from the Save As Type drop-down menu.

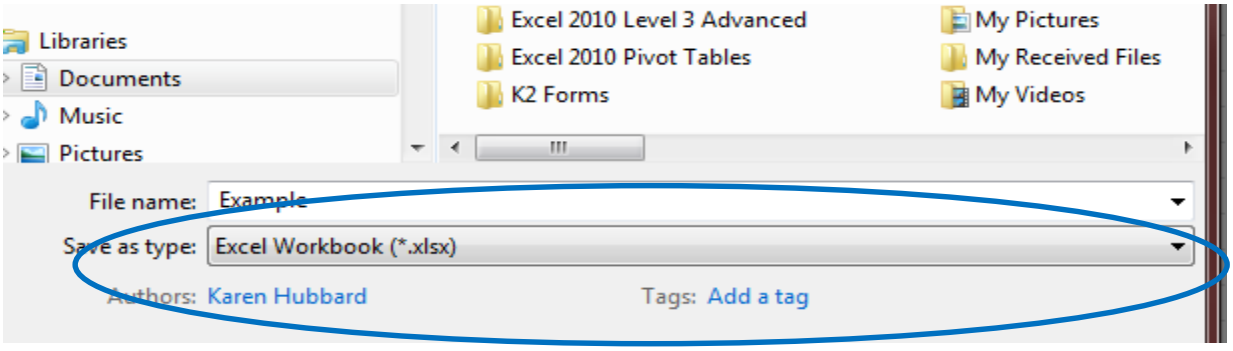

Figure 2: Save As Dialog Box

Running macros

You can run a macro by going to the developer tab and using the Macros button, using a keyboard shortcut (if one was assigned at the time of creating the macro), or adding a button to the Ribbon or Quick Access Bar.

If the macro won't run, check to make sure the file has been saved as macro-enabled. This is especially important if you have recently switched from Excel 2003 to Excel 2007 or Excel 2010.

Once a macro has been run, it cannot be undone (in other words, you can't Undo if you run a macro). You might be able to close the file without saving and start over, but you'd have to re-run the macro if you do that.

Practice activity 1:

1. Open RunningMacros.xlsm
2. Select cells A4:D4
3. Click on Macros on the Developer Tab.
4. Choose 'Column_titles' (if it's not already selected)
5. Click Run
6. Leave the file open

Practice Activity 2:

1. Select cell E4
2. Press Ctrl+Shift+C
3. Select cell E5
4. Run the Monthly_deduction macro (from the Macros dialog box)
5. Select E6
6. Press Ctrl+Shift+M

Recording Macros

You can record macros using absolute and/or relative cell references. We will work with both in the practice activities. A macro that uses absolute cell references means that the macro will run in the same cells every time you run it. (Ex. If you choose to delete the contents of the selected cell, when you run the macro, it doesn't matter where your cursor is placed when you start. The contents of the selected cell are always deleted. Or you can record the macro to delete the contents of the same cell, regardless of where the cursor starts) Relative cell references affect cells relative to the current position of the cursor. (Ex. If cell A1 is the active cell and you use relative cell referencing to put y our name in cell B2, your name will be entered into the cell one row down and one column over the from active cell. So if you click in cell C4 then run the macro, your name will be entered into cell D5). This will be made clearer during the practice activity.

Practice activity 1: Recording a macro

1. Open RecordingMacros.xlsm
2. Make sure the Loan Details worksheet is open
3. Select cell E4
4. Go to the Developer tab, click on Record Macro
5. In the Record Macro dialog box, enter a name for your macro. (NOTE: Macro names cannot contain spaces)
6. Click in the Shortcut key box and press Shift, then C
7. Store the macro in This Workbook
8. Put an appropriate description in the Description box.
9. Click OK

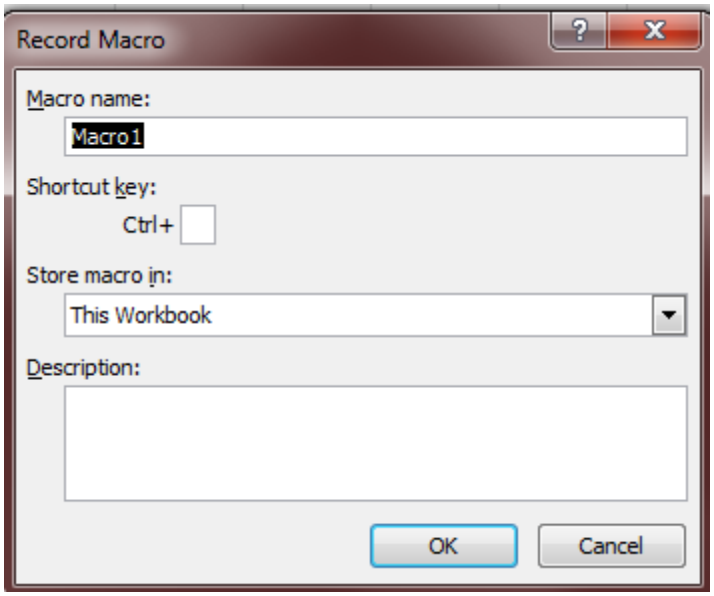

Figure 3: Record Macro Dialog Box

Now Excel will record all of your keystrokes/mouse gestures until you stop recording.

10. Go to the Home Tab
11. Select Wrap Text in the text alignment group.
12. Select Right align
13. Select bottom align (vertical alignment)
14. Select Bold from the font group.
15. Select a different color for the font from the Font group.
16. Down in the bottom left of the worksheet, in the status bar, click the stop recording button. (It's the small square button next to the word Ready)
17. Undo all of those actions and run the macro on cell E4 from the macro button on the developer tab.
18. Choose cell D4 and run the macro using the shortcut you assigned to it.
19. Select cells A4:C4 and run the macro again.

Because of the way we recorded the macro (we selected the cell before we recorded), the formatting changes are applied to the cells that we select every time we run the macro.

Practice activity 2: Recording a macro using absolute cell references

1. Switch to the Aircraft Sales worksheet
2. From the developer tab, click the Record Macro button
3. For the macro name, type in Absolute
4. Skip the other boxes and click OK
5. Select cell B4
6. Delete the current contents and type February.
7. Press Enter
8. Click the stop recording button
9. Delete the contents of cell B4 and select cell K20.
10. Run the Absolute macro.

In this example, because we selected the cell AFTER we began recording the macro, the actions will always be performed in cell B4, and not in the selected cells.

Practice activity 3: Recording a macro using relative cell references

1. Delete the contents of cell B4, then select cell A3.
2. Now click the Record Macro button.
3. Name the macro Relative (skip the other boxes) and click OK.
4. Turn on the Relative References button (right below the Record Macro button).

The relative references button is either toggled on or off. It doesn't matter if you click it first, then click Record Macro, or do it like we did here, after we clicked Record Macro. Either way, it's toggled on when we start recording the macro.

5. Click in cell B4 and type in February. Press enter.
6. Click the stop recording button. (You can also toggle the use relative references off if you'd like, or leave it on, doesn't matter)
7. Delete the contents of cell B4 and select cell A3.
8. Run the Relative macro.
9. Select cell G16.
10. Run the Relative macro.
11. Leave the file open

Notice that February is not entered into B4, it is now entered into H17 because the macro used relative cell references to enter the information into the cell that was one row below, and one column over from the original selected cell.

Assigning Macros to a Command Button

It is sometimes inconvenient (and/or difficult) to remember a shortcut key for one macro (especially if you have lots of them!) and constantly going to the developer tab to run them takes more time. You can always assign a macro to a command button that you can put on the Ribbon or the Quick Access bar to make it easier to access.

Practice Activity 1: Assigning a macro to a command button-Quick Access Bar

1. Leave the ReadingMacros.xlsm file open (or re-open it if you closed it)
2. On the Quick Access Bar, click the More button and choose More commands.
3. Choose Macros from the drop-down menu (instead of Popular commands)
4. From the list, choose Column_Titles and push Add
5. Click OK.
6. Observe the Quick Access Bar now has a macro icon
7. Click in cell A5.
8. Click the new command button.
9. The macro has now been applied to the selected cell with the push of one button.

Practice Activity 2: Assigning a macro to a command button-Ribbon

1. Go to the Developer tab and right-click in an empty area. Choose to Customize the Ribbon
2. Click on Developer Tab in the right side box.
3. Click on New Group at the bottom
4. Choose an icon (the Macro one is top left, I think)
5. Name the Group something you recognize (My Macros is always nice)
6. Change the Popular commands to Macros and select Column_Titles
7. Verify that the new group you just created is selected, then click Add.
8. Click OK
9. Select cell A6
10. Click the Column_Titles button on the Developer tab to run the macro.

Practice Activity 3: Inserting a macro button into the worksheet

1. From the developer tab, go to the Controls group and click Insert.
2. Choose Form controls/Button (first one-top row, left)
3. Draw the button in an empty area of the worksheet (you can change the size later if you need)
4. Click on the Column_Titles macro to put it in the Macro name box.
5. Click OK
6. Right click on the button and choose Edit Text
7. Change the text and click on a blank spot on the spreadsheet
8. Right click the button and choose Format Control
9. Check out the 7 tabs available for you to format the look of the button
10. Click OK
11. Select a cell in column A with text in it.

Saving Recorded Macros – Options

When you save a macro, you have a few options.

1. Personal Macro Workbook – This is a hidden workbook (Personal.xlsb) that is stored with the owner/user of the computer/Excel software. A macro stored here will run in any workbook and does not have to be recorded for each new/existing file.
2. This workbook – stored only for use in one particular workbook
3. New workbook – runs in all new workbooks

Storing the macros in your personal workbook is probably best if you switch between laptop/desktop or use a certain macro over and over again in all of your files. Macros saved in anything but the personal macro workbook are not available to use unless the specific workbook is open.

Recording and Running Macros in MS Word

There are some very slight differences from Excel when running or recording macros in Word, but the basics are the same.

Practice Activity 1: Recording & running a macro in Word

1. Open WordMacros.docx
2. Go to the End of the document (Ctrl+End)
3. Place the insertion point here at the end of the document
4. Go to the View Tab and click on the Macros button
5. Choose Record Macro
6. In the name box, enter Insert_Table (remember, macro names can't contain spaces)
7. In the Store in box, choose to store it in the WordMacros.docm folder
8. Click on the Button Icon (the keyboard with the hammer)
9. From the list, choose Project.NewMacros.Insert_Table
10. Click Add
11. Click OK
12. The cursor should now have a little cassette tape next to it.
13. Click the Insert tab
14. Choose Table
15. Create a Table of 3 columns and 2 rows
16. Make sure the insertion point is in the first cell of the table.
17. Press Shift + → three times to select the first row
18. Press Ctrl+B to bold the text in the first row
19. Stop recording (Go to View/Macros/Stop recording or click the stop recording button on the status bar)
20. Save the document. (You'll have to do a file, Save As, and select macro-enabled document from the list)
21. Test the macro by putting your insertion point anywhere in the document and running the macro. Check the bolding by typing text into the first row.

Viewing and Editing Macros

To edit or view a macro you must use the Visual Basic Editor. You can view the Visual Basic Editor (VBE) by going to the Developer Tab and clicking on the Visual Basic icon. Visual Basic is based on the concept called Object Oriented Programming (OOP). In Object oriented, programmers define not only the data type of a data structure, but also the types of operations (functions) that can be applied to the data structure. In this way, the data structure becomes an object that includes both data and methods that can modify the data. For example, an employee application might use an Employee object to capture basic pieces of data about each employee -- their age, their Social Security number, their job title, and so on. Possible methods in the Employee object could be ChangeWage or PrintAddress methods.

VBA Terminology

Term	Description
Comment	A line of text within a procedure, which you us to describe the line of code or the entire procedure. Comments always start with an apostrophe. (See the Property example)
Event	An event is when a user does something (like click the mouse), and Visual Basic reacts. How VB reacts can be controlled by *event procedures*. Event procedures are codes that VB will run when an event takes place.
Keywords	Special VBA terms that appear in blue in the code window.

Method	An action that's performed by an object. (Ex. Might be to Calculate or maybe we want to empty the cup to fill it with something else.)
	
	`Cup.EmptyCup`
	`Cup.FillCup`

Module	A file containing code or information you add to your project.
Object	Any element of an application with specific characteristics and behavior. It's a component that combines code and data.
Operators	Used as they are in formulas in a worksheet (+, -, *, /)
Procedure	Procedures are sequences of instructions that perform specific tasks. (Ex. You can have a procedure to wash the cup before putting it away, or save changes before closing a workbook)
Procedure Call	A statement that calls a procedure from another procedure.

Property	A property describes an object—it's behavior and it's look Ex. Object is a cup. Properties would be its shape, size, color, ceramic or plastic, what it contains, etc) If we want to change an object's properties, we would do it like this:
	
	`Cup.color = Blue 'changes the cup color to blue`
	`Cup.Size = 24 oz 'changes the size to 24 oz`

Variables	Used to store values. You can use a variable to store the results of a formulas (for example).

VBA and Macros

When you create (record) and run a macro, Excel (or Word) use VBA to write the macro. The macro will perform a set of instructions, then stop. (This is why macros are useful for automating repetitive tasks). When you write a procedure in VBA code, the execution path can branch based on conditions. In other words, VBA code can evaluate conditions and make decisions based on those conditions, then change the flow of execution. Macros can follow one singular execution path.

The Visual Basic Editor

To access the Visual Basic Editor (VBE), Click on the Developer tab, then choose Visual Basic from the Code group.

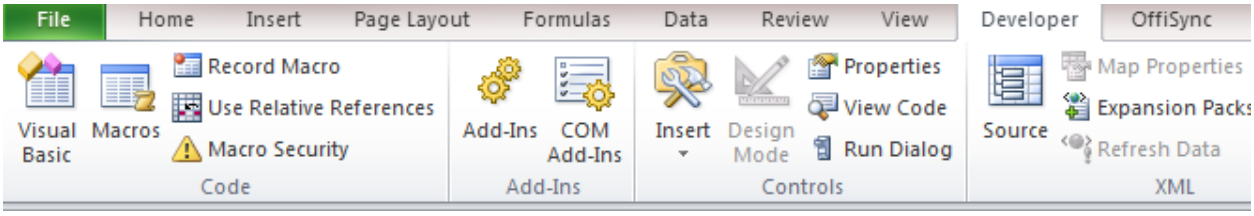

Figure 4: Developer Tab-Visual Basic

The Visual Basic Editor has three windows within the main window.

- Project explorer window – this window helps with navigation and management. It displays project for each workbook or template that's open. A project is a collection of modules. The name of the project corresponds to the saved workbook name. Each project contains folders for the objects in it. The Explorer also contains folders for other items, such as forms.
- Properties window – lists the properties of the selected object. You can use this window to change the object properties.
- Code window – This is where you enter, edit, and view the actual VBA code.

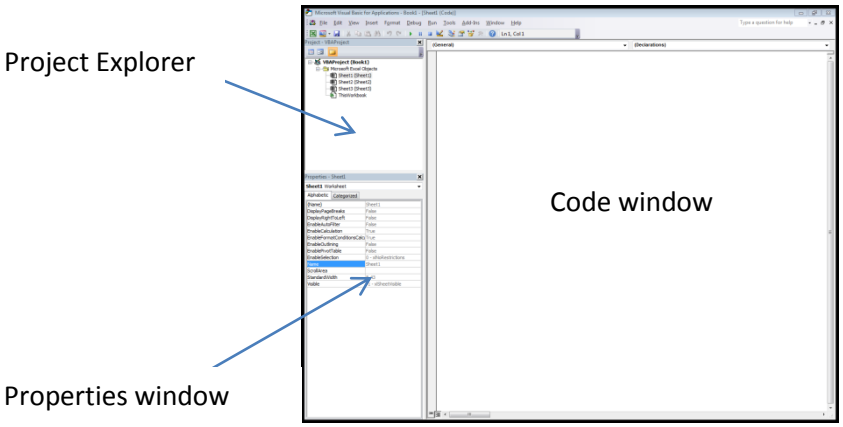

Figure 5: VBE Window (for blank Excel Workbook)

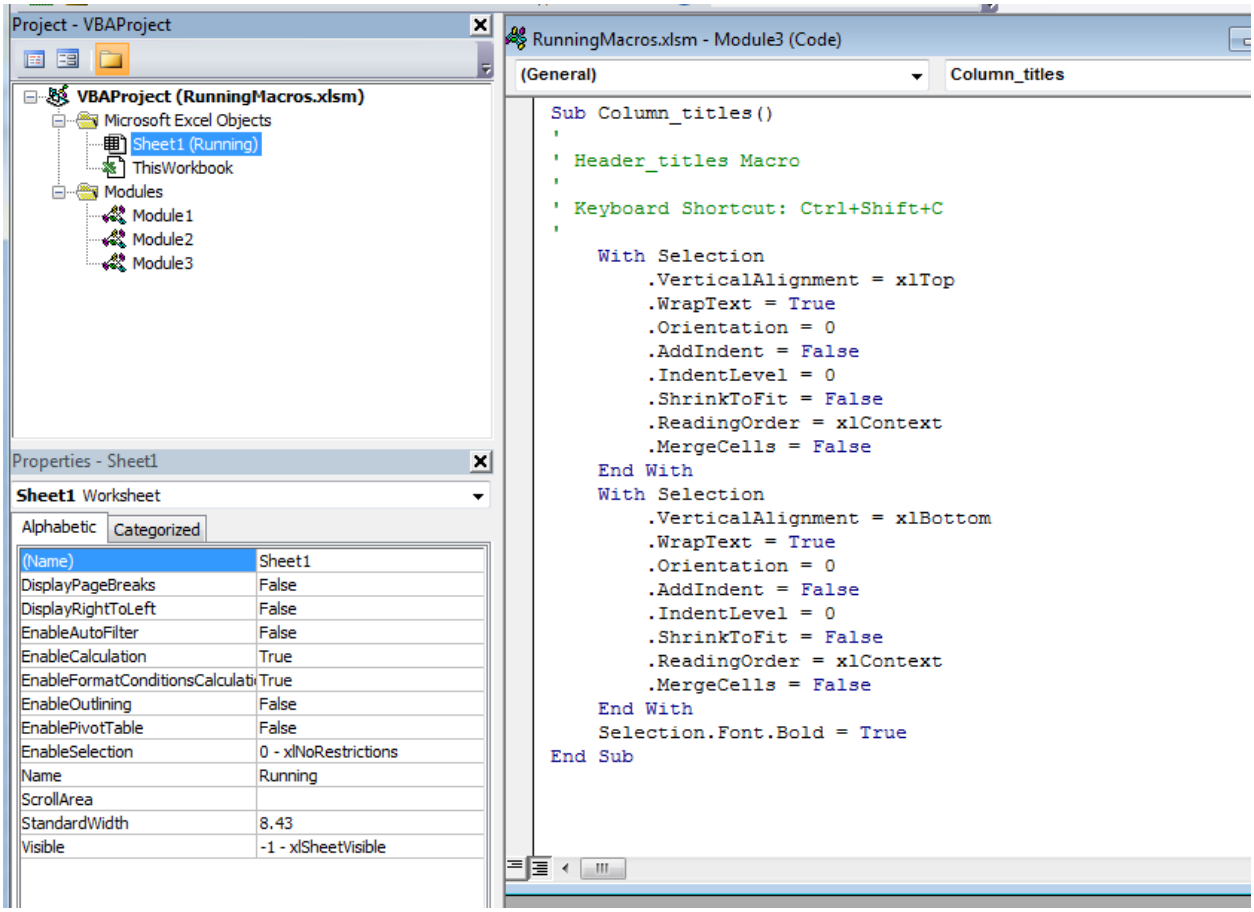

Figure 6: VBE Editor for RunningMacros Workbook

From the above figure, notice that Sheet 1 is selected in the Project Explorer window, so below, in the properties window, the properties of Sheet1 are visible. In the code window, we see the code for the column_titles macro that we wrote in an earlier exercise. If you click on Module 2 you will see the code for the payment calculation in the code window. Module 3 shows the column_title code. If the code is in a module, it is project-wide (in this case a workbook). If you write the procedures in the code window of an object (say Sheet1) the procedure will apply only to that object (Sheet1). So if you put a VB procedure into the Sheet1 object, Sheet2 cannot access or run that procedure.

Comments are in green, keyword commands are blue, and the rest of the code is black. To edit the code, put your cursor in the code window and type (or backspace) as needed – just like you would in a regular document.

When you create a macro in Excel or Word, the editor will put the tabbing and line breaks in as they appear above. When you are creating your own code, you can put line breaks in and indent lines of code as you wish/need. It's often easier to read the code when you indent certain lines or put line breaks between commands.

Practice Activity 1: Editing code in the Visual Basic Editor

1. Open RunningMacros.xlsm
2. Click on the Macros button (on the Developer tab or on the View tab)
3. Select the Column_Titles macro and click on Edit
4. When the Editor opens, click on different objects in the Explorer window and observe the property window changing.
5. Click over in the code window.
6. Go to the line that says: Selection.Font.Bold = True
7. Change Bold to Italic
8. Save the changes and go back to the workbook.
9. Run the macro.
10. Note that the titles are now italicized instead of being bolded.

Further learning on this topic:
K2 Technologies offers a six-hour class on VBA programming for Excel or Word. More advanced programming and editing techniques are offered during those classes.