VBA1-Introduction to Macros

Designed for those who are new to macro programming, this session covers recording macros, using the Relative References option and storing macros in the Personal Macro Workbook. The session introduces the VBA Editor and the VBA programming language so you can modify your recorded macros.

Settings to Start

Developer Tab

1. Turn on Developer Tab

Move Selection

1. Select File | Options | Advanced.
2. After pressing Enter, move selection is first item.
3. Turn Off.

Quick Access Toolbar

1. Select dropdown arrow on the right of the QAT.
2. Move QAT below the Ribbon.

EXERCISE: Change Options

- Review Absolute and Relative

Workbook and Personal Macro Workbook

1. Macros can be stored in the current workbook.
2. The file must have an .XLSM extension.
3. The macros are only available in that workbook.
4. Macros stored in the Personal Macro Workbook are available whenever Excel is loaded.
5. Personal Macro Workbook is a file in Excel Startup folder. Can be copied and shared.
6. Personal Macro Workbook is created the first time a macro is recorded in the Personal Macro Workbook.

XLSX and XLSM Files

1. Normal Excel files are saved as .XLSX files. (.XLS in pre-2007). They cannot store macros.
2. Macro files end in .XLSM. You can only save a file with macros in an .XLSM file.
3. Make sure File Extensions are visible on your computer.

Relative vs. Absolute Macro Recording

1. Macros recorded as Absolute references, i.e., select cell D4 and the macro will always select D4. Like B4/B7.
2. Use Relative References to record using a Relative reference, i.e., the macro will move so many rows and so many columns from the current cell. Like B4/B7.

Macro Names

1. No spaces. Use underscore (_) if you want a space.
2. Use Mixed case. MyMacro is easier to read than mymacro or MYMACRO.
3. Begin with a letter.
4. No reserved names (such as Cut, Copy or Paste).
5. Macros are called Subroutines in VBA. Use interchangeably.

Recording a Macro

1. Select Developer
2. Choose Record Macro
    a. Name the macro
    b. Select location
    c. Shortcut key (optional)
    d. Select Relative vs. Absolute reference
3. Excel will record all keystrokes and most mouse operations and convert to VBA statements.
4. When you are finished, click Stop Recording.

EXERCISE: Record a Macro

1. Start in B4.
2. Record a macro: AbsoluteRef
3. Move down two, right two. (End in D6).

Running a Macro

1. On the Developer tab, click Macros
2. Select your macro from the list.
3. Filter the list by file: all open files, Personal Macro Workbook, this workbook, or current workbook.
4. Store in Personal Macro Workbook.
5. The macro will run.

EXERCISE: Running a Macro

1. Move to H7.
2. Run AbsoluteRef.

EXERCISE: Recording and Running a Second Macro

1. Start in B4.
2. Record a macro: RelativeRef
3. Record in Personal Macro Workbook.
4. Turn on Use Relative References.
5. Move down 4 and right four.
6. Run RelativeRef.
7. You should be in F8.
8. Run RelativeRef again. You should be in J12.

Adding a Macro to the Quick Access Toolbar

1. Click dropdown arrow to the right of the Quick Access Toolbar.
2. Select More Commands from the list.
3. Select File | Options | Advanced.
4. Select Macros from the Choose Commands dropdown.
5. Select your macro and click Add.
6. Click Modify at bottom of dialog.
7. Change the icon.
8. Change the display name to something less techie.
9. Click OK until you are back in Excel.

EXERCISE: Add Macros to the Toolbar

1. Add AbsoluteRef.
2. Change icon used to Green square.
3. Change display name to Absolute Macro.
4. Add RelativeRef.
5. Change icon used to Blue square.
6. Change display name to Relative Macro.
7. Save toolbar.
8. Test each icon.
9. Create a new file.
10. Test each icon.

Saving the Personal Macro Workbook

1. When you exit Excel, you are asked whether to save the Personal Macro Workbook. Say Yes to save it, No to abandon your changes.
2. You can also save the Personal Macro Workbook from the Visual Basic Editor. You'll see that later.

EXERCISE: Exit Excel and Run it Again

1. Exit Excel. Save the Personal Macro Workbook.
2. Run Excel.
3. Test your macros.

VBA Editor

1. The Visual Basic Editor
    a. Developer | Macros | Visual Basic
    b. ALT+F11
2. Elements of the Editor
    a. Use View if they are not visible.
3. Project Explorer
    a. Lists all open files as VBAProjects.
    b. Each Project lists all Sheets, ThisWorkbook, Macro Modules, Forms, and Class Modules
    c. Macros can be embedded on Sheets and ThisWorkbook but Modules are much better.
    d. Macros can be run automatically using the built-in Auto_Open and Auto_Close functions. Microsoft does not publicize these because of viruses.
    e. Add a Procedure, Module, Form, or Class Module using Insert.
    f. Macros are called Subroutines or Procedures in VBA. You'll see each macro listed as Sub

```
MacroName()
       [code]
End Sub
```

4. Properties Window
    a. Shows the properties of the selected object. Use to rename a Module or work with a form. More on this later.
5. Code Window
    a. Shows your VBA code. Best to maximize this window.
6. Immediate Window
    a. Displays the results of a line of VBA. Useful for testing.
7. Locals Window
    a. While a subroutine is running, it shows you the value of all variables in the code. More on this later.
8. Watch Window
    a. Set a "watch" on a variable so that when it reaches a certain value, the code stops and displays variable values in the Watch window.
9. The Debug toolbar
    a. View | Toolbars | Debug
    b. Right-click on Standard toolbar, select Debug
    c. Dock to the right of the Standard toolbar.

   d. Icons of note:
      i. Triangle = Run
      ii. Double bars = Break (Pause running)
      iii. Square = Stop subroutine
      iv. Others to be explained later.
   e. Your key tool in debugging and testing macros.

10. Intellisense
    a. Keywords shown in blue.
    b. Comments (text that does not execute) shown in green.
    c. VBA code in black.
    d. Errors in red after pressing Enter.
       i. Use Tools | Options | Editor |Auto Syntax check to disable warning dialogs.
    e. Pressing dot (.) after keyword will display a list of methods and properties. (i.e. available options).

Introduction to VBA

1. The key to understanding macros and programming languages is action:
   a. If you want to set a value based on a test, use an Excel function: =IF(A1=4,27,0)
   b. If you want to perform an action, such as formatting, deleting, changing cells, printing, opening/closing, etc. you need a macro.
2. Macros are created using a programming language. They are programs.
3. Visual Basic for Applications (VBA) is a programming language based on Visual Basic (VB).
4. VBis a standalone programming language. You can create programs that run as EXE files.
5. VBA is tied to Microsoft Office. There is a version of VBA for Word, Excel, Access, PowerPoint, and Outlook. They use the application's "object libraries" which contain definitions of the application's components.
6. VBA combines programming actions with the components of the selected application. This allows you to control the elements of the application using programming code.
   a. You can automate tasks, create forms, and more using VBA.
   b. VBA code is always tied to the application, either in an "application-wide" file (Personal Macro Workbook or NORMAL.DOTM) or the specific file.
   c. They cannot execute as separate EXE files.
7. Like most programming languages, VBA has the following:
   a. Variables – containers for specific values. Usually "typed" i.e., limited to a specific type of data (integer, string, etc.)
   b. Statements—combinations of keywords (programming commands) and specifiers that perform some operation.
   c. Flow of Control—versions of the IF THEN ELSE statement that lets the program execute different instructions based on conditions (If variable = value, If the cell is blank, etc.)
   d. Loops—commands that let the program repeat an action or series of actions until some condition is reached. (**`WHILE ActiveCell <> "" [code] WEND`**.
   e. Forms—dialog boxes that you can customize with text, buttons, dropdown lists, etc.

8. Modern “object-oriented” programming languages use constructs called objects.
    a. A variable has a value.
    b. An object has properties and methods.
    c. Properties include a value, formatting, etc.
    d. Methods are actions that can affect the object (Close, Open, Print, Select, Delete, etc.)
    e. VBA is object oriented. Don’t get hung upon whether something is a property or a method. Just know that the thing does what you want it to.

HOMEWORK EXERCISE: Record several macros that perform tasks you often perform manually.