Short Courses in Microsoft Excel VBA
Course Outlines

Aimed at business users, these half-day training courses each focus on a specific area of expertise in Microsoft Office applications. They apply to all current versions of Microsoft Office and are suitable for users at all levels of experience in the programs. The following courses are currently available:

Recording Excel Macros

Writing Your First Macro

Repeating Code with Loops

Variables and Constants

Interacting with the User

Handling Errors

Building UserForms 1 – Simple Forms

Building UserForms 2 – Advanced Dialogs

Building Custom Functions

Creating Add-Ins

Automation - Working with Other Programs

Recording Excel Macros

All users can benefit from recorded macros in Excel regardless of their skill level, and they don't have to learn VBA programming to do so. A macro is simply a collection of commands written in the VBA programming language. Using Excel's Macro Recorder tool the user completes a task. As they do so Excel records and saves their commands. When the task has to be repeated the user runs the macro and Excel performs the sequence of commands by itself. Recorded macros can be edited and improved if required as the user's knowledge of VBA increases and they are an excellent introduction to Excel macro programming. Topic include:

- **What Recorded Macros Can and Can't Do:** As they stand, recorded macros can be very powerful but there are certain things that cannot be recorded and have to be added manually.
- **Naming a Macro:** There are rules and conventions to be observed when naming a macro. Giving a macro a sensible and meaningful name is an important part of the task.
- **Where to Store a Macro:** Excel stores its macros within the workbook itself so where you choose to keep your macros depends on how you intend to use them.
- **Recording Simple Macros:** Learn how to use the Macro Recorder by recording a few simple commands.
- **Recording Complex Macros:** How best to give a sequence of more detailed commands, and make use of the Relative References option.
- **Editing A Recorded Macro:** Examine the VBA code created by the Macro Recorder and make changes to streamline and improve it.
- **Macro Security:** The presence of macros in a workbook has an important implication on Security. Learn how to comply with the rules.

Writing Your First Macro

This course introduces the Visual Basic Editor, a component part of all the principal Microsoft Office programs, and where macros can be created and edited. Learn the basic of writing VBA code and see how the Editor can offer help and check your code for errors as you create it. See how recorded macros can be transformed with a little extra work and create new macros from scratch. The course includes:

- **Naming a Macro:** All macros need a concise, meaningful name and there are rules and naming conventions to be observed.
- **Navigating the Program:** How to address and refer to the various parts of an Excel workbook. Learn how to "talk to" Excel.
- **Giving Commands:** Let the Visual Basic Editor help you create commands in the correct way so that Excel understands what you want it to do.

- **Create Branching Code:** Create decision-making code with the aid of If Statements and Case Statements.
- **Add Comments:** Comments are used to annotate and add notes to macro code and are a vital tool to help other users (and to remind you) what a macro or a particular code statement does. They can also be used to temporarily disable code during testing.
- **Testing and Debugging a Macro:** Before using and distributing macros they should be thoroughly checked and tested. The Visual Basic Editor offers several tools to help with this task.

Repeating Code with Loops

Perhaps the most significant thing you can do with VBA that cannot be recorded is to automatically repeat a code sequence. The Macro Recorder can record a specific task but what if that task needs to be repeated multiple times on different cells or worksheets? In VBA there are several different ways to repeat or "loop" code either a specific number of times or as many times as necessary, allowing you to create incredibly powerful macros.

- **What Loops are For:** When and why you might to repeat a code sequence and how to write code that is suitable for use in a loop.
- **Looping through collections:** Excel's Object Model defines a number of collections of objects (worksheets in a workbook, cells in a range etc.). Using a loop lets you apply a macro to each member of a specified collection.
- **Choosing the Best Loop for the Job:** As you can see there are several kinds of loops. Learn how to choose the right sort of loop for the job in hand.
- **The Do/Loop Loop:** This is the simplest kind of loop. It will repeat its commands indefinitely unless it receives an instruction to stop.
- **The Do Until/Loop Until Loop:** This loop can be instructed to continue until a specified condition is met.
- **Do While/Loop While Loop:** This loop can be instructed to continue as long as a specified condition prevails.
- **For/Next Loop:** Usually combined with a counter, this loop is used when the number of iterations can be specified (i.e. go round a specific number of times).
- **For Each/Next Loop:** A special kind of loop for working through collections of objects such as each worksheet in a workbook, or each cell in a selection.
- **Nested Loops:** A common requirement is to place one loop inside another (e.g. to work on each worksheet in each open workbook). When used like this the loops are said to be "nested".

Variables and Constants

Variables are an essential component of most macros. They are mostly used to temporarily store data during the course of a macro, but they can also be used to represent the members of a collection or as counters in code loops. This course includes:

- **Declaring Variables, How and Why:** Good practice dictates that all variables are "declared" before they are used. This process involves giving a name and description to the variable. Discover how to do this and the naming rules and conventions you should observe.
- **Choosing an Appropriate Data Type:** More good practice is to specify a data type for the variable so that Excel handles it efficiently.
- **Understanding and Controlling the Scope of a Variable:** The scope of a variable defines the limits of its use, whether global or limited to a specific macro.
- **Passing Values to and Reading from a Variable:** The value of a variable can be changed as often as required and may be read once or multiple times during the course of a macro.
- **Using a Variable as a Counter:** Some loops use a variable to keep count of how many iterations of the loop are made.
- **Working with Array Variables:** Regular variables hold only one piece of data at a time. Array variables can hold many values and can have one or multiple dimensions.
- **Using Constants:** Unlike a variable, a constant is assigned a value which never changes.

Interacting with the User

A useful aspect of VBA programming is the ability to interact with the user, either by displaying a message or by asking the user for a decision or to provide some information. The appropriate use of these tools can greatly improve the usability and flexibility of your macros. The tools commonly used are the Message Box and the Input Box. This course explores their use in detail.

- **Communicating Effectively with the User:** A simple message in Excel's Status Bar will let the user know that the macro is working and a Message Box can tell them when the macro has finished.
- **Simple Message Boxes:** Used to convey a piece of information, the simple "OK Only" message box can reassure the user and add branding to your macros,
- **Multiple-Choice Message Boxes:** When a macro cannot make a decision for itself it becomes necessary to ask the user. To facilitate this there is a wide selection of multi-choice message boxes such as "Yes/No", "Yes/No/Cancel" etc. Learn how to interpret and act on the user's choice.
- **Using Input Boxes:** When the macro needs to ask the user a question requiring a more detailed response, using an Input Box you can display a message that includes a textbox for the user to provide information that you can use in the macro.

Handling Errors

Even the most carefully planned and written macro can fail. Perhaps you overlooked something, or maybe the user did something you didn't anticipate. Whatever the cause, the macro crashed. The question is what happens next?

- **Excel's Built-in Error Routine:** If you do nothing Excel's own error handler takes over. Is it OK to depend on this?
- **Why Error Handlers are Necessary:** An error handler can help you take control and rescue the situation when something goes wrong.
- **When is it Safe to Ignore an Error:** Some errors aren't really a problem but what sort of error is it safe to ignore?
- **Creating an Effective Error Handler:** An error handler should prevent things getting worse, rescue the situation as far as possible, and let the user know what has happened.
- **Logging and Reporting Errors:** You can't always rely on the user to tell you what an error message said and what they were doing at the time. A simple error logging routine can automatically record the information you need to find out what went wrong and fix it.
- **How to Prevent Errors Occurring:** Perhaps the most important job is to anticipate any problems and prevent errors occurring in the first place.

Building UserForms 1 – Simple Forms

VBA UserForms provide the ultimate in user interaction. Using the Visual Basic Editor's graphical interface you can create dialog boxes that can be simple or as complex as you need, simply by dragging-and-dropping tools on to your form. Add the necessary code to drive the UserForm and you can create powerful tools for Excel. In this course you are introduced to the UserForm and learn to build and code simple forms and dialogs. Topics include:

- **Creating a UserForm:** How to create new UserForm and define its size, name and caption.
- **Working with the Toolbox:** Discover the wide range of controls (objects or tools) that can be placed on a UserForm.
- **Adding and Resizing Controls:** Place controls where you want them on the UserForm and change their size and position.
- **Defining a Control's Properties:** Manage the behaviour of each control by defining its properties.
- **Opening and Closing a UserForm:** Use VBA code to open and close a UserForm either automatically or upon the user's command.
- **Basic UserForm Coding:** Use event driven code to initialize a UserForm, power command buttons, read from and write to an Excel worksheet.
- **Building a Simple Dialog Box:** Create a simple and useful dialog box using the skills learned in this course.

Building UserForms 2 – Advanced Dialogs

Building on the skills learned in the first course, this course will help you create more sophisticated UserForms making use of some of the more complex tools that are available. The following topics are covered:

- **Adding and Coding Multi-Select List Boxes:** List Boxes are great tools for assisting and controlling user input but when the user is allowed to make multiple selections special coding is required to read and handle their choices.
- **Check Boxes and Option Buttons:** Check boxes normally work alone and option button work in groups. Learn how to use each type and write the code to control their behaviour.
- **Enabling and Disabling Controls:** Help guide the user by offering them only those controls that are relevant. For example an "OK" button can remain disabled until they have completed all the necessary tasks on the UserForm.
- **Working with MultiPage Forms:** The multi-page control is the ideal tool to display a large number of controls on a UserForm, or when it is required to separate controls with different purposes.
- **Building a Complex Dialog Box:** Create a wizard-type dialog box using the advanced skills learned in this course.

Building Custom Functions

Excel provides has a large collection of ready-made functions, more than enough to satisfy the average user. Many more can be added by installing the various add-ins that are available. Most calculations can be achieved with what is provided, but it isn't long before you find yourself wishing that there was a function that did a particular job, and you can't find anything suitable in the list. The solution is to build one yourself. Creating a UDF (User Defined Function) is easy and requires only basic VBA skills. In this course you will learn how to:

- **Create and Name a User-Defined Function:** As with Excel macros there are naming rules and conventions to observe.
- **Specifying Function Arguments:** Most functions require at least one, and sometimes several arguments (information that the user supplies). A name and data type must be specified for each one.
- **Working with Optional Arguments:** Sometimes one or more arguments are optional. If the user does not supply the information because it is not relevant or because they want the default value to apply. The UDF's coding needs to take account of this.
- **Saving and Distributing User-Defined Functions:** Like Excel macros, a UDF created in one workbook can be made available to other workbooks, and can be distributed to other users. Learn about the scope of UDFs and how best to share them with other users.

Creating Add-Ins

Having created useful VBA macros and custom functions you will probably want to distribute them to your colleagues and other users. The best way to do this is by creating an Excel Add-In. An Add-In is a special type of Excel file in which you can store macros and custom functions. When an Add-In is installed and activated Excel opens it each time the program is started, making its macros and functions available to the user. Functions will appear automatically in Excel's Paste Function tool and it is even possible to include VBA commands to create new menu and toolbar items (Excel 2003 and earlier) or new ribbon tabs and commands (Excel 2007 and later) to assist the user.

- **Assembling the Macros and Functions:** The first step is to bring all the macros and custom functions you want to include in the Add-In.
- **Securing an Add-In:** In addition to thoroughly testing its contents, you should lock an Add-In's code project and secure it with a password.
- **Saving and Documenting the Add-In:** In addition to its filename, the Add-in can be given a friendly descriptive name and a description of its contents.
- **Distributing Add-Ins:** Having built your Add-in you will need to know how to install and activate it.
- **Building an AddIn:** During the course you will build an Add-In containing some useful functions and macros.
- **Introduction to Ribbon Development:** Although worthy of at least one course itself, this topic will be explained and demonstrated to give you an idea of what is required.

Automation - Working with Other Programs

In VBA the term Automation refers to the process of communication between one program and another. It lets you control one program from another. You have all the programming tools available to you that you would have if you were working within the program itself. You could, for example create entries in your Outlook Calendar or create an Outlook Task using information from an Excel worksheet. You could create a Word document using information in an Excel spreadsheet then save and print it or attach it to an Outlook Email. The possibilities are endless. This course teaches you how Automation works using a range of useful examples.

- **Early Binding and Late Binding:** These are the two methods of establishing a connection to another program. Find out which is the best one to use for your project.
- **Setting a Reference to Another Application:** VBA uses Object Libraries to let one program understand the vocabulary of another. This is how it is done.
- **The Principles of Client Server Automation:** How to create a connection to another program and use VBA to send commands to it.
- **Other Automation Tools:** Do you want the user to see the other program? Should you hand over control of the other program to the user when your code finishes, or would you like to close it? These are some of the additional considerations when using Automation.