

Mr. Prakhar Mody

Class: MSc

Subject : Application of IT- Basics and Advance Excel

Chapter: Unit 2 Chapter 1

Chapter Name: Overview of VBA & Macros

What are Macros?

- *A Macro is a piece of programming code that runs in excel environment, and it helps to automate routine tasks. In other words, a macro is a recording of your regular steps in excel, which you can replay using a single button.*
- *When you create a macro, you are recording your mouse clicks and keystrokes. After you create a macro, you can edit it to make minor changes to the way it works. You can use a macro as many times as you want. It reduces the amount of time you will require to do a task.*
- *Suppose that every month, you create a report for your accounting manager. You want to format the names of the customers with overdue accounts in red, and also apply bold formatting. You can create and then run a macro that quickly applies these formatting changes to the cells you select.*

What is VBA?

- *Visual Basic for applications (VBA for short) is a programming environment designed to work with Microsoft's Office applications (Word, Excel, Access, and PowerPoint).*
- *Components in each application (for example, worksheets or documents) are exposed as objects to the programmer to use and manipulate to a desired end. Almost anything you can do through the normal use of the Office application can also be automated through programming.*
- *The main difference between VBA and Macro is that **VBA is the programming language to create Macros while Macros are programming codes** that run on Excel environment to perform automatic routine tasks.*

Why use VBA & Macros?

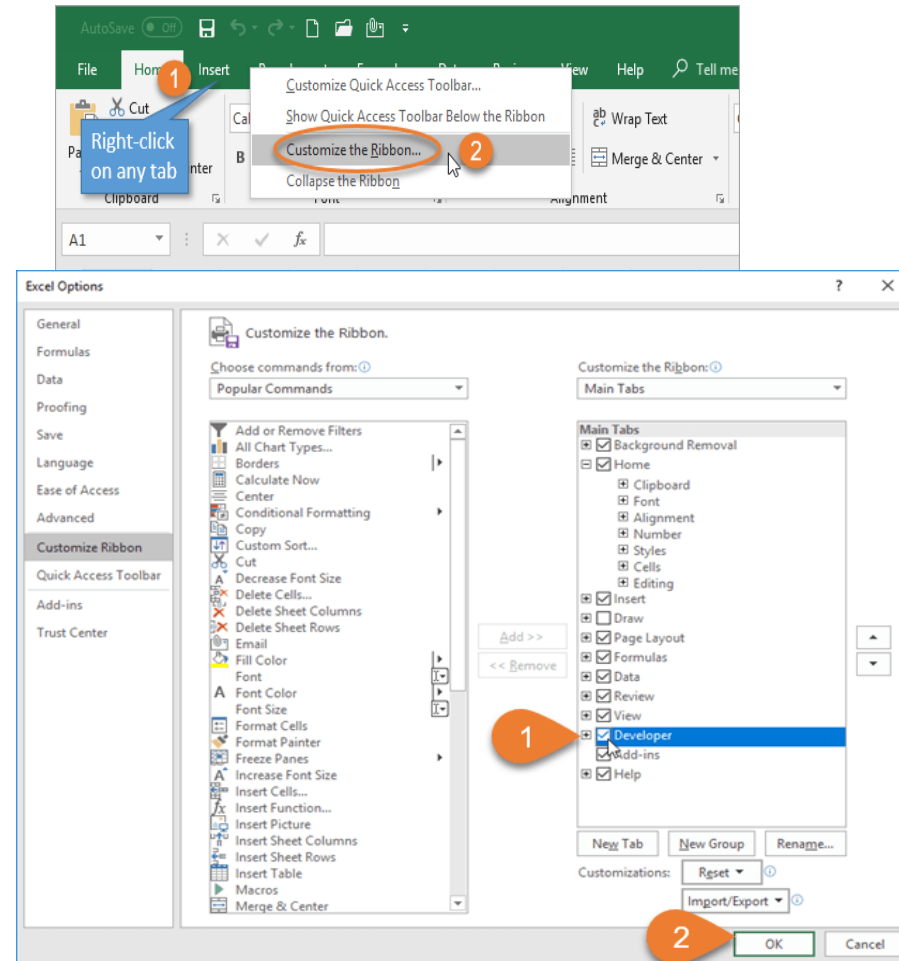
- **Automates repetitive and routine tasks:** By learning VBA, a process such as receiving emails in Outlook, generating and sending a responding email, processing data in Excel, and even copy and paste work becomes easy.
- **Accessibility to other users:** With VBA, other users do not have to install anything provided you write a script for everyone in the department. VBA can also allow you to add user-friendly variables that other users can modify to a certain degree. All in all, there is quick access to information from other users.
- **Reduces the turnaround time:** People working in the finance department are always under pressure to submit back their reports. It's usually a tedious task for them and which under stressful conditions may lead to inaccurate reports. VBA removes this burden and makes it easy to prepare reports and templates within a short time.

Getting Started

Macros and VBA tools can be found on the Developer tab, which is hidden by default, so the first step is to enable it.

To enable developer tab:

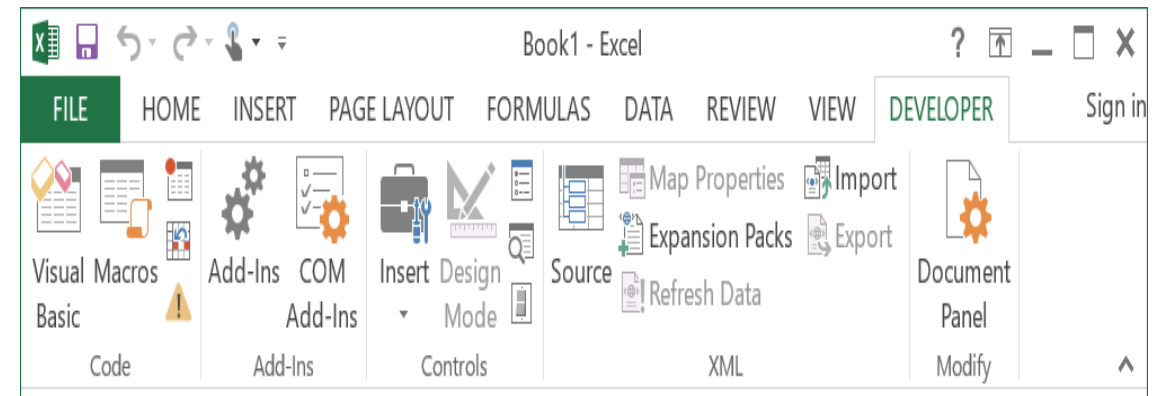
- 1. Right click on any existing tab on the ribbon*
- 2. Select Customize Ribbon.*
- 3. Under Customize the Ribbon and under Main Tabs, select the Developer check box.*



Developer Tab

The Developer tab is the place to go when you want to do or use the following:

- *Write macros.*
- *Run macros that you previously recorded.*
- *Use XML commands.*
- *Use ActiveX controls.*
- *Create applications to use with Microsoft Office programs.*
- *Use form controls in Microsoft Excel.*



My First Macro

There are two ways to create a macro

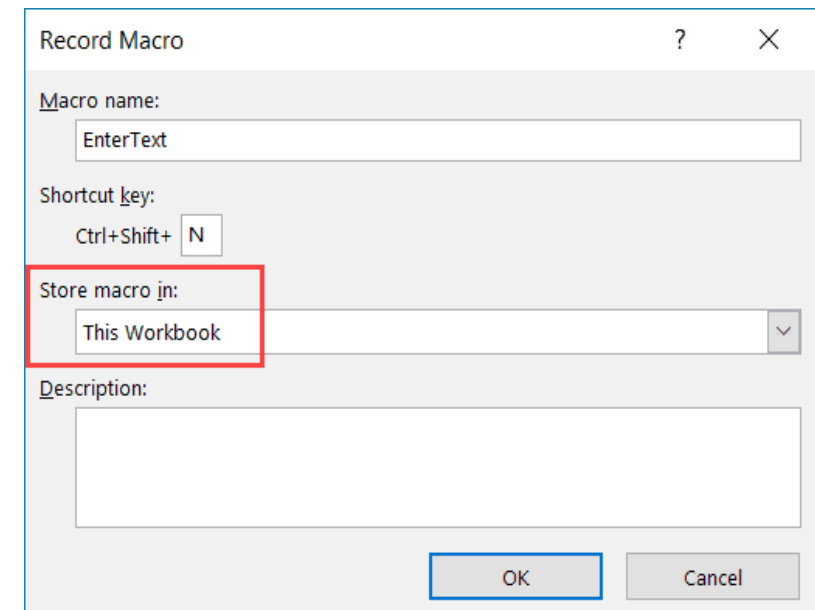
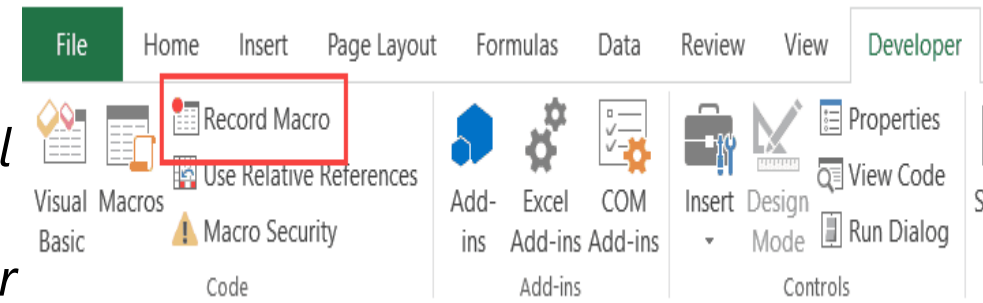
1. **By recording a macro**: *Recording a macro is the simplest way to create a macro. One doesn't need any prior knowledge of VBA or programming to use it. Once you start recording a macro, you only need to carry out your task, how you would've carried it out in general. Excel will translate each step that you do in a language that it understands and create a macro.*
2. **By writing your own code**: *To overcome limitations in recording a macro, one can create a macro by writing their own code.*

Recording a Macro

The steps to record a macro are as follows:

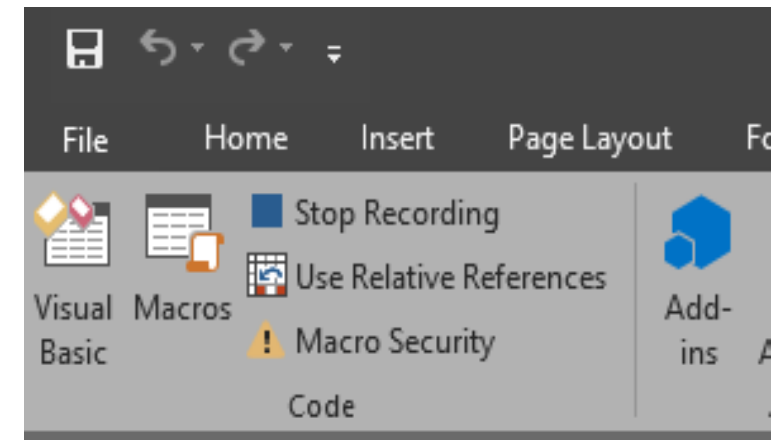
- 1. Click the Developer Tab*
- 2. In the Code group, click on the Macro button. This will open the 'Record Macro' dialog box.*
- 3. In the Record Macro dialog box, enter a name for your macro. I am using the name EnterText. Remember that you cannot use spaces in between.*
- 4. You can assign a keyboard shortcut if you want. In this case, we will use the shortcut Control + Shift + N. Remember that the shortcut you assign here would override any existing shortcuts in your workbook.*

Note: For example, if you assign the shortcut Control + S, you will not be able to use this for saving the workbook (instead, every time you use it, it will execute the macro)



Recording a Macro

6. *In the 'Store macro in' option, make sure 'This Workbook' is selected. This step ensures that the macro is a part of the workbook. It will be there when you save it and reopen again, or even if you share it with someone.*
7. *You can enter a description if you want.*
8. *Click OK. As soon as you click OK, it starts to record your actions in Excel. You can see the 'Stop recording' button in the Developer tab, which indicates that the macro recording is in progress.*



Recording a Macro

Once you have started recording a macro you can start carrying out your task and every single action will be recorded. For example:

- *Type your name in the active cell*
- *Move the cell pointer to the cell below and enter this formula: =NOW()*
- *Select the formula cell, and press Ctrl+C to copy that cell to the Clipboard.*
- *Choose Home ⇌ Clipboard ⇌ Paste ⇌ Values (V).*
- *With the date cell selected, press Shift+up arrow to select that cell and the one above it (which contains your name).*
- *Use the controls in the Home ⇌ Font group to change the formatting to Bold and make the font size 16 point.*
- *Choose Developer ⇌ Code ⇌ Stop Recording*

Running a Macro

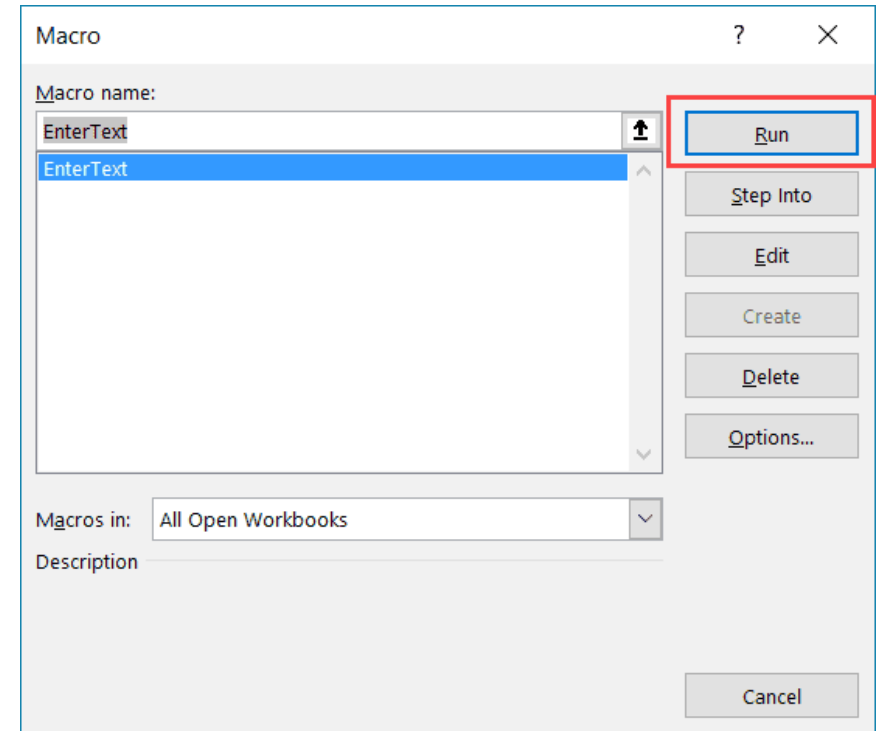
There are multiple ways to run a macro:

- *To run your macro, move to an empty cell and press Ctrl+Shift+N (or whatever shortcut you assigned)*

OR

- *Choose Developer ⇌ Code ⇌ Macros (or press Alt+F8) to display the Macros dialog box.*
- *Select the macro from the list (in this case, NameAndTime), and click Run.*

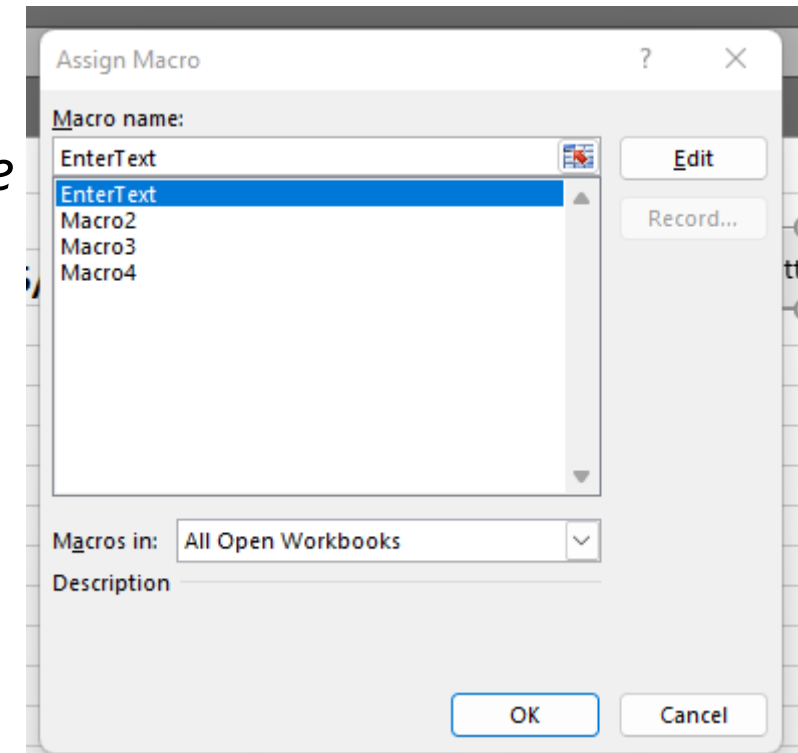
(Ensure that you have cleared what you have typed in the workbook to determine whether the macro is working correctly.)



Running Macros Through Button

4. Click anywhere on the worksheet. This will insert the button wherever you click and automatically open the 'Assign Macro' dialog box.
5. In the Assign Macro dialog box, you will see a list of all the macros that you have in the workbook
6. Click the Macro name that you want to assign to this button. In this example, I will click on the macro named 'EnterText'
7. Click on OK

Whenever you will press the button it will run that respective macro.



Absolute Vs Relative Reference

- *Just like absolute & relative reference in excel there is the same concept in macros too.*
- *If you use an absolute reference option to record a macro, the VBA code would always refer to the same cells that you used.*
- *For example, if you select cell A2, enter the text Excel and press Enter, every time – no matter where you are in the worksheet and no matter which cell is selected, your code would first select cell A2, enter the text Excel, and then move to cell A3.*
- *If you use a relative reference option to record a macro, VBA wouldn't hardcode the cell references. Rather, it would focus on the movement when compared with the active cell.*
- *For example, suppose you already have cell A1 selected, and you start recording the macro in the relative reference mode. Now you select cell A2, enter the text Excel, and hit the enter key. Now, when you run this macro, it will not go back to cell A2, instead, it will move relative to the active cell.*
- *You can switch on relative reference: Developer ⇨ Code ⇨ Use Relative Reference.*

What Recording a Macro does in the Backend?

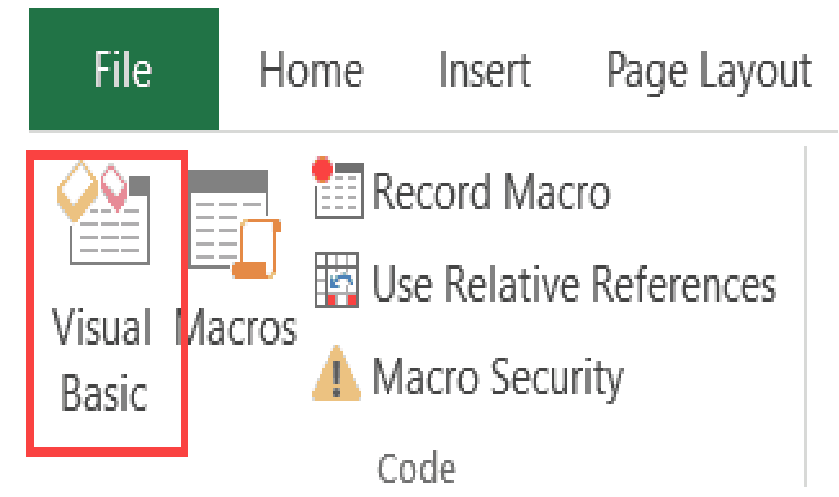
Now let's go to the Excel backend – the VB Editor – and see what recording a macro really does.

Here are the steps to open the VB Editor in Excel:

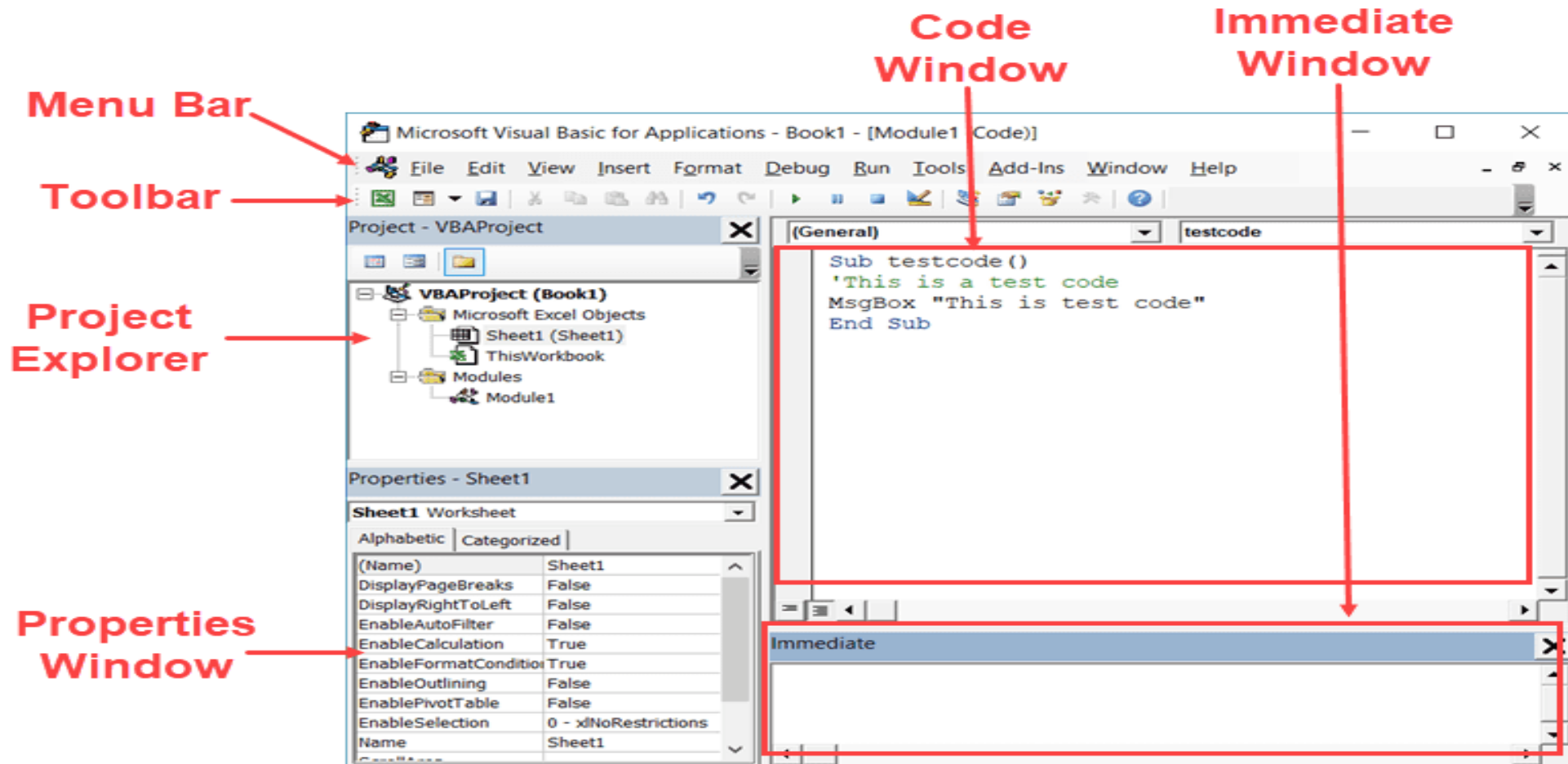
- 1. Click the Developer tab.*
- 2. In the Code group, click the Visual Basic button.*

OR

Use shortcut – ALT + F11 (hold the ALT key and press F11)



VBA Sheet Parts



VBA Sheet Parts

- *Menu Bar: This is where you have all the options of VB Editor. Consider this as the ribbon of VBA. It contains commands that you can use while working with the VB Editor.*
- *Toolbar – This is like the Quick Access Toolbar of the VB editor. It comes with some useful options, and you can add more options to it. Its benefit is that an option in the toolbar is just a click away.*
- *Project Explorer Window – This is where Excel lists all the workbooks and all the objects in each workbook. For example, if we have a workbook with 3 worksheets, it would show up in the Project Explorer. There are some additional objects here such as modules, user forms, and class modules.*
- *Code Window – This is where the VBA code is recorded or written. There is a code window for each object listed in the Project explorer – such as worksheets, workbooks, modules, etc. We will see later in this tutorial that the recorded macro goes into the code window of a module.*
- *Properties Window – You can see the properties of each object in this window. To show this, click the view tab and select Properties Window.*
- *Immediate Window –It's useful when you want to test some statements or while debugging. You can make it appear by clicking the View tab and selecting the Immediate Window option.*

What happened when we recorded a macro?

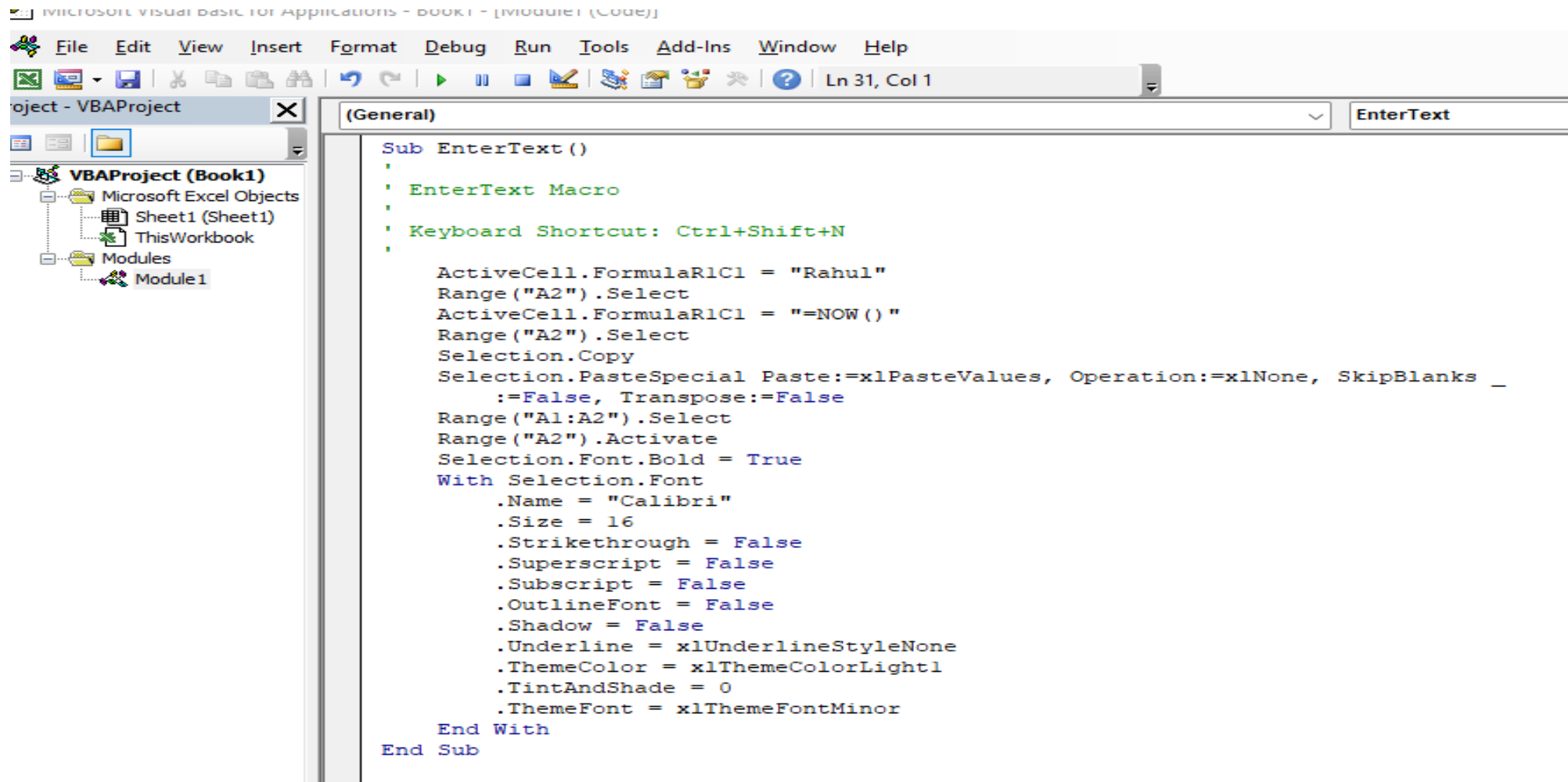
When we recorded the macro – EnterText, the following things happened in the VB Editor:

- 1. A new module was inserted.*
- 2. A macro was recorded with the name that we specified – EnterText*
- 3. The code was written in the code window of the module.*

So if you double-click on the module (Module 1 in this case), a code window as shown below would appear.

- You can edit the code as you want, maybe change the name or font.*
- When you rerun the macro, you will see that changes have replicated in the excel too.*

What happened when we recorded a macro?

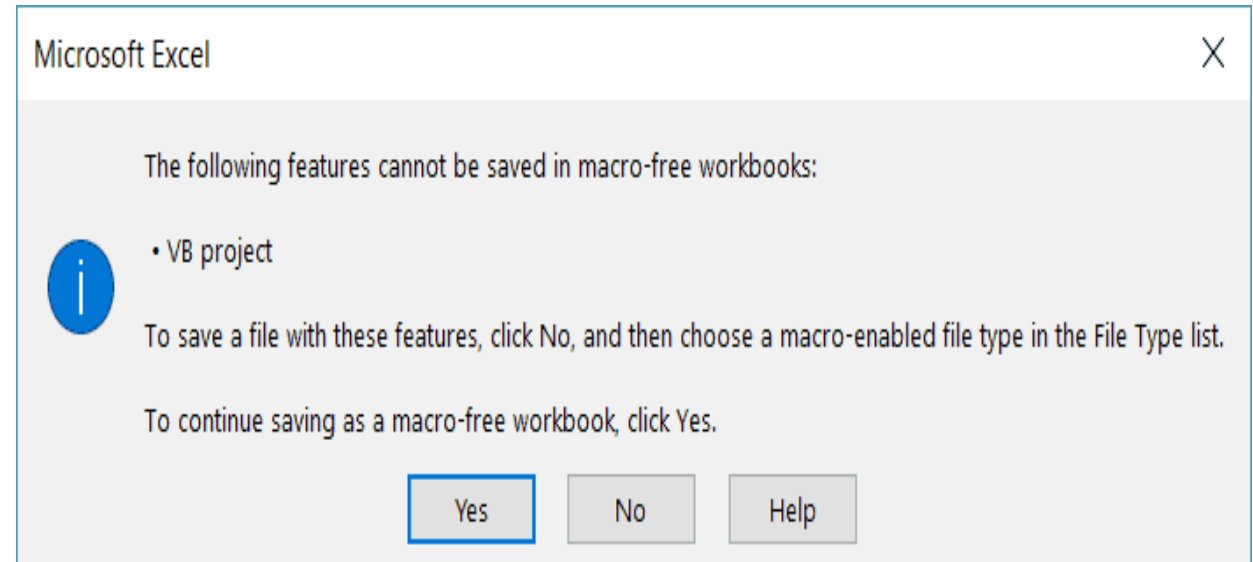


The screenshot displays the Microsoft Visual Basic for Applications (VBA) editor interface. The title bar reads "Microsoft Visual Basic for Applications - BOOK1 - [Module1 (Code)]". The menu bar includes File, Edit, View, Insert, Format, Debug, Run, Tools, Add-Ins, Window, and Help. The toolbar contains various icons for editing and running code. The left-hand Project Explorer shows the "VBAProject (Book1)" structure, including "Microsoft Excel Objects" (Sheet1 (Sheet1), ThisWorkbook) and "Modules" (Module1). The right-hand editor window is titled "(General)" and shows the code for a macro named "EnterText". The code performs the following actions: sets the active cell formula to "Rahul", selects cell A2, sets the formula to "=NOW()", copies the content, pastes it with values only, selects the range A1:A2, activates cell A2, and applies a bold Calibri font with size 16. The macro is assigned the keyboard shortcut Ctrl+Shift+N.

```
Sub EnterText ()  
    ' EnterText Macro  
    ' Keyboard Shortcut: Ctrl+Shift+N  
    '   
    ActiveCell.FormulaR1C1 = "Rahul"  
    Range("A2").Select  
    ActiveCell.FormulaR1C1 = "=NOW()"   
    Range("A2").Select  
    Selection.Copy  
    Selection.PasteSpecial Paste:=xlPasteValues, Operation:=xlNone, SkipBlanks _  
        :=False, Transpose:=False  
    Range("A1:A2").Select  
    Range("A2").Activate  
    Selection.Font.Bold = True  
    With Selection.Font  
        .Name = "Calibri"  
        .Size = 16  
        .Strikethrough = False  
        .Superscript = False  
        .Subscript = False  
        .OutlineFont = False  
        .Shadow = False  
        .Underline = xlUnderlineStyleNone  
        .ThemeColor = xlThemeColorLight1  
        .TintAndShade = 0  
        .ThemeFont = xlThemeFontMinor  
    End With  
End Sub
```

Saving Workbooks with Macro

- *If you store one or more macros in a workbook, the file must be saved as a macro-enabled file type.*
- *In other words, the file must be saved with an XLSM extension rather than the normal XLSX extension.*
- *While saving your file a dialog box will appear reminding you to save it as a macro enabled file.*



Limitations of Recording a Macro

Macro recorder is great at following you in Excel and recording your exact steps, but it may fail you when you need it to do more.

- *You can't execute a code without selecting the object.*
- *You can't create a custom function with a macro recorder.*
- *You can't run codes based on Events: In VBA you can use many events – such as opening a workbook, adding a worksheet, double-clicking on a cell, etc, to run a code associated with that event. You can use a macro recorder to do this.*
- *You can't create loops with a macro recorder*
- *You can't analyze conditions: You can check for conditions within the code using macro recorder. If you write a VBA code manually, you can use the IF Then Else statements to analyze a condition and run a code if true (or another code if false).*
- *You can't pass arguments in a macro procedure: When you record a macro, it will never have any arguments. A subroutine can take input arguments that can be used within the macro to perform a task.*