

VBA

What Is Visual Basic for Applications (VBA)?

Visual Basic for Applications (VBA) is part of Microsoft Corporation's legacy software Visual Basic. VBA is used to write programs for the Windows operating system and runs as an internal programming language in Microsoft Office (MS Office, Office) applications such as Access, Excel, PowerPoint, Publisher, Word, and Visio. VBA allows users to customize beyond what is normally available with MS Office host applications.

VBA is an event-driven tool, which means that you can use it to tell the computer to initiate an action or string of actions. To do this, you build custom macros—short for macroinstructions—by typing commands into an editing module.

A macro is essentially a sequence of characters whose input results in another sequence of characters (its output) that accomplishes specific computing tasks.

VBA allows users to manipulate graphical user interface (GUI) features such as toolbars and menus, dialogue boxes, and forms. You may use VBA to create user-defined functions (UDFs), access Windows (APIs), and automate specific computer processes and calculations.

KEY TAKEAWAYS

- Visual Basic for Applications is a computer programming language developed and owned by Microsoft.
- With VBA you can create macros to automate repetitive word- and data-processing functions, and generate custom forms, graphs, and reports.
- VBA functions within MS Office applications; it is not a stand-alone product.

How to Access VBA in Excel

To access VBA in Excel, simply press Alt + F11. Your existing Excel workbook will remain running, but a new window will appear for Microsoft Visual Basic for Applications.

When to Use VBA and Why?

Automation of Repetitive Tasks

Imagine a spreadsheet in Excel containing data on sales of different employees and you want to single out the ones who sold higher than a certain amount. A normal option would be to use the conditional formatting feature to highlight them and find out. Now imagine there are thirty of those spreadsheets in the workbook. Of course, you can follow the same procedure thirty times. Nevertheless, it would be boring and time-consuming. If we use VBA instead, we could just write the code that does the required action once in a module and run the code for as many numbers as possible.

This is one of the most important reasons people use VBA.

Extending Scopes of User Interaction

VBA provides some functionalities that normal Office applications do not offer through features. Keep in mind that there are so many features that are not showing on the ribbon that you still need to enable through settings. Still, it doesn't compare to the features VBA can deliver. And if you are in a scenario where you need to utilize those features, you might be in dire need to use VBA.

Interacting Between Office Applications

VBA makes moving information from one office application to another quite easy. For example, you might need to move your Excel data for a Powerpoint presentation or maybe in a report in Word. A simple copy and paste

sometimes don't cut it when you need to present it in a certain way or the process might be too lengthy. VBA offers ways to do such things with ease.

Creative Way of Doing Things

Sometimes using VBA is just fun. Building up logic to process information and complete actions is fun stuff for many people. And Experts tend to gravitate towards using it. One cause of it is repeatability of course. Although you may need time to adjust to the coding languages and environment, especially if you are new to coding and stuff, VBA provides fun ways to perform otherwise complex processes.

Common Uses of VBA

Some practical uses of VBA include but are not limited to:

Usage in Finance

Microsoft Excel in general covers a wide variety of tools to cover tasks in financial fields. VBA makes the job a lot easier. Common uses of VBA in finance include:

Analysing Large Amounts of Data: In the finance world, it is pretty common to store, analyse and evaluate huge amounts of data on a daily basis for decision-making purposes. Although Excel provides some benefits for that, it still cannot rival VBA.

Create/Maintain Complex Models: We can use Excel VBA to create trading and risk, management models. We can use these models to track, evaluate, and forecast trends of stock and find a suitable time to buy a particular one.

Creating Better Investment Options: Excel VBA offers more flexibility when it comes to analysing and forecasting. So, we can evaluate and predict any business model more closely and invest according to the results of the predicted return on investments. This removes the interference of human error and provides calculated risks which in turn help decision makers make logical decisions.

Usage in Marketing

VBA can widely be used in marketing sectors also. With the help of VBA, we can analyse marketing metrics for social media, collect data online, create interactive content calendars, create forecasting and graphic models, create invoices and develop charts that can help us promote a certain product or help strengthen the position of an organization.

Usage in Programming

VBA's usage in programming is limitless. VBA provides more flexibility than any of the Office applications can provide. It is not only limited to helping us store data more easily, analyse scientific data, and develop databases and charts from databases but also provides means of performing Excel or other Office actions and many more. One relevant example would be to create custom functions and define tasks that it would do when the function is called upon. That way we can make our custom functions to perform certain tasks that Excel, by default, does not provide.

Features of VBA

VBA in Office applications, especially Excel, contains the following features.

Macro: Macro has always been an important part of Excel VBA going back as far as Excel 2003. You can record a macro in Excel through the **Record Macro** option from the **Developer** tab. Or as experts tend to prefer, you can write your own lines of code. Both have their advantages and disadvantages.

Workbook and Worksheet Object: Excel VBA can contain objects. These objects can contain other objects which can have another within them. This can go on. In fact, the whole workbook itself is an object in Excel VBA that we consider has different objects within it. The workbook leads the hierarchy of the objects.

Range Objects: Range is a collection of cells in Excel. A range object is the same. Range objects can start from 2 cells to the spreadsheet's maximum capacity. It is the most important object in terms of VBA's usage. Different functions of range objects include copying, moving, naming these ranges, formatting, etc.

Collections: Excel functions objects can both be singular and plural forms. Such as a worksheet and worksheets, range and ranges, etc. The plural variants are collections.

Loop: Loops are the fundamental portion of Excel VBA that automates long and repetitive tasks. Loop, as the name suggests, creates a loop, and executes a single task multiple times for all object components or any set number of times described in a loop. There are three types of loops in Excel VBA- the For loop, the Do While loop, and the Do Until loop.

Array: Like other programming languages, Arrays in VBA indicate a structure that contains the same type of data in the same place. Arrays can both be one-dimensional and multi-dimensional.

String Manipulation: String manipulation means the conversion of other types of data into a string type to work with it in different scenarios. VBA's STR function does that. It is a built-in function and one of the popular ones.

Date/Time: The DATE function in Excel can help with the manipulation of date and time. It returns the current date system.

Properties and Methods: You can set properties for objects. It calls out their approaches. Setting a property changes the control of an object. Meanwhile, calling out means calling objects makes the object perform its task.

Procedure: This is the execution style of the code. VBA, like any other programming language, goes through line by line and executes each line of code one after the other. There are two types of procedures. One is sub procedure and the other is called a function.

Statement: A statement is an instruction or a set of instructions. We can break it down into two parts- a declaration or naming of variables, ranges, functions, etc., and an executable statement. An executable statement is the designated piece of code that defines what a particular action is.

Variables: Generally, by variable, we mean the storage location for a particular item. Excel can store ranges, particular values, cell values, and even worksheets as variables.

Logical Operators: Logical operators are of two types- TRUE and FALSE. Both indicate whether a condition is fulfilled or not in any scenario.

Where to Write Code in VBA?

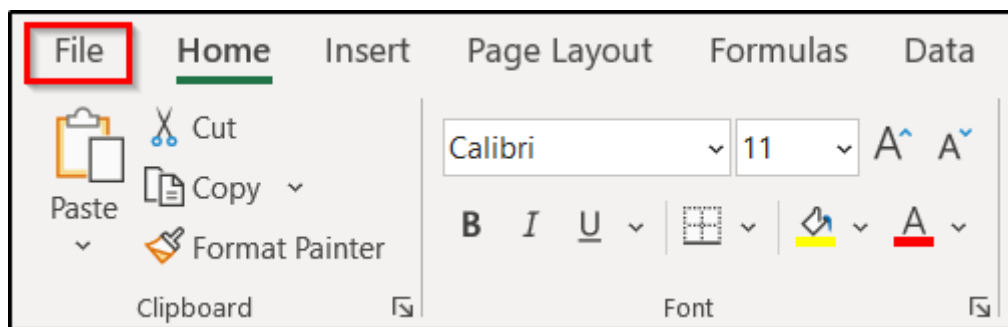
Enough with the comments of VBA. But how do you actually use it in office applications? Here is a simple introduction to the procedure of using VBA features and applications.

All of the Office applications have a **Developer** tab in their ribbon. You may find it hidden usually. You need to enable it through the settings option. We are more focused on the usage and application of Microsoft Excel here. But it is almost the same for any other Office application. Nevertheless, here is a demonstration of enabling the tab, using **modules** and running a sample code to show you how it works.

Enabling Developer Tab

If you do not have the **Developer** tab on your ribbon, follow these steps:

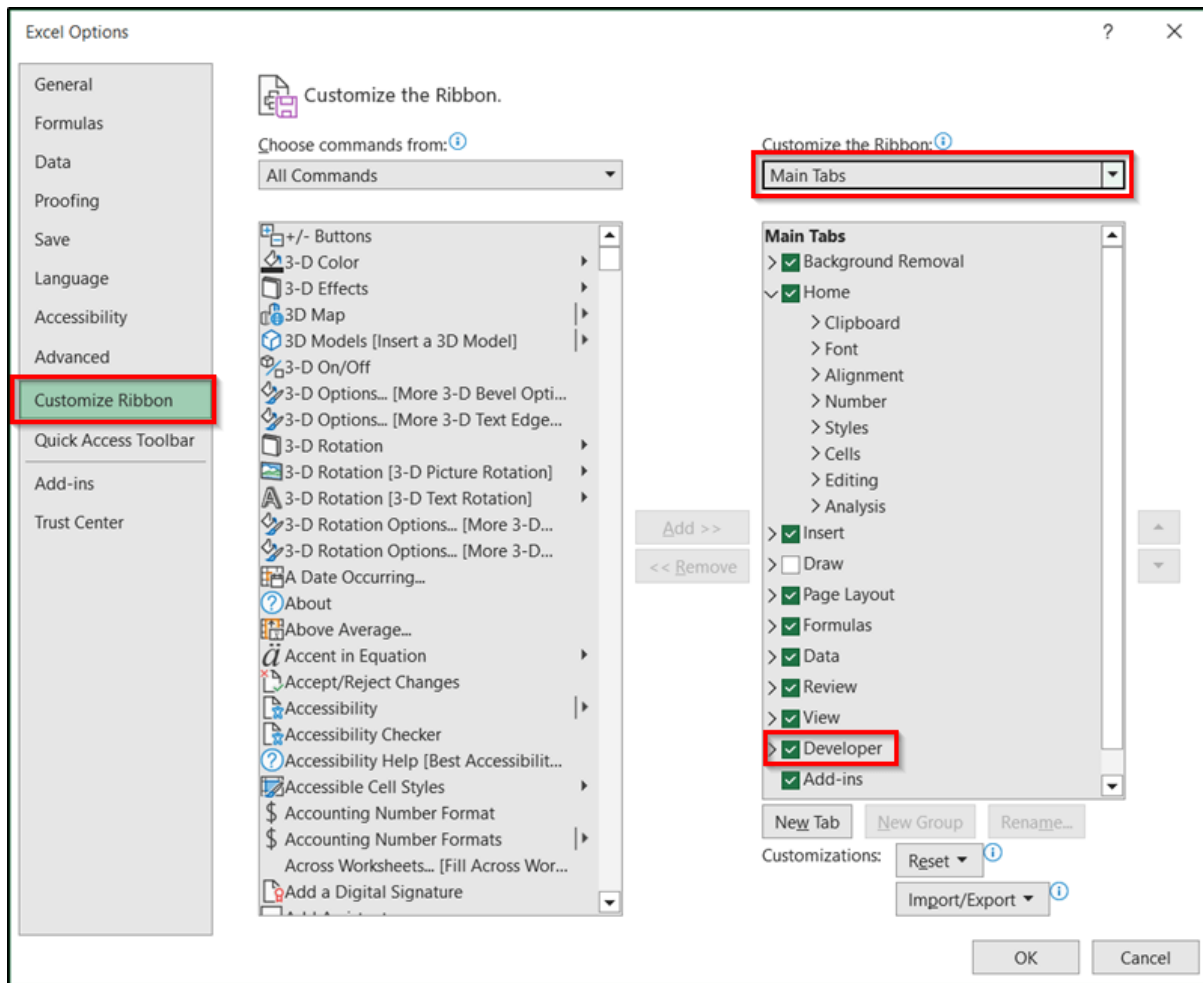
- First, click on the **File** tab on your ribbon.



- Then select **Options** from the backstage view.

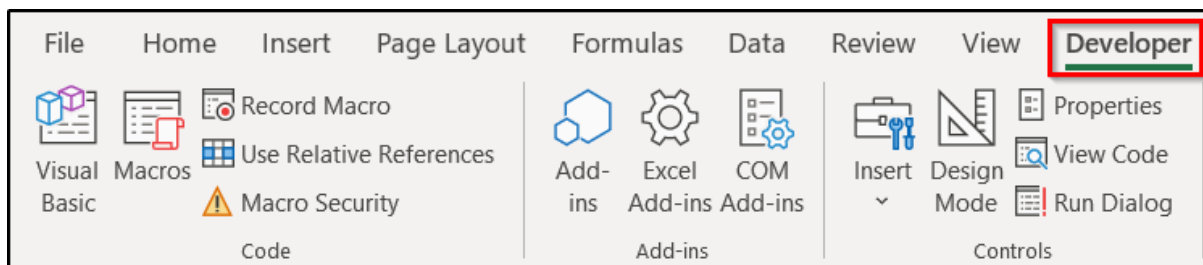


- As a result, the **Excel Options** box will open up.
- Now go to the **Customize Ribbon** tab here.
- After that, select **Main Tools** under **Customize the Ribbon**.
- Then check **Developer** under **Main Tools** in the box below it.



- Finally, click on **OK**.

This will add the **Developer** tab to your ribbon.

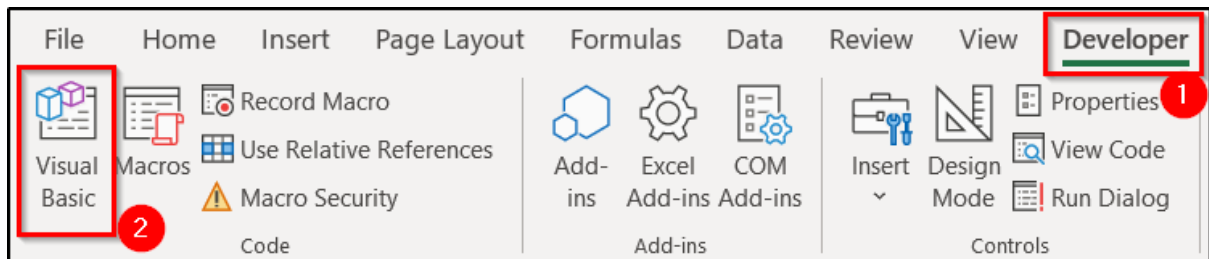


Inserting Module in VBA

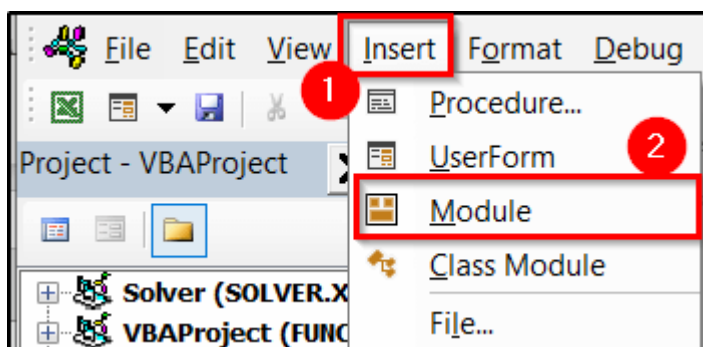
Once you have the **Developer** tab, you will have access to the VBA editor window that will enable you to add, edit or remove a VBA code. Individual VBA codes are generally stored in a place called **Module**. It is usually a good practice to store different codes in different modules. Here is how you can create a module in the VBA window.

Steps:

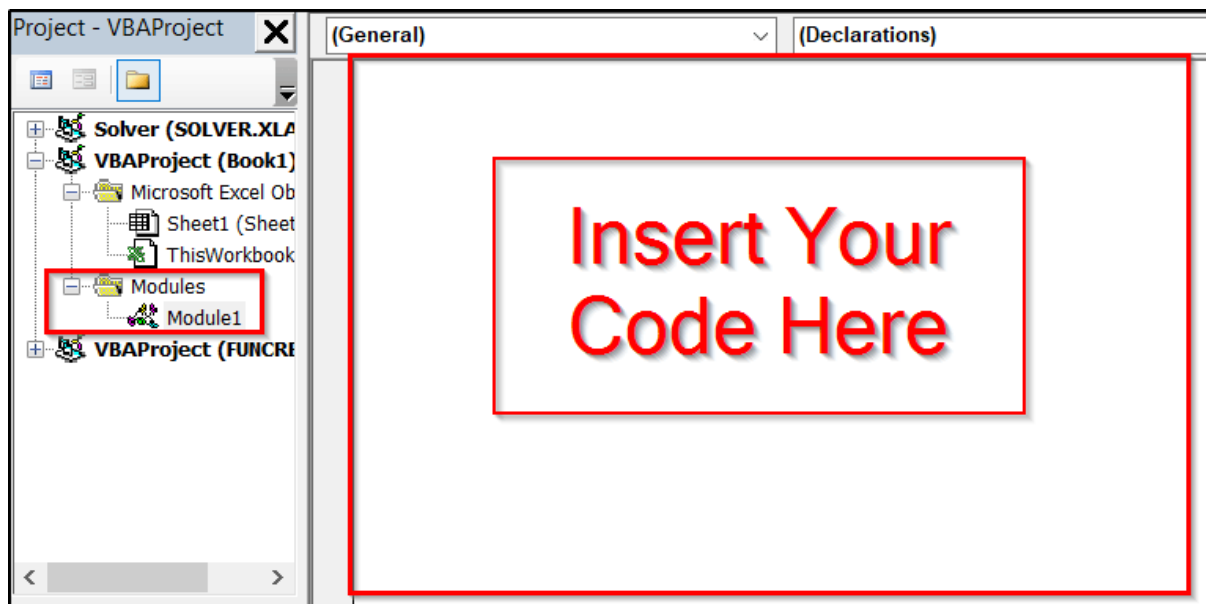
- First, go to the **Developer** tab on your ribbon.
- Then select **Visual Basic** from the **Code** group section.



- At this point, you will notice the VBA window open up on your screen. Now click on the **Insert** tab on the window.
- Then select **Module**.



This will create a module in the VBA window. You can find the modules on the left side of the window and on the right you can write the code.



Writing and Running VBA

After you create a module, you can write or insert any valid code there to run it. If there is no error in the code, VBA will execute it line by line and return the final result.

For example, let's take a simple VBA code that turns a cell color blue. Follow the previously described steps to insert a module and insert the following code there.

```
Sub change_cell_blue()  
Range("B2").Interior.ColorIndex = 37  
End Sub
```

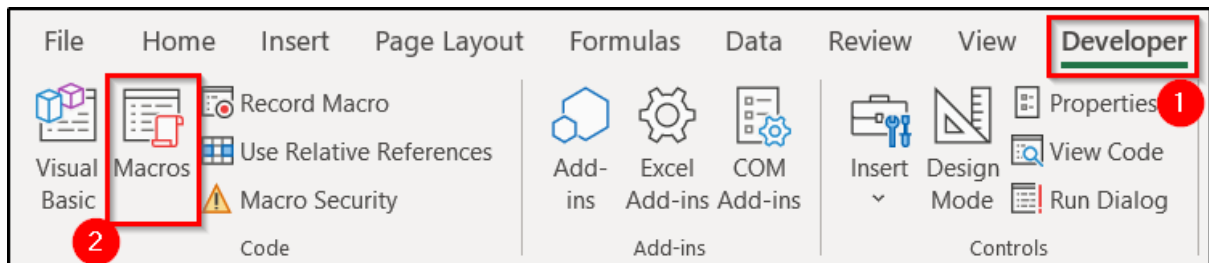
Copy

Now press F5 to run the code instantly.

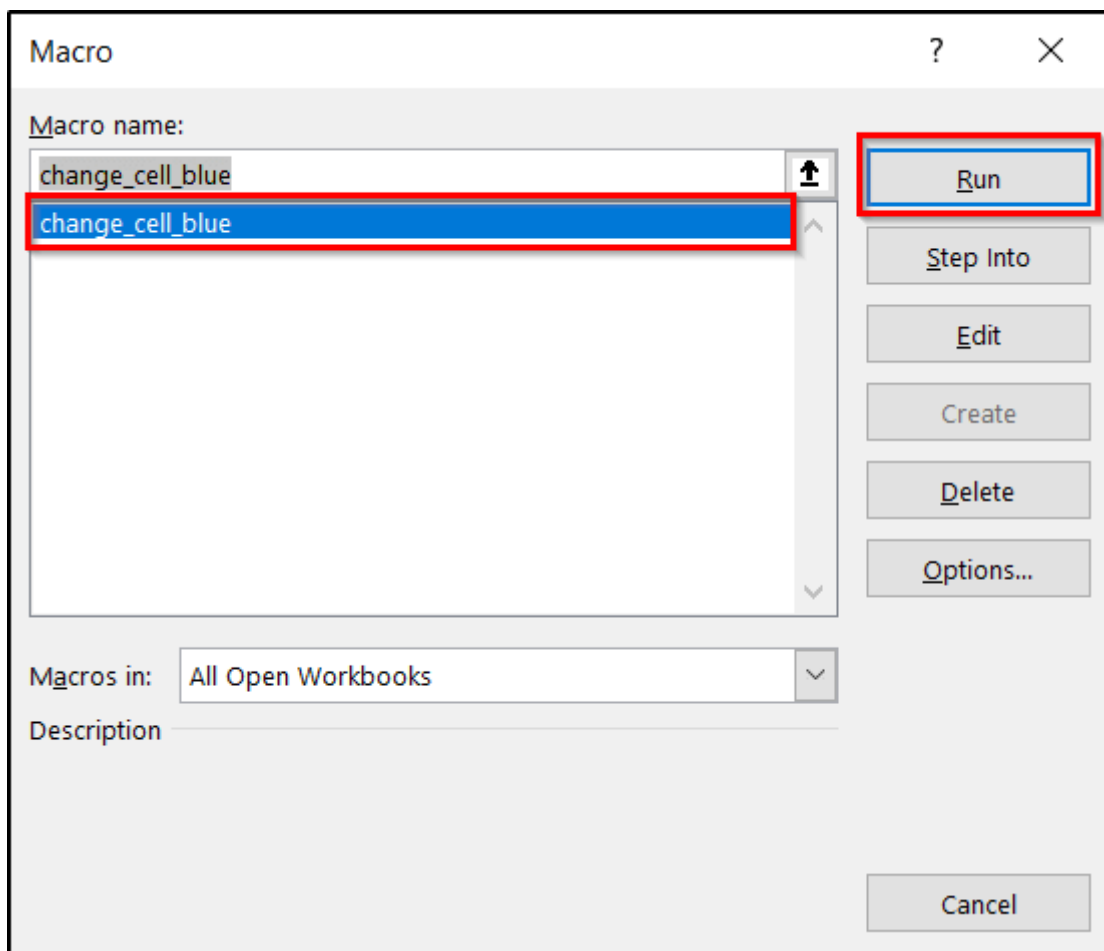
	A	B	C
1			
2			
3			
4			

You will notice the cell colour of **B2** of the active spreadsheet will change to light blue.

If you want to run the code later, you can also do that by going to the **Developer** tab and selecting **Macros** from the **Code** group.



Then you can see the list of macros the workbook contains. Select the one with the proper sub name (the name you enter after “sub” in the VBA code) and click on **Run**.



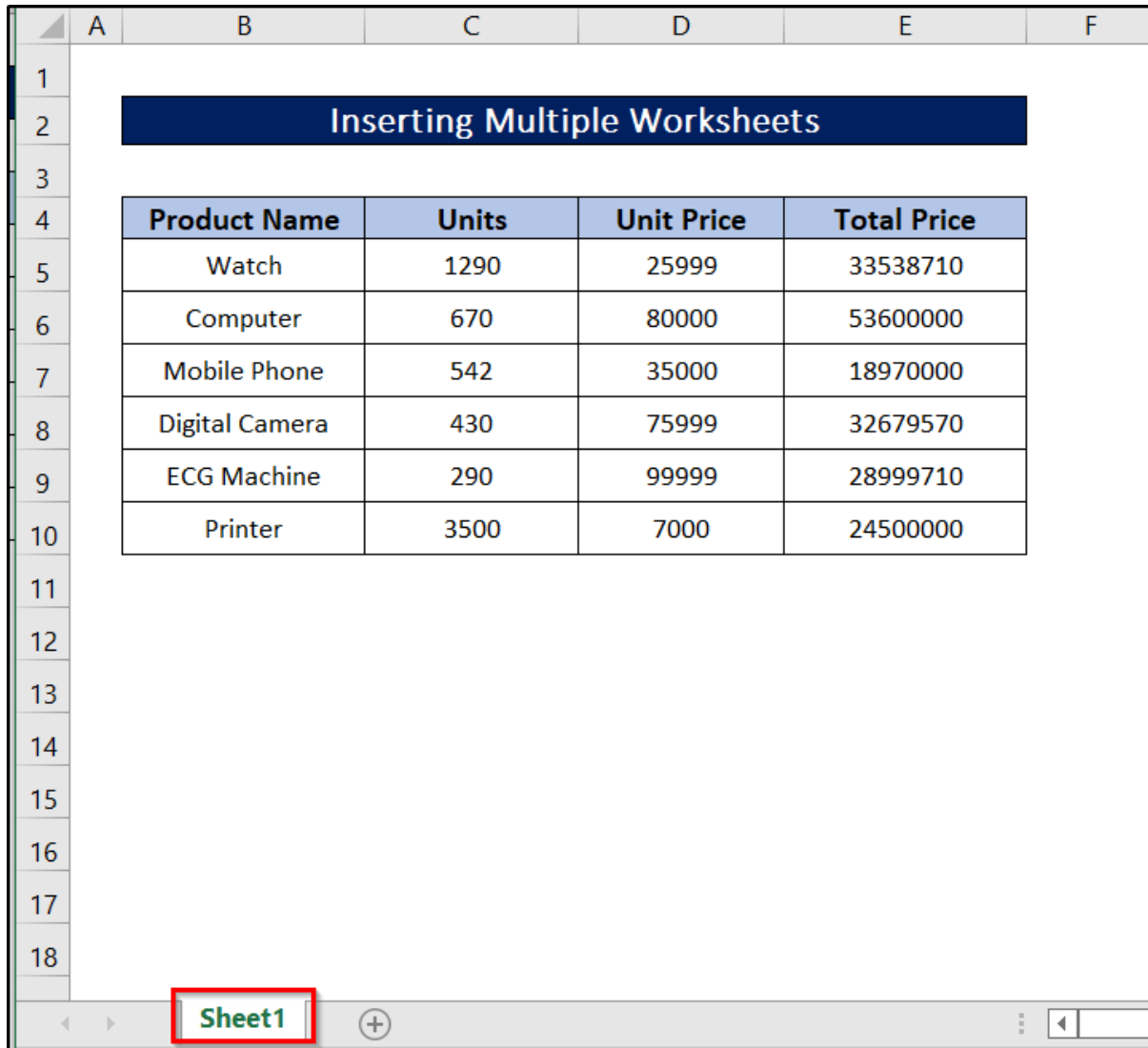
Examples of VBA Codes That Will Give You an Idea of Its Application

Moving on, let's see a bit more advanced examples as the introduction of VBA features and applications for different purposes. Some of these processes take a longer time to complete using manual processes. We can perform them effortlessly through VBA.

For demonstration, we have a sample dataset like this.

	A	B	C	D	E
1					
2		Introduction to VBA Features and Applications			
3					
4		Product Name	Units	Unit Price	Total Price
5		Watch	1290	25999	33538710
6		Computer	670	80000	53600000
7		Mobile Phone	542	35000	18970000
8		Digital Camera	430	75999	32679570
9		ECG Machine	290	99999	28999710
10		Printer	3500	7000	24500000
11					

Inserting Multiple Worksheets



The screenshot shows an Excel spreadsheet with a dark blue header bar at the top containing the text "Inserting Multiple Worksheets". Below this, a table with four columns is displayed: "Product Name", "Units", "Unit Price", and "Total Price". The table contains six rows of data. At the bottom of the spreadsheet, the sheet tab "Sheet1" is visible and highlighted with a red rectangle. To the right of the sheet tab is a plus sign icon for adding new sheets.

Product Name	Units	Unit Price	Total Price
Watch	1290	25999	33538710
Computer	670	80000	53600000
Mobile Phone	542	35000	18970000
Digital Camera	430	75999	32679570
ECG Machine	290	99999	28999710
Printer	3500	7000	24500000

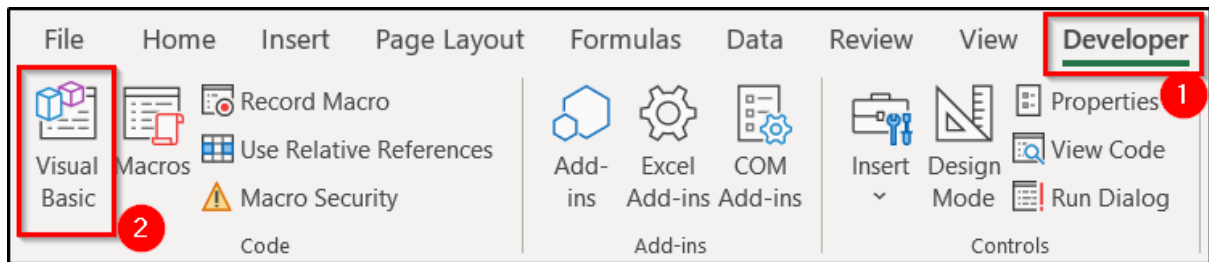
Now let's try a code that can enter multiple sheets at a time. Usually, you can just click on the plus sign beside the sheet name at the bottom. But the beauty of the code is you can do that for any number of sheets avoiding all the clicks.

Follow these steps to see how you can do that.

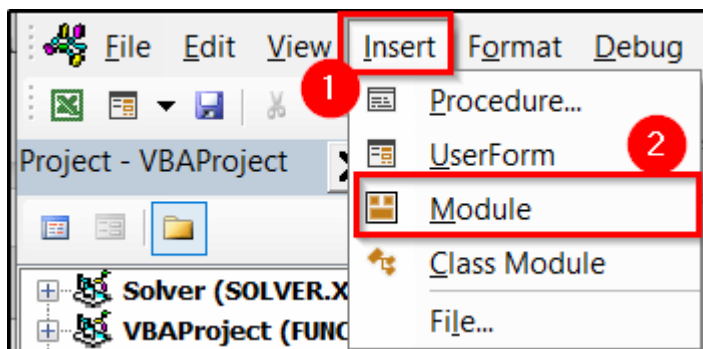
Steps:

- First, go to the **Developer** tab on your ribbon.

- Then select **Visual Basic** from the **Code** group section.



- Next, click on **Insert** in the VBA window and select **Module** in from the drop-down.

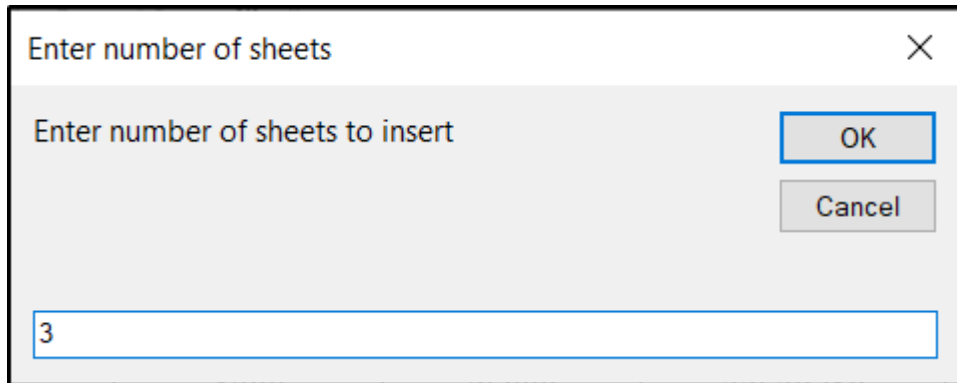


- Now select the module by double-clicking on it if it isn't selected already.
- Then insert the following code in it.

```
Sub Add_Multiple_Sheets()
'Declaring variable
Dim SheetsNumber As Integer
'Put the number of sheets
SheetsNumber = InputBox("Enter number of sheets to insert",
"Enter number of sheets")
'Adding the additional sheets after the current active sheet
Sheets.Add After:=ActiveSheet, Count:=SheetsNumber
End Sub
```

Copy

- After that, press **F5** to run the code instantly.
- As a result, a pop-up box will appear asking for the number of sheets you want to insert. For the demonstration, let's say we insert 3.



Enter number of sheets

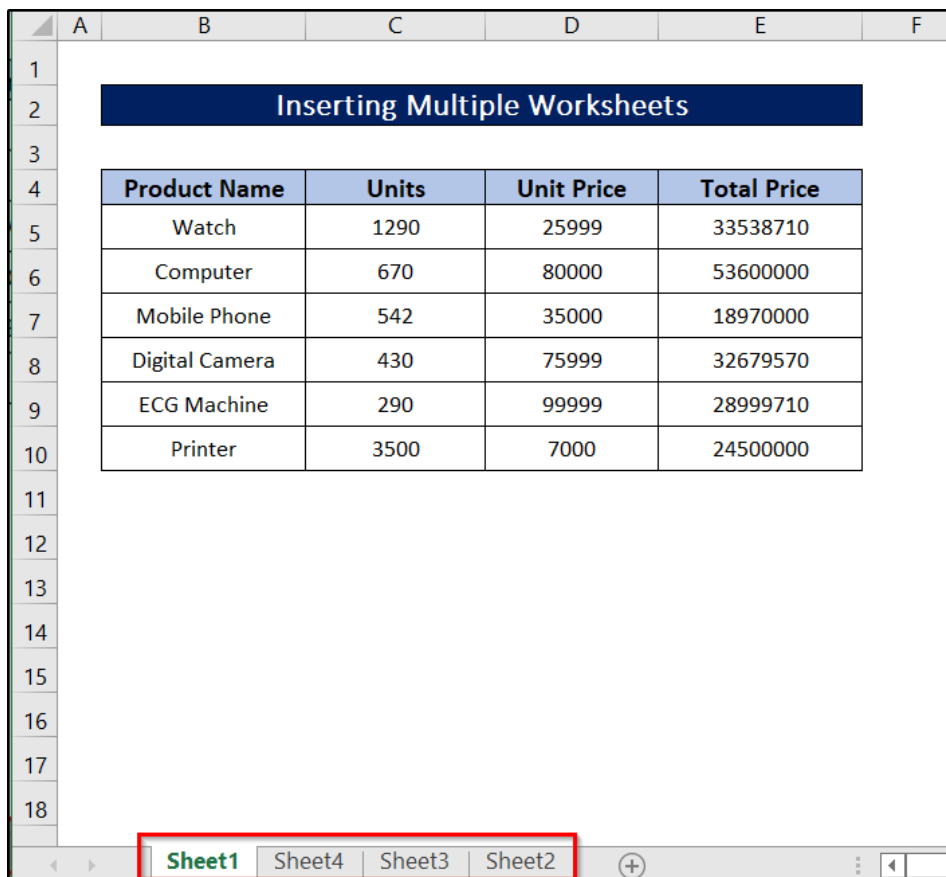
Enter number of sheets to insert

OK

Cancel

3

- Then press **OK**.
- You will notice there will be new sheets in the workbook because of the execution of the code.



Inserting Multiple Worksheets			
Product Name	Units	Unit Price	Total Price
Watch	1290	25999	33538710
Computer	670	80000	53600000
Mobile Phone	542	35000	18970000
Digital Camera	430	75999	32679570
ECG Machine	290	99999	28999710
Printer	3500	7000	24500000

Sheet1 Sheet4 Sheet3 Sheet2

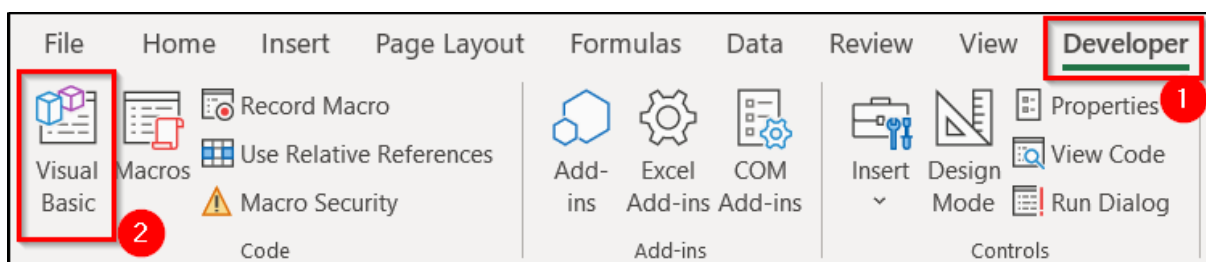
Changing Letters from Upper Case to Lower case

Let's now look at the usage of **FOR** and **IF** as part of the introduction to VBA features and applications. We will be utilizing them for changing string values in a range to all upper-case letters.

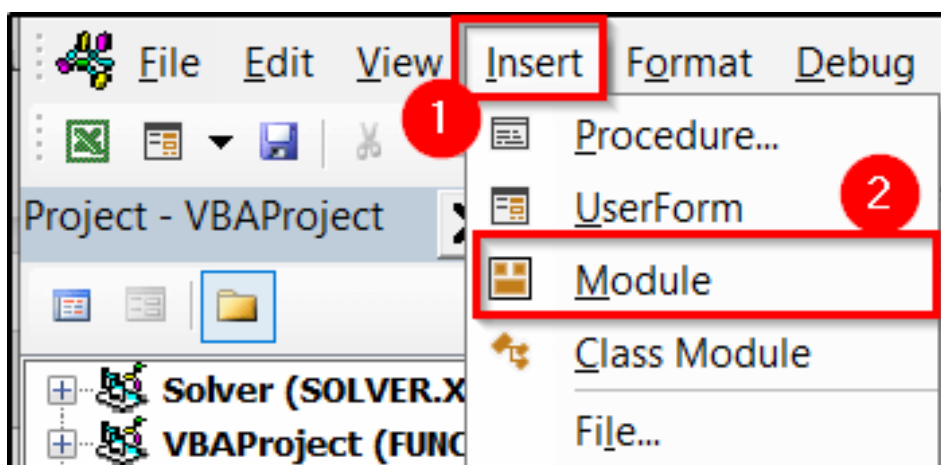
Follow these steps to see how we can do that.

Steps:

- First, select the range you want to change to upper case. Here, we are selecting the range **B5:B10** for the demonstration.
- go to the **Developer** tab on your ribbon.
- Then select **Visual Basic** from the **Code** group section.



- Next, click on **Insert** in the VBA window and select **Module** in from the drop-down.



- Now select the module by double-clicking on it if it isn't selected already.
- Then insert the following code in it.

```
Sub Convert_To_UPPER_Case()  
'Declaring variable  
Dim Xrng As Range  
'Using For Each loop  
For Each Xrng In Selection.Cells  
'Using If Statement  
If Xrng.HasFormula = False Then  
'Specify the Range  
Xrng.Value = UCase(Xrng.Value)  
End If  
Next Xrng  
End Sub
```

Copy

- Now press F5 to run the code instantly.

The dataset will look like this now with all the cells in the selected range containing upper letters.

	A	B	C	D	E
1					
2		Changing Letters from Upper Case to Lower Case			
3					
4		Product Name	Units	Unit Price	Total Price
5		WATCH	1290	25999	33538710
6		COMPUTER	670	80000	53600000
7		MOBILE PHONE	542	35000	18970000
8		DIGITAL CAMERA	430	75999	32679570
9		ECG MACHINE	290	99999	28999710
10		PRINTER	3500	7000	24500000
11					

Some VBA Shortcuts

After some VBA features and applications, here are some of the important shortcut keys you can use as an introduction.

F1: Opens up help options

F5: Runs the current code in the module

F8: Steps into a code by lines

F9: Inserts a breakpoint

Shift+F2: Motions to the definition of function or procedure

Shift+F8: Starts executing the code by one procedure at a stretch

Shift+F10: Shows shortcut menu for the selected item

Ctrl+F6: Moves to the next window

Ctrl+F8: Run to the position of the cursor

Ctrl+F10: Activates menu bar

Ctrl+Shift+F8: Step out of the code

Ctrl+Shift+F9: Clear all breakpoints

Alt+F11: Toggles between VBA and application window

Ctrl+A: Select all in the module

Ctrl+C: Copy selection

Ctrl+S: Save the current state

Ctrl+V: Pastes clipboard materials

Ctrl+X: Cuts selection

Shift+Insert: Pastes current clipboard material

Tab: Indents line of code

Shift+Tab: Unindent code

Enter: Insert a new line in the code

Shift+Home: Select the start of the line

Shift+End: Selects end of line

Ctrl+Tab: Moves to the next module