# **Visual Basic Editor – How to Open and Use it in Excel**

## **What is Visual Basic Editor in Excel?**

Visual Basic Editor is a separate application that is a part of Excel and opens whenever you open an Excel workbook. By default, it’s hidden and to access it, you need to activate it.

VB Editor is the place where you keep the VB code.

There are multiple ways you get the code in the VB Editor:

1. When you [record a macro](https://trumpexcel.com/record-macro-vba/), it automatically creates a new module in the VB Editor and inserts the code in that module.
2. You can manually type VB code in the VB editor.
3. You can copy a code from some other workbook or from the internet and paste it in the VB Editor.

## **Opening the VB Editor**

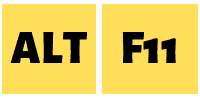
There are various ways to open the Visual Basic Editor in Excel:

1. Using a Keyboard Shortcut (easiest and fastest)
2. Using the Developer Tab.
3. Using the Worksheet Tabs.

Let’s go through each of these quickly.

### **Keyboard Shortcut to Open the Visual Basic Editor**

The easiest way to open the Visual Basic editor is to use the keyboard shortcut – **ALT + F11** (hold the ALT key and press the F11 key).



As soon as you do this, it will open a separate window for the Visual Basic editor.

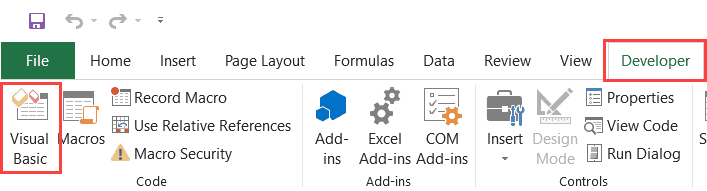
This shortcut works as a toggle, so when you use it again, it will take you back to the Excel application (without closing the VB Editor).

*The shortcut for the Mac version is* ***Opt + F11*** *or* ***Fn + Opt + F11***

### **Using the Developer Tab**

To open the Visual Basic Editor from the ribbon:

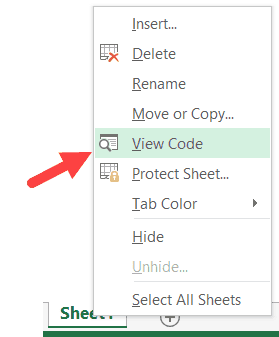
1. Click the Developer tab (if you don’t see a developer tab, [read this](https://trumpexcel.com/excel-developer-tab/) on how to get it).
2. In the Code group, click on Visual Basic.



### **Using the Worksheet Tab**

This is a less used method to open the Vb Editor.

Go to any of the worksheet tabs, right-click, and select ‘View Code’.



This method wouldn’t just open the VB Editor, it will also take you to the code window for that worksheet object.

This is useful when you want to write code that works only for a specific worksheet. This is usually the case with worksheet events.

## **What is an Immediate Window in Excel VBA?**

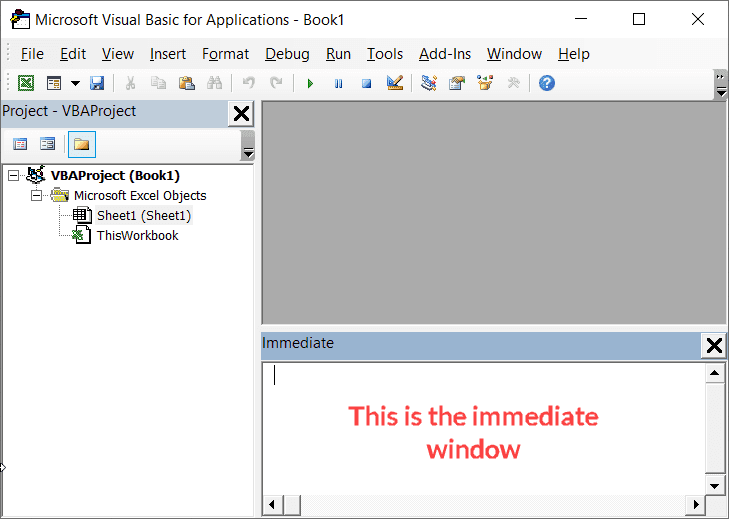
Immediate Window is a part of the **VB Editor** which you can use to do some quick stuff.

To give you an example, if you have a workbook with some hidden worksheets and you want to quickly know the count of total worksheets, you can do that in a few seconds with the Immediate window.

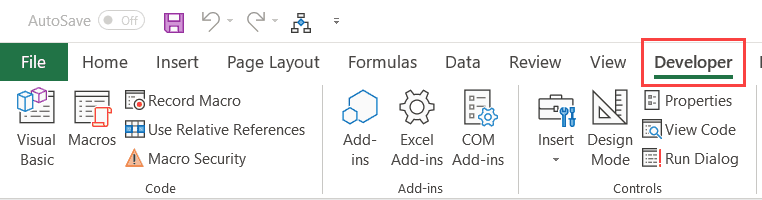
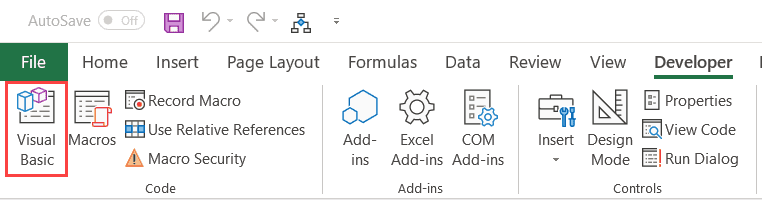
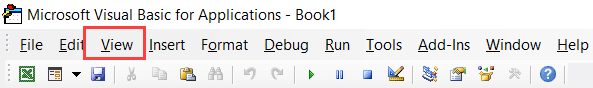
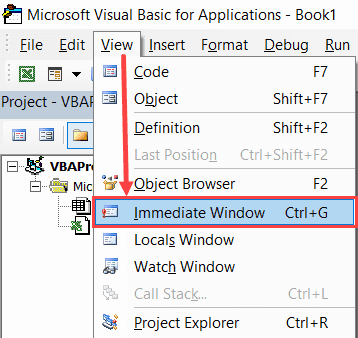
Immediate Window does a lot more (as you’ll see later in this tutorial). But to begin with, let’s just think of an immediate window as a tool to help you speed up your work in Excel VBA.

## **Where to Find the Immediate Window?**

When you open the Excel VB-Editor, you may already see the immediate window as a part of the VB Editor. And if you don’t see it already, you can easily make it show up.



Below are the steps to open the VB Editor and make the Immediate Window visible:

1. Click the ‘Developer’ tab in the Excel ribbon
2. In the Code Group, click on ‘Visual Basic’. This will open the VB Editor
3. In the VB Editor, click on the ‘View’ option in the menu
4. Click on Immediate Window. This will make the immediate window show up in the VB Editor.

If you’re more comfortable with shortcuts, below are some to speed up the above steps:

* To open the VB Editor – **ALT + F11** (this works even if you don’t have the Developer tab in the ribbon)
* To show the immediate window – **Control + G** (use this one the VB Editor is open)

## **Examples of Using Immediate Window in Excel VBA**

Now that you know where to find the immediate window, let’s see some awesome examples where you can use it when working with Excel VBA.

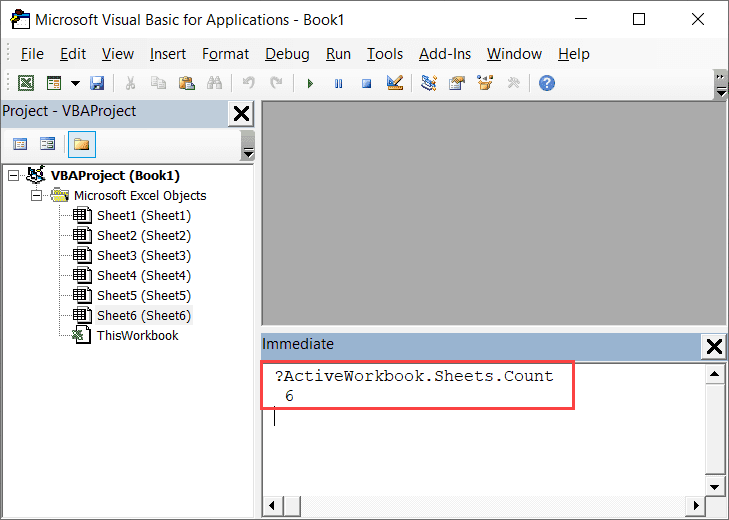
### **Get Information About the Files/Workbooks**

The ‘Immediate Window’ is a place where you can ask questions about the workbooks and it will give you the answer right away.

For example, suppose you have a workbook and you want to know how many sheets are there in the workbook, you can type the below code in the immediate window and hit the enter key.

?ActiveWorkbook.Sheets.Count

This will instantly tell you the total number of sheets in the active workbook.



This can be useful when you have a workbook that has a lot of [sheets and you can’t count](https://trumpexcel.com/count-sheets-excel/) it manually (or don’t want to), or when you have a workbook where there are hidden sheets and you want to know the total count.

The ? (question mark) is supposed to be used before the query so that VBA can understand that you’re asking a question. If you don’t use this question mark, immediate window will not give you the detail/answer.

Now, this is a really simple example where Immediate Window gives you some information and saves time.

Below are some more examples:

To get the name of the active workbook

?ActiveWorkbook.Name

To get the name of the active sheet

?Activesheet.Name

To get the path of the Workbook (the address where it’s saved)

?ActiveWorkbook.Path

So if you need something about an object (such as Workbook, sheets, charts, shapes, range, etc.), you can use immediate window to quickly get this information.

### **Get Debug.Print Information**

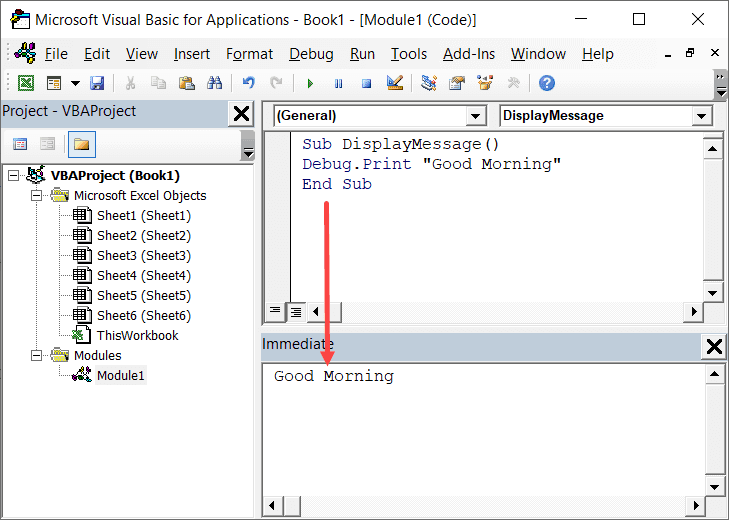
In Excel VBA, Debug.Print is used to show the value of a variable instantly in the immediate window.

For example, the below code would instantly show the message “Good Morning” in the immediate window.

Sub DisplayMessage()

Debug.Print "Good Morning"

End Sub



You can use the Debug. Print a line in your code to quickly get some data in the immediate window or to debug your code.

For example, if you want to get the names of all the sheets in a workbook, you can use the below code:

Sub GetSheetNames()

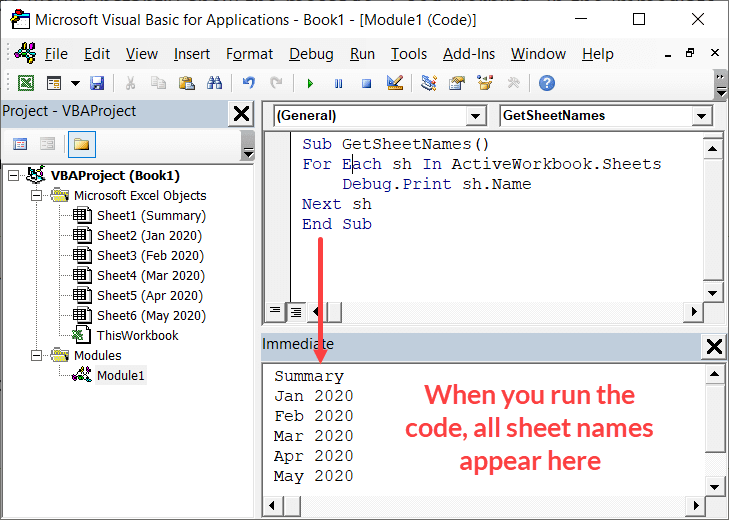
For Each sh In ActiveWorkbook.Sheets

Debug.Print sh.Name

Next sh

End Sub

The above code goes through each sheet in the active workbook and gives the name in the immediate window.



Debug.Print is a useful technique to debug your code. For example, if you’re running a loop and want to see how many times the loop was run, you can simply place a Debug.Print line that simply shows the incrementing numbers when each loop runs.

If you’re using *Debug.Print* to debug the code, remember to remove it when you’re done.