**Debugging and Immediate Window**

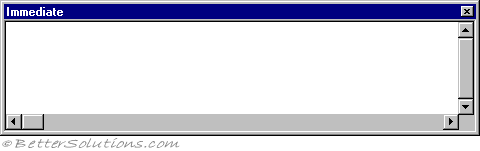
<http://www.bettersolutions.com/vba/VOK159/YD519120554.htm>

What is the Immediate Window?

* The Immediate window is extremely useful and allows you to test and execute small snippets of code.
* This can be used for executing VBA statements directly, testing statements and helping to debug your code
* Lets you run any VBA statement, manually while a program is running.
* The immediate window is particularly useful for testing variables or properties
* The Immediate window can be dragged and positioned anywhere on your screen unless you have made it a dockable window from the Docking Tab of the Options dialog box.
* You can close the window by clicking the Close box. If the Close box is not visible, double-click the Title bar to make the Close box visible, then click it.

**(View > Immediate Window)**

**Shortcut key (Ctrl + G).**



There are two ways to print to the immediate window, either by including Debug.Print statements in the actual code or by entering print statements directly.

* Debug.Print Range("A1").Value
* ? Range("A1").Value

' "?" is a shortcut for "print"

**Print in "Immediate Window"**

Printing from Application Code

Debug.Print "Salary = "; Salary

**Printing Values of Properties**

? BackColor

? Text1.Height

? Form1.BackColor

? Form1.Text1.Height

' "?" is a shortcut for "print"

?cells(2,1)

?range("A5")

print Cells(2,1)

# **Printing Information in the Immediate Window**

<http://msdn.microsoft.com/en-us/library/aa716276(v=vs.60).aspx>

## **Printing from Application Code**

The Print method sends output to the Immediate window whenever you include the Debug object prefix:

**Debug.Print** [*items*][;]

For example, the following statement prints the value of Salary to the Immediate window every time it is executed:

**Debug.Print "Salary = "; Salary**

This technique works best when there is a particular place in your application code at which the variable (in this case, Salary) is known to change. For example, you might put the previous statement in a loop that repeatedly alters Salary.

**Note**   When you compile your application into an .exe file, Debug.Print statements are removed. Thus, if your application only uses Debug.Print statements with strings or simple variable types as arguments, it will not have any Debug.Print statements. However, Visual Basic will not strip out function calls appearing as arguments to Debug.Print. Thus, any side-effects of those functions will continue to happen in a compiled .exe file, even though the function results are not printed.

## **Printing from Within the Immediate Window**

Once you're in break mode, you can move the focus to the Immediate window to examine data.

**To examine data in the Immediate window**

1. Click the **Immediate** window (if visible).

-or-

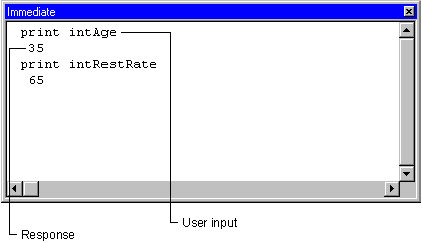
From the **View** menu, choose **Immediate Window**.

Once you have moved focus to the Immediate window, you then can use the Print method without the Debug object.

1. Type or paste a statement into the **Immediate** window, and then press ENTER.

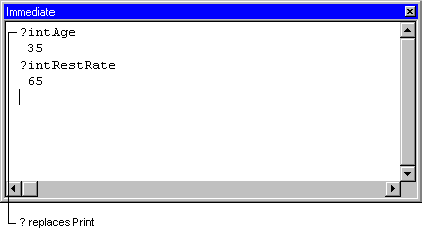
The Immediate window responds by carrying out the statement, as shown in Figure 13.19.

**Figure 13.19   Using the Print method to print to the Immediate window**



A question mark (?) is useful shorthand for the Print method. The question mark means the same as Print, and can be used in any context where Print is used. For example, the statements in Figure 13.19 could be entered as shown in Figure 13.20.

**Figure 13.20   Using a question mark instead of the Print method**



## **Printing Values of Properties**

You can evaluate any valid expression in the Immediate window, including expressions involving properties. The currently active form or module determines the scope. If the execution halts within code that is attached to a form or class, you can refer to the properties of that form (or one of its controls) and make the reference to the form implicit with statements like the following:

? BackColor

? Text1.Height

Assuming that Text1 is a control on the currently active form, the first statement prints the numeric value of the current form's background color to the Immediate window. The second statement prints the height of Text1.

If execution is suspended in a module or another form, you must explicitly specify the form name as follows:

? Form1.BackColor

? Form1.Text1.Height

**Note**   Referencing an unloaded form in the Immediate window (or anywhere else) loads that form.

**For More Information**   To learn about changing properties and values in the Immediate window, see "Assigning Values to Variables and Properties" later in this chapter.

--================================

Debugging VBA Code

<http://www.cpearson.com/excel/DebuggingVBA.aspx>

*Stepping Through Code*

One of the first methods to debug code is to step through the code one line at a time. To step through code, put the cursor on the first line of code to be analyzed and press **F8** or choose **Step Into** on the Debug menu. The next line of code to be executed will be displayed in yellow background with a black font. Note that the highlighted line of code has not yet been executed -- it is the next line to execute.

If your code calls another procedure, stepping through the code with F8 will cause execution to enter the called procedure in a line-by-line sequence. If you want to execute the called procedure without stepping through it, press **SHIFT F8**. This will execute the called procedure and then pause on the line of code after calling the procedure.

If you are already stepping through a procedure, you can press **CTRL F8** to resume code execution line-by-line. At any time you are paused either in step-by-step mode or at a breakpoint (see below), you can press **F5** or Continue from the Run menu to cause VBA to run to completion or until a pause statement is encountered.

Whenever you are paused in step-by-step mode, you can query or change a variable's value from the Immediate window.

*Break Points And The Stop Command*

A breakpoint is a marker placed on a line of code that causes execution to pause immediately before executing that line. You can add a breakpoint to a line of code by putting the cursor on the line of code in question and pressing **F9**, choosing **Toggle Breakpoint** on the Debug menu, or clicking in the left margin next to the line of code. When a breakpoint is set, the line is displayed in brick-red background with a white font. When you run the code, execution will pause immediately before the line of code with the breakpoint and will display it in yellow background with a black font. Note than the line in yellow has not yet been executed -- it is the next line of code to run.

While the code is paused at the breakpoint, you can issue commands in the Immediate window to change or query a variable's value. To view the content of a variable, enter a ? character followed by the name of the variable and then press ENTER. You can change a variable's value by entering VariableName = NewValue in the Immediate window and pressing ENTER.

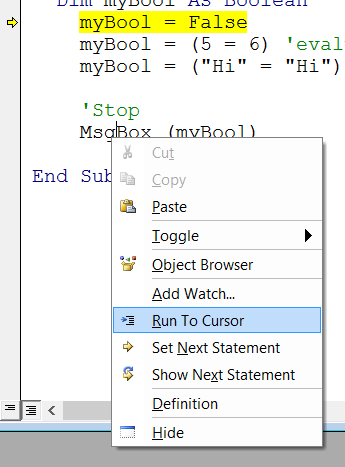
If the Immediate window is not visible (typically at the bottom of the VBA Editor screen), press CTRL G or chooseImmediate Window from the View menu to make the window visible.

To remove a breakpoint, put the cursor on the line of code and press F9. You can clear all breakpoints by choosing Clear All Breakpoints from the Debug menu or pressing **CTRL SHIFT F9**.

VBA also provides the **Stop** command. This simply stops code execution on that line of code and enters break mode.

Once you are finished debugging the code, be sure to go back and clear all breakpoints (choose Clear All Breakpoints from the Debug menu or press CTRL SHIFT F9) and be sure to remove or comment out all Stopstatements.

When you are paused at a breakpoint or in step-by-step mode, you can change the next line to be executed, either before the current line to re-run a section of code, or after the line to skip statements. Right-click the line where you want execution to resume and right-click and choose Set Next Statement or choose Set Next Statement from the Run menu. Execution will resume at the selected line of code.



*The Debug Command*

VBA provides a Debug object with two properties, Print and Assert that you can use display a variable's value and to control the program flow. Debug.Print will write what follows it to the Immediate window. Code execution is not interupted. After displaying the text in the Immediate window, code execution continues to run. You can mix literal text with variable names in the Debug.Print statement. For example,

Debug.Print "The value of variable X is: " & X

You can display several variables at once in the Immediate window by separating them with commas. For example,

Debug.Print X, Y, Z

The Debug.Assert command is a conditional breakpoint that will cause execution to pause on the Debugstatement if the expression that following the Assert statement is False. For example,

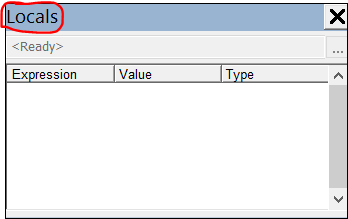
Debug.Assert Var >= 0

This will pause on the Debug.Assert statement if Var >= 0 is False; that is, it will pause if Var is negative. It may seem backwards that execution is paused when the condition is False rather than True, but the Assertmethod was adopted from the C language, and its usage remained the same as in C.

Be sure to remove or comment out the Debug.Print and Debug.Assert statements when you are finished debugging. You generally don't want these statements to be operative during normal usage of your application.

*The Locals Window*

The Locals windows allows you to view the value of all the variables in a procedure when you are stepping through the procedure. To display the Locals window, choose Locals Window from the View menu. Using the Locals window is easier to display variable values than examining the value from the Immediate window. For simple variable types (e.g., Long and String variables), the value is displayed on one line. For complex types or objects (e.g., a Range variable), its properties are displayed in a collapsible tree-like structure.

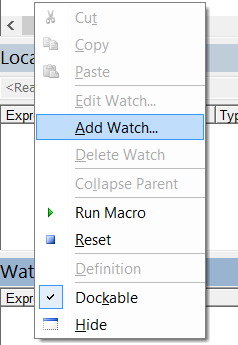


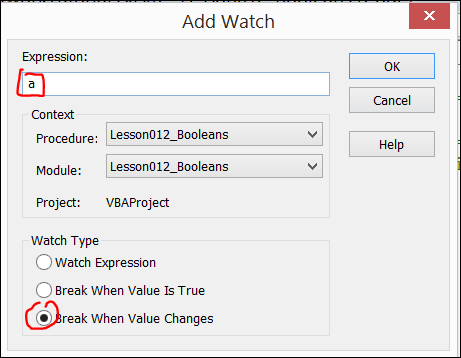
*The Watch Window*

The Watch window displays all the Watches in effect. You can display the Watch window by choosing Watch Window from the View menu.

A Watch is an instruction to VBA to pause code when an expression is True or when the variable being watched changes value.

To create a Watch on a variable, open the Watch window and right-click in the Watch window and choose **Add Watch**... from the popup menu or choose Add Watch... from the Debug windows. In the Add Watch dialog, **enter in the Expression text box a variable name whose value you want to watch**. Then choose Break When Value Changes. When you run the code, execution will pause at the line after the line that modifies the variable's value. When code pauses, the value of the variable will have already been updated.





To remove a Watch, right-click it in the Watch window and choose Delete Watch from the popup menu. To modify a Watch, right-click it in the Watch window and choose Edit Watch from the popup menu.

*The Call Stack*

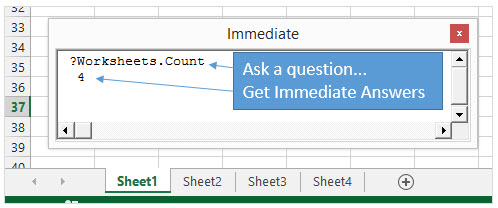
The Call Stack is a data structure maintained by VBA that tracks which procedure called another procedure. For example, if procedure AAA calls BBB which calls CCC, the Call Stack window will display the list of procedures starting with the most recent procedure and below that, the chain of procedures that were executed to get to the current position. You can view the Call Stack by choosing Call Stack from the View menu. This is useful to track the flow of execution that ended up in the current location. Unfortunately, there is no programmatic way to get information from the call stack.

--==============================================================

## 5 Ways to Use the VBA Immediate Window in Excel

<http://www.excelcampus.com/vba/vba-immediate-window-excel/>

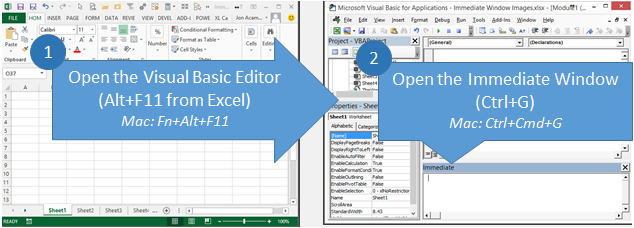
The VBA Immediate Window is an awesome tool that allows you to get immediate answers about your Excel files, and quickly execute code.  It is built into the Visual Basic Editor, and has many different uses that can be very helpful when writing macros, debugging code, and displaying the results of your code.

**[](http://www.excelcampus.com/wp-content/uploads/2014/09/VBA-Immediate-Window-Excel-Worksheets-Count.jpg)**

Every Excel user can benefit from the Immediate Window, even if you’re not writing macros.  This post will explain 5 different uses for the Immediate Window.  Once you understand the capabilities of this tool, you will find yourself using it all the time.

## Where is the Immediate Window?

The Immediate window is located in the **Visual Basic Editor window**.

**[](http://www.excelcampus.com/wp-content/uploads/2014/09/Open-Visual-Basic-Editor-and-Immediate-Window-Excel-Mac.png)**

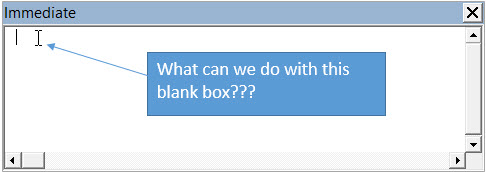
The fastest way to get to the Immediate Window is to:

1. Press **Alt+F11** (hold Alt key, then press F11 key) from anywhere in Excel.  The Visual Basic Editor window will open.  (Mac version is Fn+Alt+F11)
2. Pressing **Ctrl+G** opens the Immediate Window and places the text cursor in it.  Begin typing your code.  (Mac version is Ctrl+Cmd+G)

When you open the VB Editor (Alt+F11) you might see the Immediate Window automatically appear in the bottom right corner.  This is its default location.  If it’s not there you can press Ctrl+G or View menu > Immediate Window.

## This Blank Box is Magical!

When you click inside the Immediate Window you will just see a blank box with the text cursor flashing.  At first glance this doesn’t look too exciting, but the Immediate window can be a very **powerful and useful tool**.

**[](http://www.excelcampus.com/wp-content/uploads/2014/09/What-Does-the-Immediate-Window-Do.jpg)**

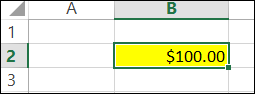
Think of it like a blank cell in a worksheet.  It’s pretty boring until you add a formula to it, right?  Well the Immediate Window is very similar, so let’s look at**5 examples that will help you get the most out of this magical box**.

# **#1 – Get Info About The Active Workbook**

The simplest use for the Immediate window is to quickly get information about the workbook that you currently have open and active in the background.  **You can evaluate any line of VBA code in the Immediate Window**, and it will immediately give you the result.

For example, to find out how many sheets are in the active workbook, type the following line of code in the Immediate window and then press the Enter key.

**?Activeworkbook.Worksheets.Count**



**?Range("B2").Value**

100

**?Activecell.NumberFormat**

$#,##0.00

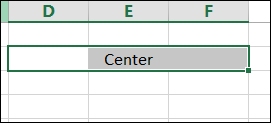
**?Activecell.Interior.Color**

65535

# **#2 – Execute a Line of VBA Code**

You don’t have to write a whole macro if you just need to perform one line of code to your workbook.

Remove the question mark at the front of the statement and the Immediate Window will execute or perform that line of code.



**Selection.HorizontalAlignment = xlCenterAcrossSelection**

You can also use the following line of code to make a worksheet “very hidden”.

**Worksheets("Sheet2").Visible = xlVeryHidden**

Another example is to hide the contents of a cell by making its font color the same as its fill (background) color.

**Range("A1").Font.Color = Range("A1").Interior.Color**

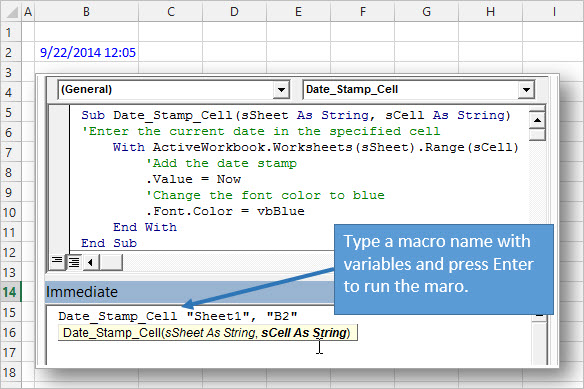
# **#3 – Run a Macro**

You can run a macro from the Immediate Window by typing the name of the macro (procedure), and pressing Enter.

Of course you can also do this by pressing the F5 key or the Run Macro button in the VB Editor, but what if your macro contains arguments?

**A macro cannot be run from within the procedure if it contains arguments.** However, you can call the macro from the Immediate Window.

The example below is a simple macro that enters the current date (Now) in the cell , and changes the font color to blue (Color = vbBlue).  This macro requires two arguments to be passed to it to run, the worksheet name and cell address where the date stamp will be entered.

**[](http://www.excelcampus.com/wp-content/uploads/2014/09/Run-VBA-Macro-from-Immediate-Window-with-Arguments-Excel.jpg)**

For a macro like this you will typically be calling it from another macro and specifying the arguments in the macro that is calling it.  But if you just want to test the macro that contains arguments, you can use the Immediate Window to call it.

This is great for writing and debugging code.  You might not want to run the entire stack of procedures (macros) in the code, but you can use the Immediate Window to only call that specific macro you’re working on.

The example above shows how you can specify the arguments after the macro name.  For arguments that are string variables (text), you will need to wrap the variable in quotation marks.

As you can see in the image, the intellisense is available in the Immediate Window, which makes it easy to specify the arguments for the macro.

The code in the image above is included in the free sample file you can download below.