1. **What are comments and what is the importance of commenting in any code?**

Sometimes when the code alone doesn't provide context or clarify intent, the developer may write extra descriptions. These descriptions are called code comments. Code comments enhance readability. They facilitate code reviews, refactoring, and maintenance.

Commenting involves placing Human Readable Descriptions inside of computer programs detailing what the Code is doing. Proper use of commenting can make code maintenance much easier, as well as helping make finding bugs faster. Further, commenting is very important when writing functions that other people will use.

1. **What is Call Statement and when do you use this statement?**

The CALL statement transfers control from one object program to another within the run unit. The program containing the CALL statement is the calling program; the program identified in the CALL statement is the called subprogram.

1. **How do you compile a code in VBA? What are some of the problems that you might face when you don’t compile a code?**

To compile VBA code, you need to follow these steps:

* Open the Visual Basic Editor in Excel by pressing Alt + F11.
* In the editor, select the project or module that you want to compile.
* Click on the Debug menu and select the Compile VBA Project option or Compile VBA Code option.

If there are any syntax or compile errors in your code, the editor will highlight them, and you will need to fix them before you can compile the code successfully.

Some of the problems that you might face if you don't compile your VBA code include:

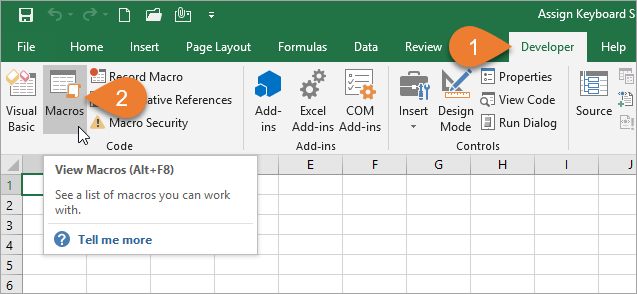
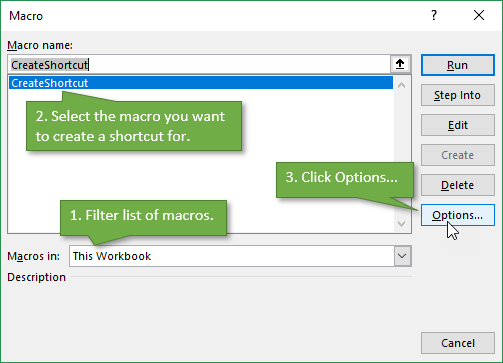
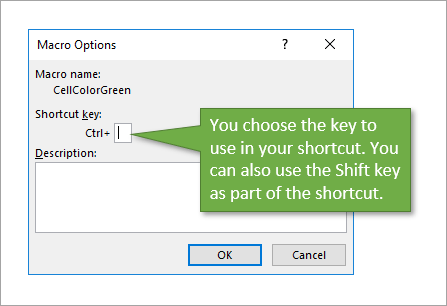
* Run-time errors: If there are syntax or logical errors in your code, you may encounter run-time errors when you run your program.
* Performance issues: Unclean or inefficient code can lead to slow execution and cause performance issues.
* Debugging difficulties: It can be more challenging to locate and fix errors if the code is not compiled, making the debugging process longer and more complex.
* Code maintenance difficulties: Uncompiled code may make it more challenging to maintain and update your codebase, particularly if there are numerous modules or projects.

1. **What are hotkeys in VBA? How can you create your own hotkeys?**

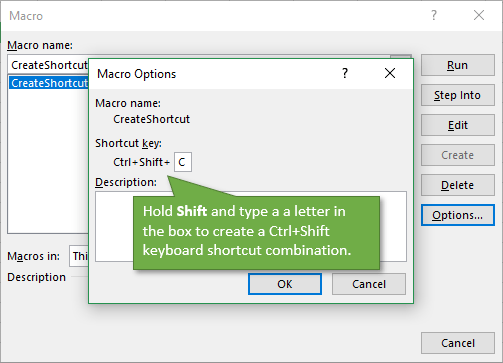
Assigning keyboard shortcuts to simple or complex macros can help you work faster in Excel.  This is especially true if you have to perform the same actions repeatedly.  In this post, we look at two popular ways to create shortcut keys.  These include the Macro Options window and VBA code for the Application. On Key method.  I also explain the pros & cons of each method.

**1. The Macro Options Window: Shortcut Key**

We can use the Macro Options window in Excel to create a shortcut key to call the macro.  Here are the instructions on how to set it up.

1. Start by going to the Developer tab and clicking on the Macros button.  (If you don't see the Developer tab on your ribbon, you can add it using [these instructions](https://www.excelcampus.com/vba/enable-developer-tab/).) Alternatively, you can use the keyboard shortcut **Alt**+**F8**.  
   
2. After selecting the macro that you want to assign the shortcut to, click the Options button.  
   
3. In the Macro Options Window, you can create the shortcut you want by adding a letter, number, or symbol.  
   

Be careful **not to override an existing shortcut** that you frequently use, such as **Ctrl**+**C** to copy. One way to avoid doing this is by adding Shift to the shortcut to make it a bit more complex. In my example, I used **Ctrl**+**Shift**+**C**.

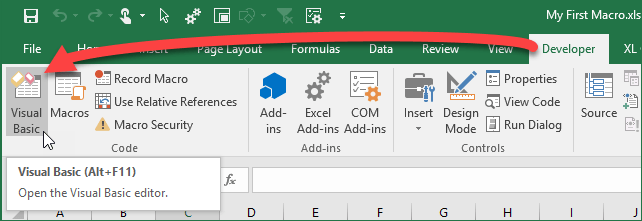


To **delete the shortcut**, simply repeat the process for accessing the Macro Options Window and then delete the character that you entered to create the shortcut.

**2. The Application.OnKey Method in VBA**

We can also use VBA code to create shortcut keys for macros.  The Application.OnKey method allows us to create and delete the shortcuts.  It also gives us more options and flexibility with our keyboard shortcuts.

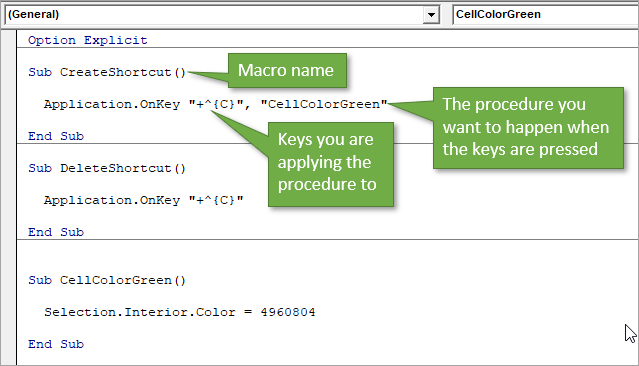
Start by accessing the VB Editor.  You can do this by clicking the Visual Basic button on the Developer tab, or pressing **Alt**+**F11**.



**Create Shortcuts with OnKey**

In the VB editor, we are going to write some simple code to **assign a macro to a keyboard shortcut**.

1. Create a new macro and name it CreateShortcut (or whatever you choose to name the procedure),
2. Add a new line and start it with the command **Application.OnKey** followed by a space.  
   The Application.OnKey method has two parameters for the Key and Procedure.  The **Key** is the keyboard shortcut combination represented by key codes.  The **Procedure** is the name of the macro that will be called when the key combination is pressed.  
   Both parameters are enclosed in quotation marks.
3. In my example I use “+^{C}” for the Key parameter. The + is the code for **Ctrl**, the ^ is code for **Shift**, and the **C** key is enclosed in curly brackets (or braces).  How are you supposed to know the code for each key? Microsoft has [this helpful document](https://docs.microsoft.com/en-us/office/vba/api/excel.application.onkey), which contains a complete list.
4. Following this code, you are going to name the procedure that you want to assign to that combination of keys. In this case, we want the key combination to run the macro called “CellColorGreen”.



**Delete Shortcuts with OnKey**

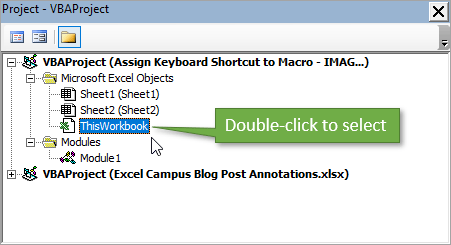
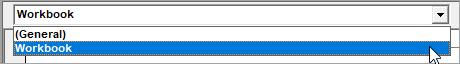
As you can also see in the image above, the **code to delete this process** is simple.  I've typed it just below the section for creating the shortcut.  Instead of “CreateShortcut” we will call it “DeleteShortcut” and we remove the procedure name (“CellColorGreen”) from the code.  The absence of a procedure tells Excel not to assign an action to that combination of keyboard strokes.

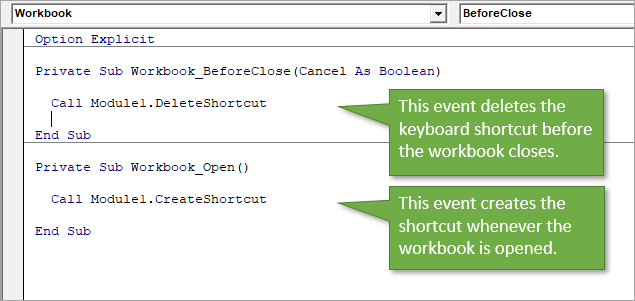
This also resets the key combination to any native Excel keyboard shortcuts.  For example, if we were using **Ctrl**+**C** instead, the keyboard shortcut would revert back to performing the Copy action when **Ctrl**+**C** is pressed.

Both the Create and Delete macros can have**multiple lines of code** with the OnKey method.  This allows you to setup different shortcuts for different macros all at the same time.

**Automate OnKey with Events**

However, you can actually **automate** this by using the Workbook\_Open and Workbook\_BeforeClose events.  Here are instructions on how to set it up (see the video above for more details).

1. In the VB Editor, double-click on ThisWorkbook in the Project Window.  
   
2. Choose **Workbook** in the drop-down box.  
   
3. That will add the **Workbook\_Open event**. Add a line of code to call the macro that you've created. In our case, the code would read “Call Module1.CreateShortcut” (without the quotation marks).  You can also add an event to delete the macro anytime you close the workbook.  Just choose **BeforeClose** in the drop-down on the right and call the same macro.



**5. What are the shortcut keys used to**

a. Run the code =  **Ctrl + Alt + N**.

b. Step into the code = F8

c. Step out of code = **Shift+F11**

d. Reset the code = **Alt + F4**