

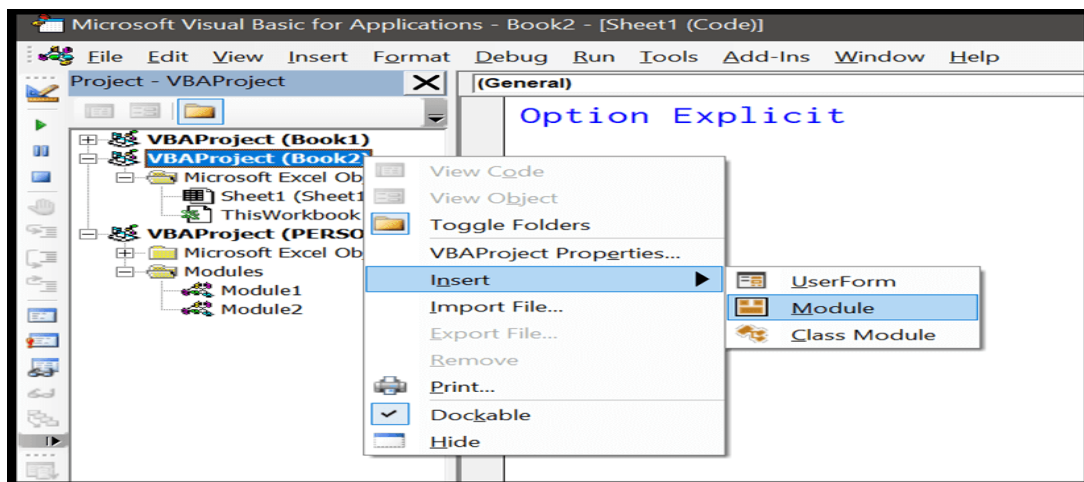
## **1. What are modules in VBA and describe in detail the importance of creating a module?**

VBA module is a “.bcf” extension file that holds the code in the visual basic editor. Each module has its own code window where you can write. You can insert a new module, delete, backup, and import it. In simple words, it’s like a text file that you open in the notepad.

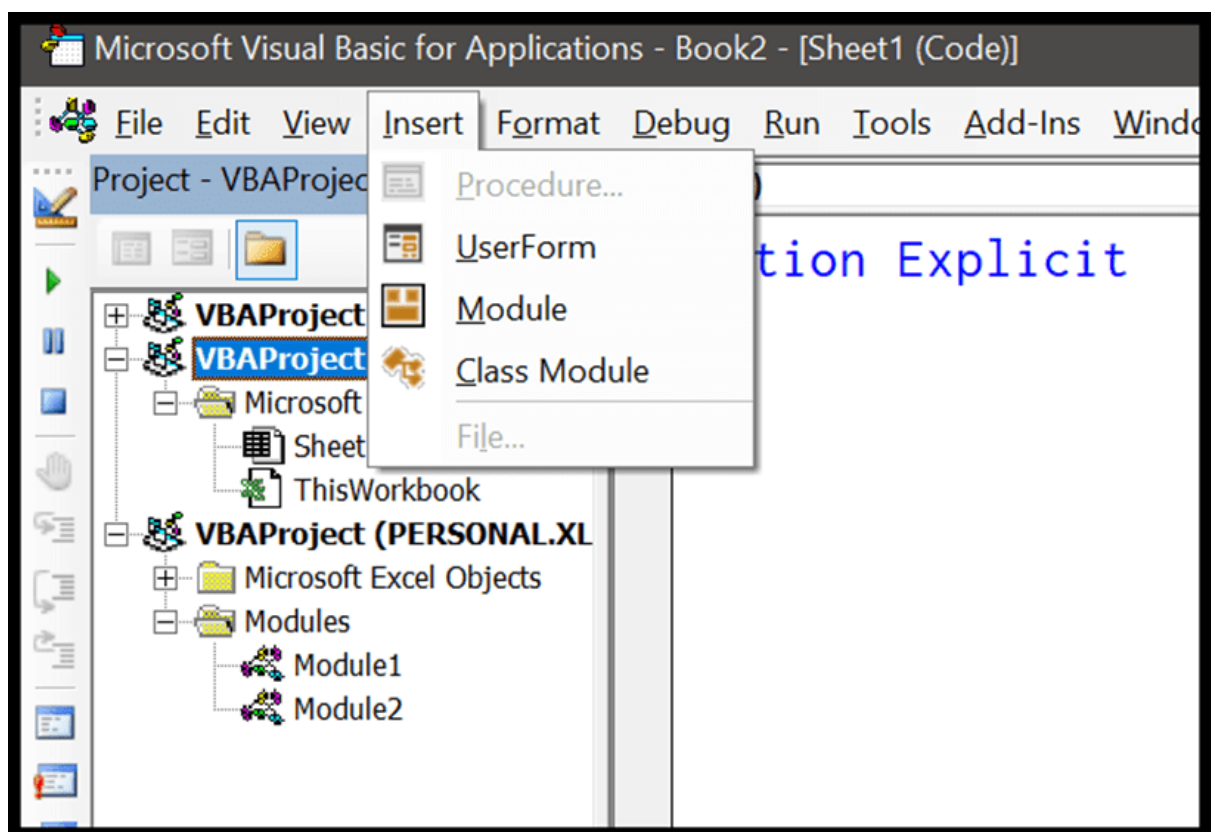
In this tutorial, we will learn all about using a module in VBA.

### **Insert a VBA Module**

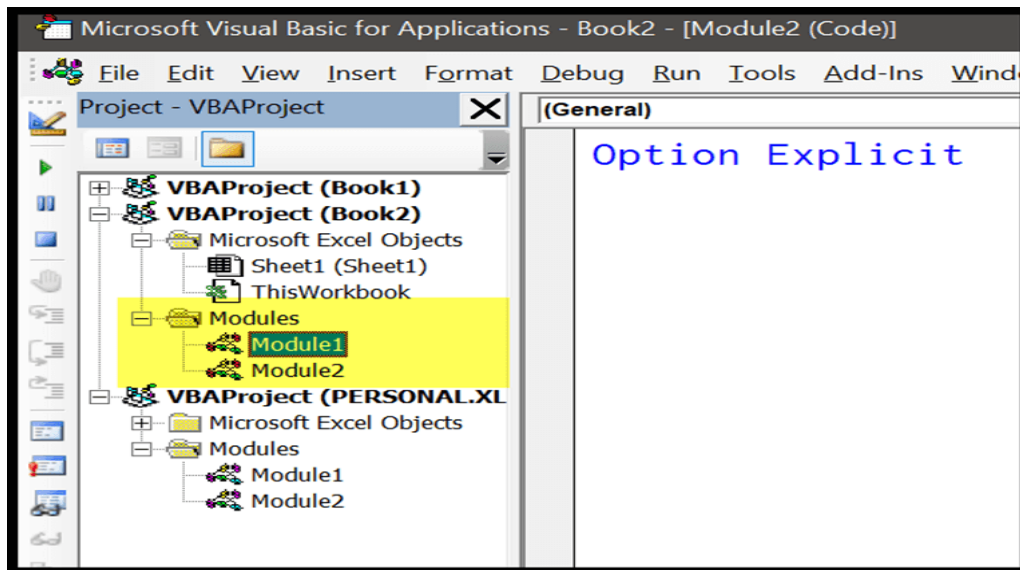
1. First of all, open the Project Window if it’s not there already.
2. Next, right-click on the project (Excel File) in which you want to insert a module.
3. After that, go to the Insert and click on the “Module”.
4. Once you do that, you will instantly get a new module and can open its code window to write the code.



You can also go to the insert menu and then the module to insert it.

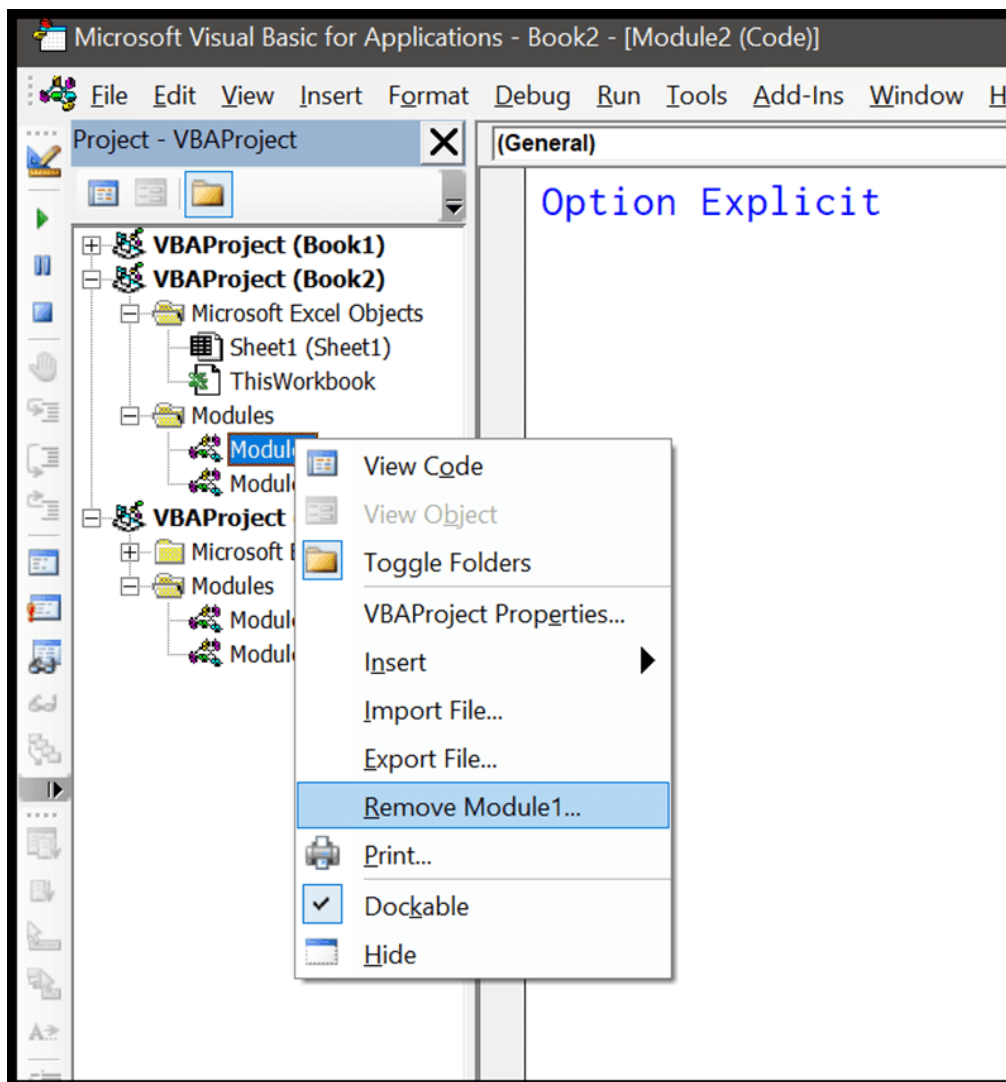


When you insert a new module, VBA creates a separate folder for all the modules that you have in the project.

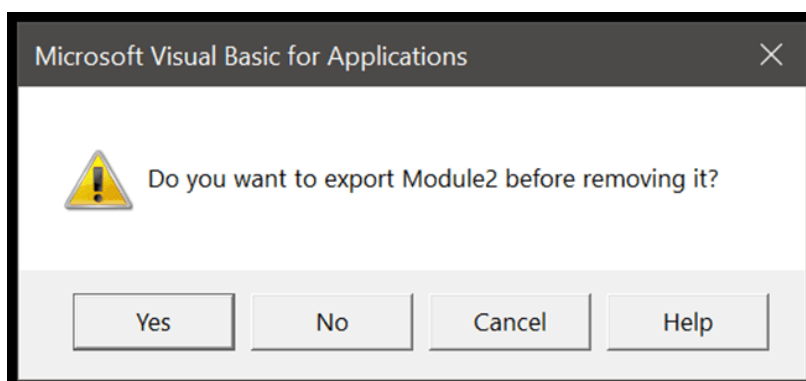


### Delete a VBA Module

1. Click on the project from which you want to delete a module.
2. Now right-click on the module that you want to delete and click "Remove".
3. After that, you need to confirm if you wish to back up the module before deleting it.
4. In the end, click on "Yes" to delete it.



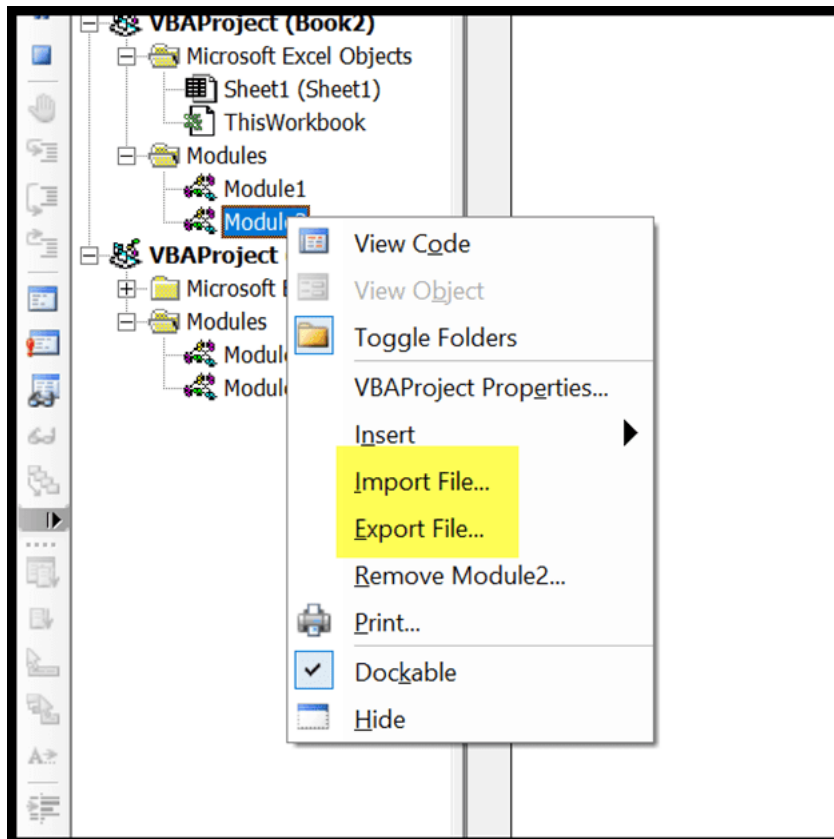
here's one thing that you need to know here when you click on the remove option, it asks you to back up your module before removing it (if you want).



It's a smart way to remind you to get the backup of the module.

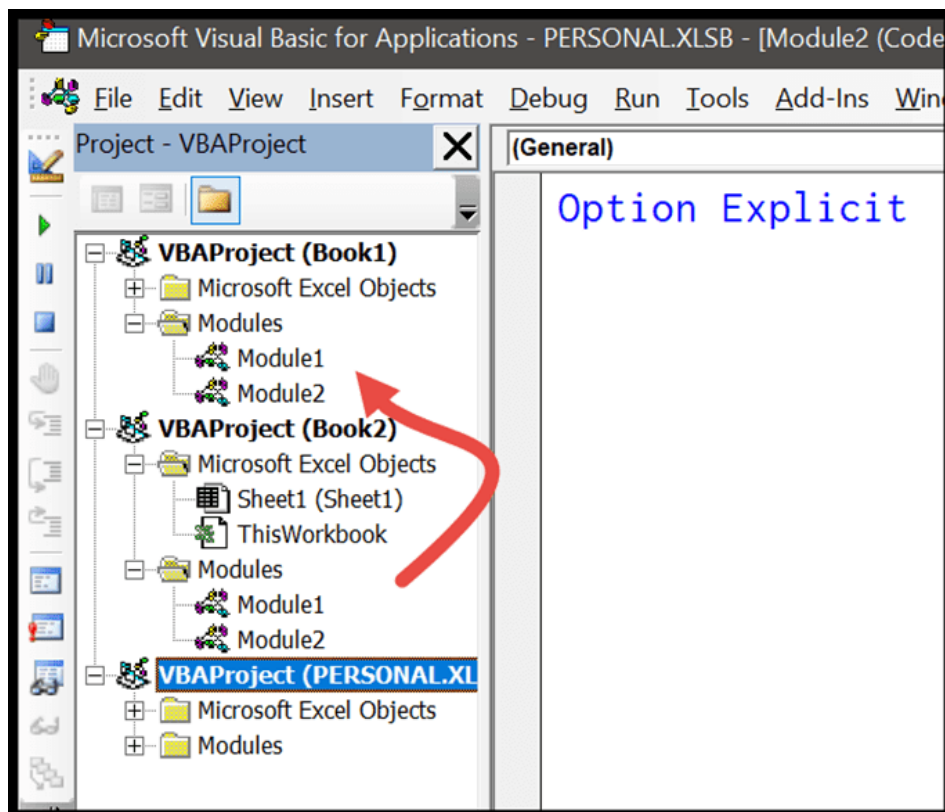
## Export and Import a VBA Module

You can also import and export a module from one Excel file to another instead of copy-pasting the codes. This is one of the best ways to share your codes with others. When you right-click on a module you have both options there.



As I said earlier, you can save a module as a “.bcf” file and import it in the future or some other Excel file.

**Quick Tip:** If you want to copy an entire module to a different project whose Excel file is open at the time. Drag and Drop that module to the project where you want to copy it.



## 2. What is Class Module and what is the difference between a Class Module and a Module?

When you insert a new module, you can see an option to insert a class module. But there's a slight difference between both modules. ***As you have understood all about the standard modules, class modules are special modules that can help you create your custom objects.*** You can also define methods, properties, and events for those objects. And when you create a new object from the class module, you can refer to it from the standard module as well.

### **3. What are Procedures? What is a Function Procedure and a Property Procedure?**

A property procedure is a series of Visual Basic statements that manipulate a custom property on a module, class, or structure. Property procedures are also known as *property accessors*.

Visual Basic provides for the following property procedures:

- A Get procedure returns the value of a property. It is called when you access the property in an expression.
- A Set procedure sets a property to a value, including an object reference. It is called when you assign a value to the property.

You usually define property procedures in pairs, using the Get and Set statements, but you can define either procedure alone if the property is read-only (Get Statement) or write-only (Set Statement).

You can omit the Get and Set procedure when using an auto-implemented property. For more information, see Auto-Implemented Properties.

You can define properties in classes, structures, and modules. Properties are Public by default, which means you can call them from anywhere in your application that can access the property's container.

For a comparison of properties and variables, see Differences Between Properties and Variables in Visual Basic.

#### **Declaration syntax**

A property itself is defined by a block of code enclosed within the Property Statement and the End Property statement. Inside this block, each property procedure appears as an internal block enclosed within a declaration statement (Get or Set) and the matching End declaration.

The syntax for declaring a property and its procedures is as follows:

VBCopy

```
[Default] [Modifiers] Property PropertyName[(ParameterList)] [As  
DataType]
```

```
    [AccessLevel] Get
```

```
        ' Statements of the Get procedure.
```

```
        ' The following statement returns an expression as the  
property's value.
```

```
        Return Expression
```

```
    End Get
```

```
    [AccessLevel] Set[(ByVal NewValue As DataType)]
```

```
        ' Statements of the Set procedure.
```

```
        ' The following statement assigns newvalue as the property's  
value.
```

```
        LValue = NewValue
```

```
    End Set
```

```
End Property
```

```
' - or -
```

```
[Default] [Modifiers] Property PropertyName [(ParameterList)] [As  
DataType]
```



The Modifiers can specify access level and information regarding overloading, overriding, sharing, and shadowing, as well as whether the property is read-only or write-only. The AccessLevel on the Get or Set procedure can be any level that is more restrictive than the access level specified for the property itself. For more information, see Property Statement.

## Data Type

A property's data type and principal access level are defined in the Property statement, not in the property procedures. A property can have only one data type. For example, you cannot define a property to store a Decimal value but retrieve a Double value.

## Access Level

However, you can define a principal access level for a property and further restrict the access level in one of its property procedures. For example, you can define a Public property and then define a Private Set procedure. The Get procedure remains Public. You can change the access level in only one of a property's procedures, and you can only make it more restrictive than the principal access level. For more information, see How to: Declare a Property with Mixed Access Levels.

## **Parameter declaration**

You declare each parameter the same way you do for Sub Procedures, except that the passing mechanism must be ByVal.

The syntax for each parameter in the parameter list is as follows:

VBCopy

[Optional] ByVal [ParamArray] parametername As datatype

If the parameter is optional, you must also supply a default value as part of its declaration. The syntax for specifying a default value is as follows:

VBCopy

Optional ByVal parametername As datatype = defaultvalue

### **Property value**

In a Get procedure, the return value is supplied to the calling expression as the value of the property.

In a Set procedure, the new property value is passed to the parameter of the Set statement. If you explicitly declare a parameter, you must declare it with the same data type as the property. If you do not declare a parameter, the compiler uses the implicit parameter Value to represent the new value to be assigned to the property

### **5. What is a sub procedure and what are all the parts of a sub procedure and when are they used?**

A Sub procedure is a series of Visual Basic statements enclosed by the Sub and End Sub statements. The Sub procedure performs a task and then returns control to the calling code, but it does not return a value to the calling code.

Each time the procedure is called, its statements are executed, starting with the first executable statement after the Sub statement and ending with the first End Sub, Exit Sub, or Return statement encountered.

You can define a Sub procedure in modules, classes, and structures. By default, it is Public, which means you can call it from anywhere in your application that has access to the module, class, or structure in which you defined it. The term *method* describes a Sub or Function procedure that is accessed from outside its defining module, class, or structure. For more information, see Procedures.

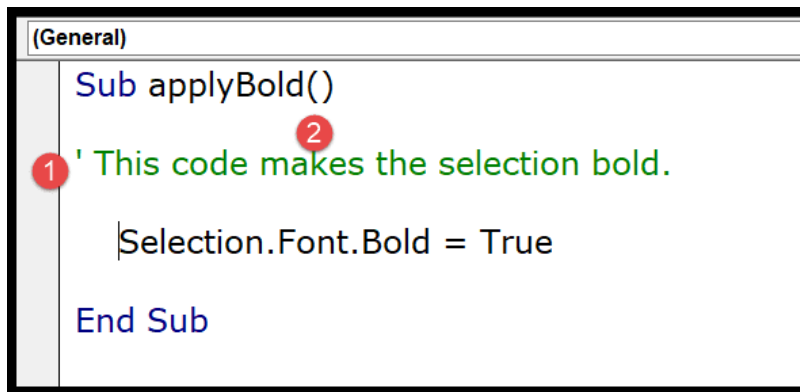
A Sub procedure can take arguments, such as constants, variables, or expressions, which are passed to it by the calling code.

## **6. How do you add comments in a VBA code? How do you add multiple lines of comments in a VBA code?**

A VBA COMMENT is a green line of text that helps you to describe the written code. In simple words, a comment is a line of text which is not a code and VBA ignores it while executing the code. It's a good practice (I'd say one of the best) to add comments in your VBA codes.

(Video) Understanding VBA Comments

How to Add a Comment in a VBA Code (Macro)



## Advantages of using a Comment

As I said commenting in a VBA code is one of the best practices and there are a few benefits that come with it.

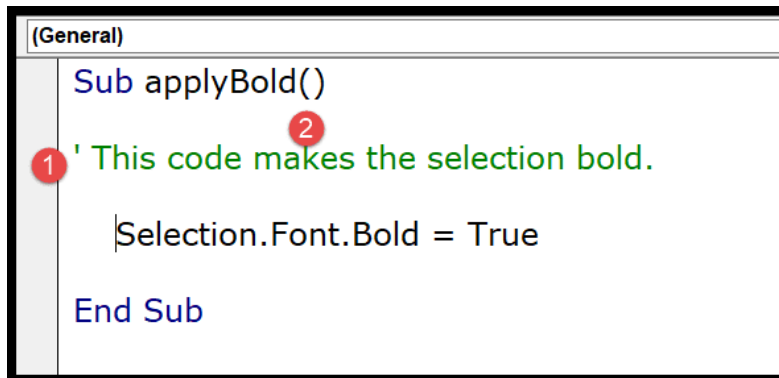
- **Helps you to Document your Work:** You can use a comment to describe how code works, which can help you in the future to recall it easily or any other user.
- **Track the Changes:** If some codes need you to change them frequently you can use comments to track or record changes within the code.
- **Describe a Function Procedure:** When you write a procedure you can add a comment at the starting to describe the purpose of this procedure and how it works.
- **Describe a Variable:** Variables are one of the most important things that you need to use while writing a VBA code and you can use a comment to describe a variable.
- **Debug Your Code:** You can use VBA comments to debug the code by converting code lines into comments for testing.

## Add a Comment in a VBA Code

Steps you need to follow to add a comment in a VBA code:

1. First, click on the line where you want to insert the comment.

2. After that, type an APOSTROPHE using your keyboard key.
3. Next, type the comment that you want to add to the code.
4. In the end, hit enter to move to the new line and the comment will turn green.



The screenshot shows a VBA code editor window with a tab labeled '(General)'. The code is as follows:

```
Sub applyBold()  
1 ' This code makes the selection bold.  
    Selection.Font.Bold = True  
End Sub
```

Red circles with numbers 1 and 2 are placed over the apostrophe and the comment text, respectively, to indicate the steps for adding a comment.

The moment you do this the entire line of the code will turn green which means that line is comment now.

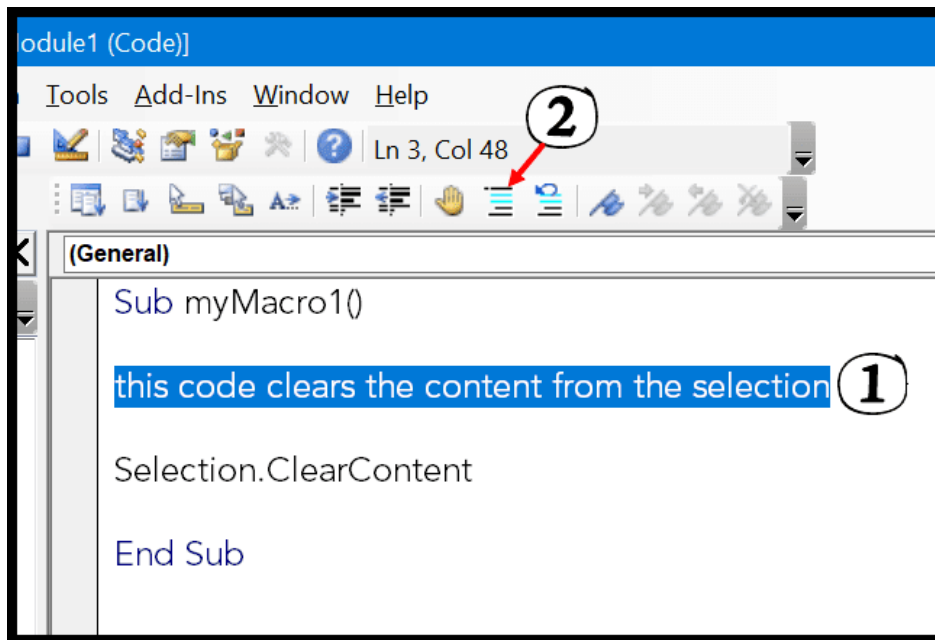
If you look at the below code where I have used a comment to add a description of the procedure.

So you simply need to add an apostrophe before turning it into a comment and VBA will ignore it while executing the code.

Use the Comment/UnComment Button from the Toolbar

The second method is to use the comment block button from the toolbar. This button simply adds an apostrophe at the start of the line.

So, to use this button, first of all, you need to select the line of the code and then click on the button.



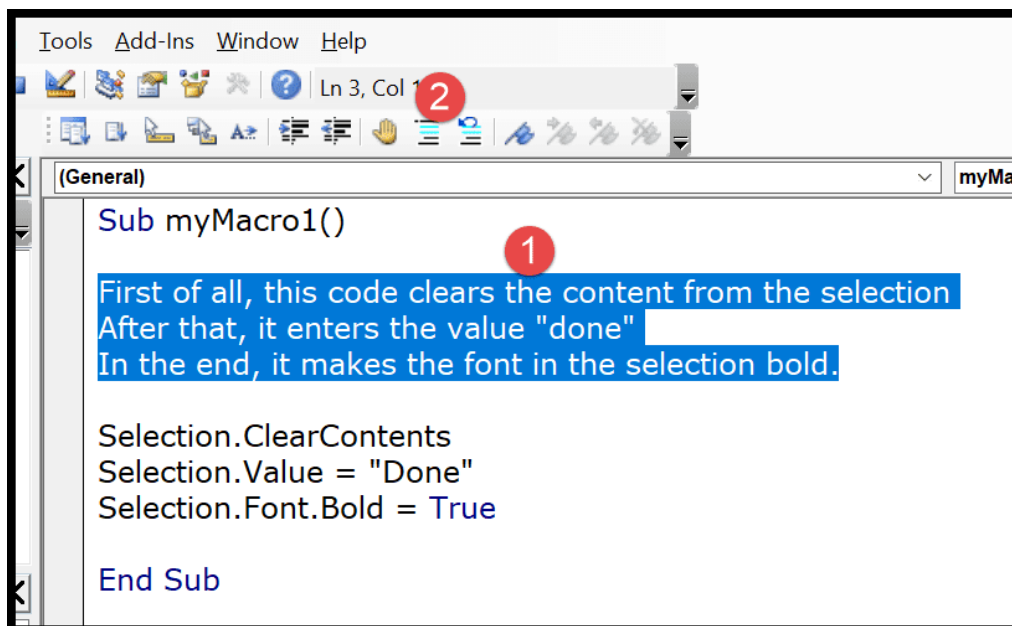
Next to the Comment button, there's one more button "Uncomment" which you can use to uncomment a line (This button simply removes the apostrophe from the line of the code).

### Enter a Multi-Line VBA Comment

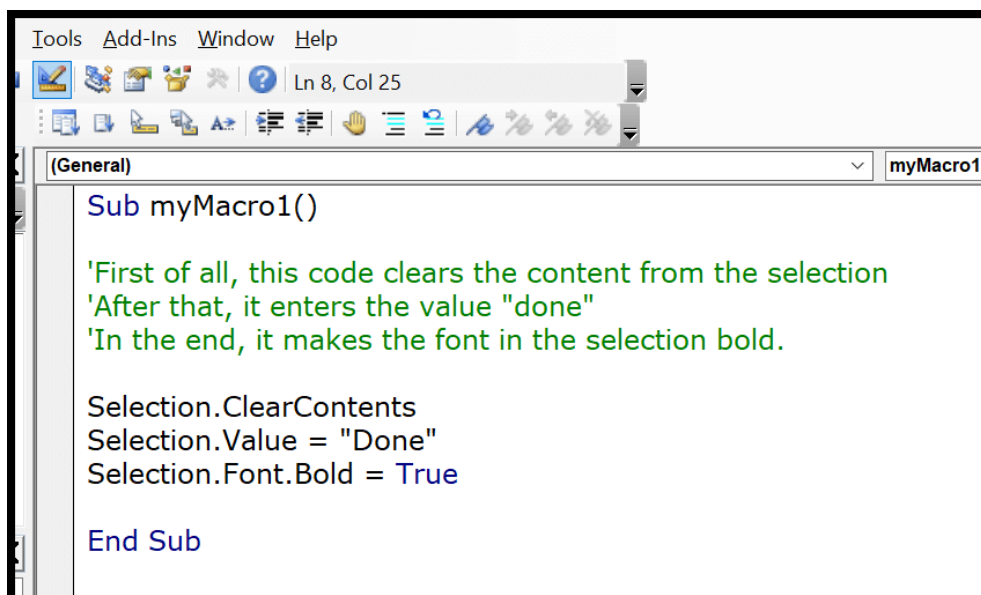
There could be a situation where you need to enter a comment in multiple lines, like a block of the comments.

**But here is** one thing which you need to note down, every line of comment needs to start with an apostrophe, so if you want to add multiple lines of comments every line should have an APOSTROPHE.

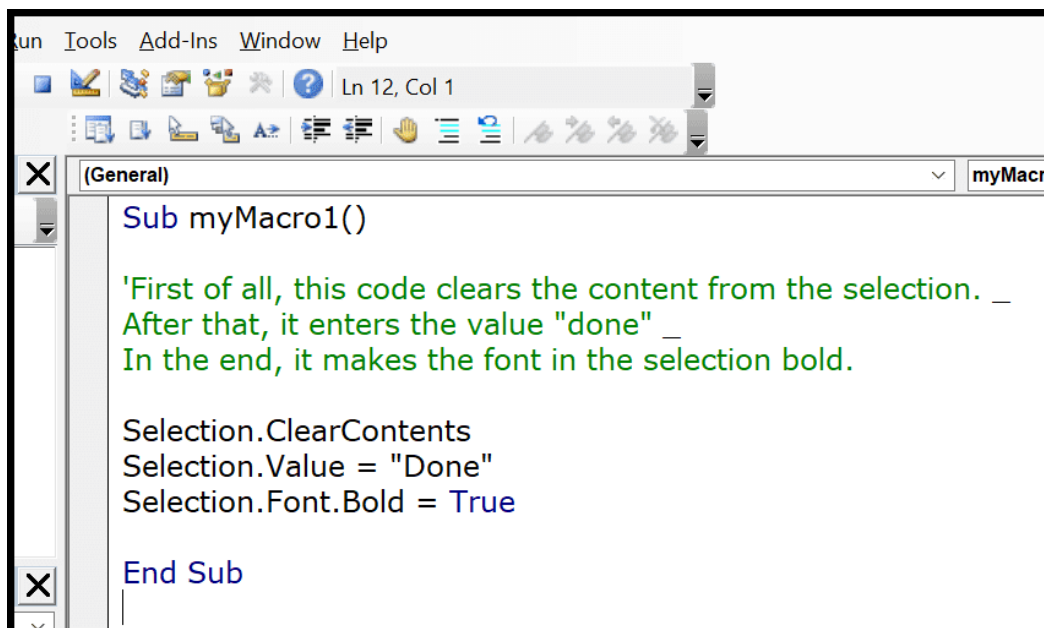
The easiest way is to select all the lines and then use the comment button from the toolbar or you can also add an APOSTROPHE at the starting of each line.



The moment you click the comment button it will convert all the lines into a multi-line comment block.



Update: There's one thing that I have discovered recently if you have a block of comment line (continuous), you can use a line continuation character (an underscore must be immediately preceded by a space).

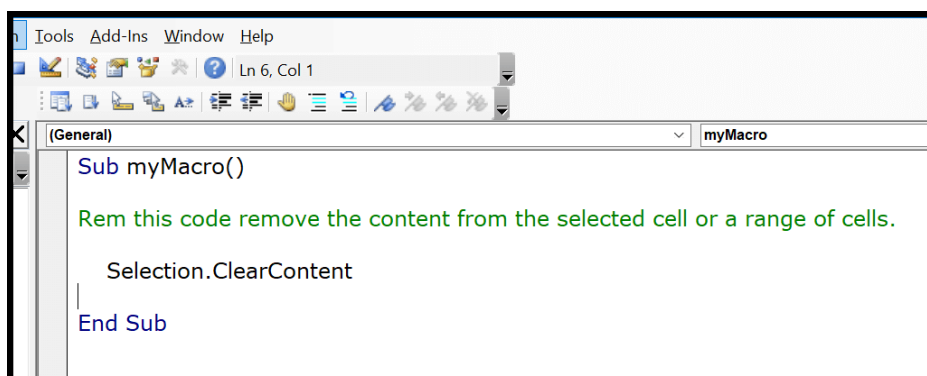


In the above example, I have used an apostrophe only at the start of the first line of the comment, the rest two-line don't have an apostrophe but I have used line continuation character to give a line break at the end of the first line and the second line.

### Use the "REM" Keyword to Insert a Comment in VBA

This is the third way to insert a comment into a VBA code. Well, this is not a popular way but still, you can use it. So instead of using an apostrophe, you can use the keyword REM at the starting of a comment line.

REM stands for remarks.



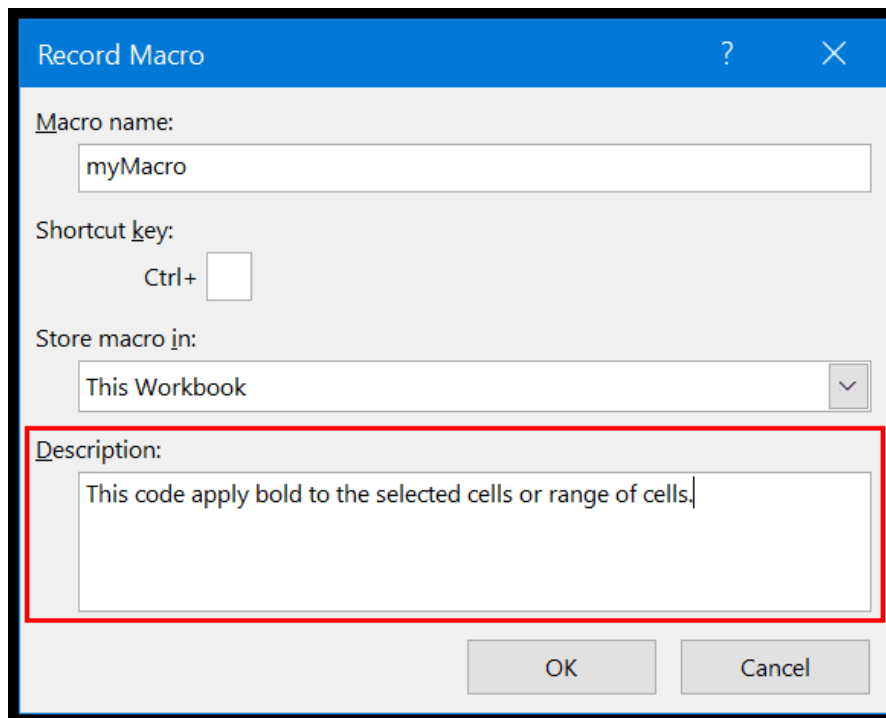


Now, in the above example, I have used “REM” at the start of the line of the code and then the line of the code. But, there’s no button to add REM, you have to type it.

### Comments while Recording a Macro with Macro Recorder

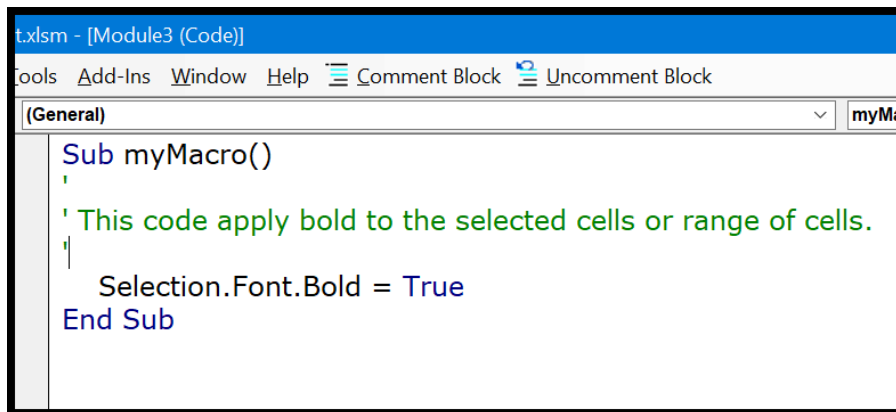
When you record a macro code using the macro recorder, you get an option to add a description before you record it.

So when you open the record macro dialog box, there is a “Description” input box where you can add your comment and then start recording.



And when you open the VBE to see the recorded code, you can see the code which you have added as a comment.

When you add a comment while recording a macro, VBA adds a few apostrophes as blank comments.



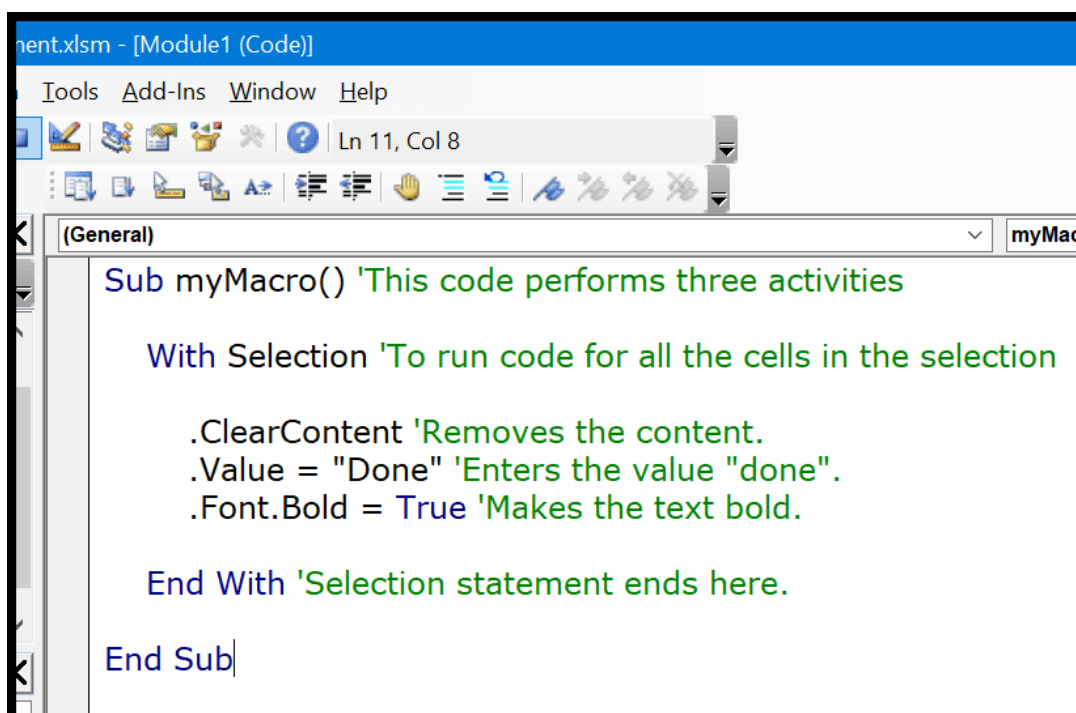
The screenshot shows the VBA editor window titled 't.xlsm - [Module3 (Code)]'. The menu bar includes 'Tools', 'Add-Ins', 'Window', 'Help', 'Comment Block', and 'Uncomment Block'. The 'General' tab is selected, and the macro name 'myMacro' is visible in the top right. The code is as follows:

```
Sub myMacro()  
' This code apply bold to the selected cells or range of cells.  
    Selection.Font.Bold = True  
End Sub
```

### Enter a Comment in the Same Line

This is a quite smart way to use a comment in the same line where you have written code.

In the below example you can see that I have three lines of code, where I have added comments after each line to describe it.



The screenshot shows the VBA editor window titled 'tent.xlsm - [Module1 (Code)]'. The menu bar includes 'Tools', 'Add-Ins', 'Window', and 'Help'. The 'General' tab is selected, and the macro name 'myMacro' is visible in the top right. The code is as follows:

```
Sub myMacro() 'This code performs three activities  
    With Selection 'To run code for all the cells in the selection  
        .ClearContent 'Removes the content.  
        .Value = "Done" 'Enters the value "done".  
        .Font.Bold = True 'Makes the text bold.  
    End With 'Selection statement ends here.  
End Sub
```

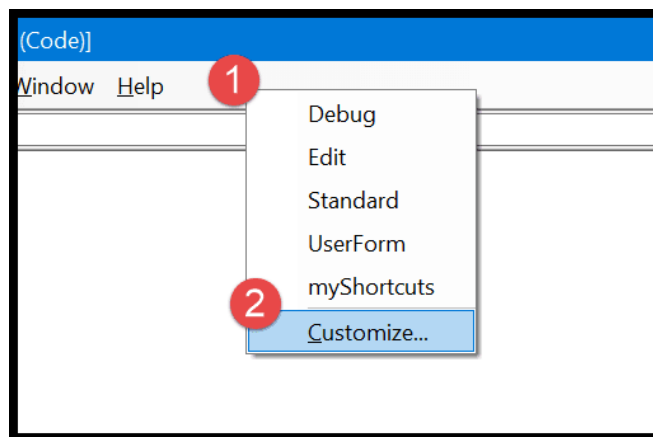
So once you complete a line of code you can use enter a comment after that in the same line.

## Shortcut Key for Add a Comment

Truly speaking there's no (by default keyboard shortcut) to use to insert a comment. But thanks to Gaurav, I have found a way to create a shortcut key to insert an apostrophe.

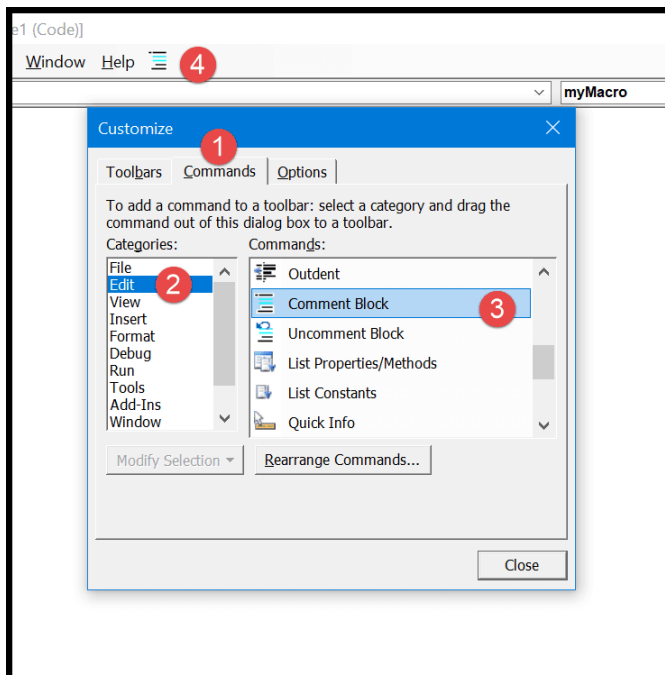
Follow the below steps:

- First of all, right-click on the toolbar and then click on the

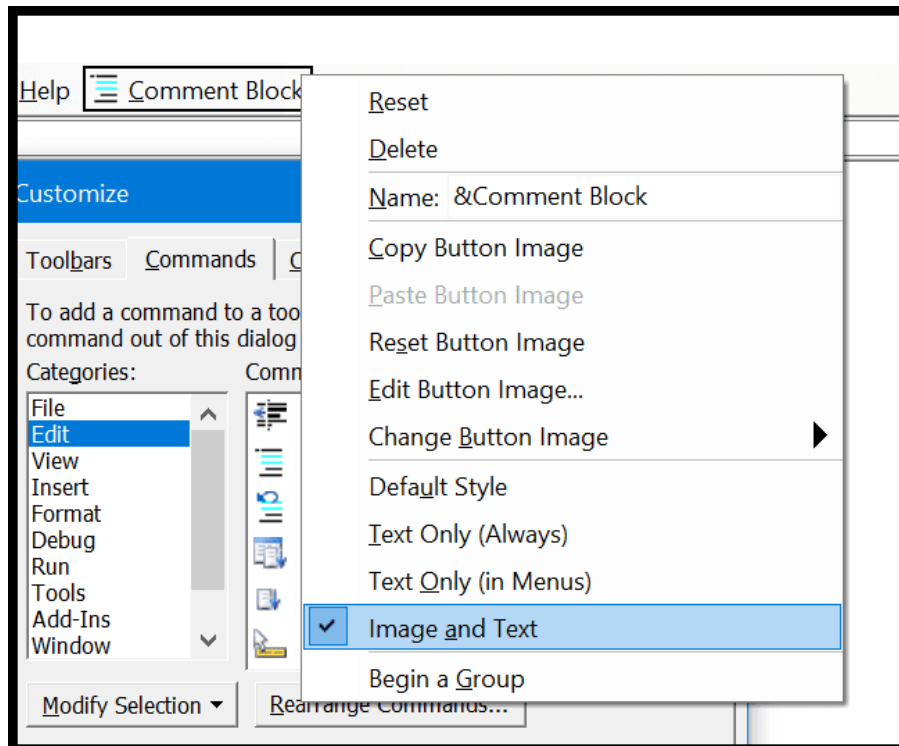


customize option.

- In the customize dialog box, go to the Command-Tab → Edit → select the comment block, and drag it to the toolbar.



- After that select the comment block icon, like below and right-click on it, and then enter “&” before the name. So it should
- Now, again right-click on the button and select the “Image and Text” and after that come back to the visual basic editor.

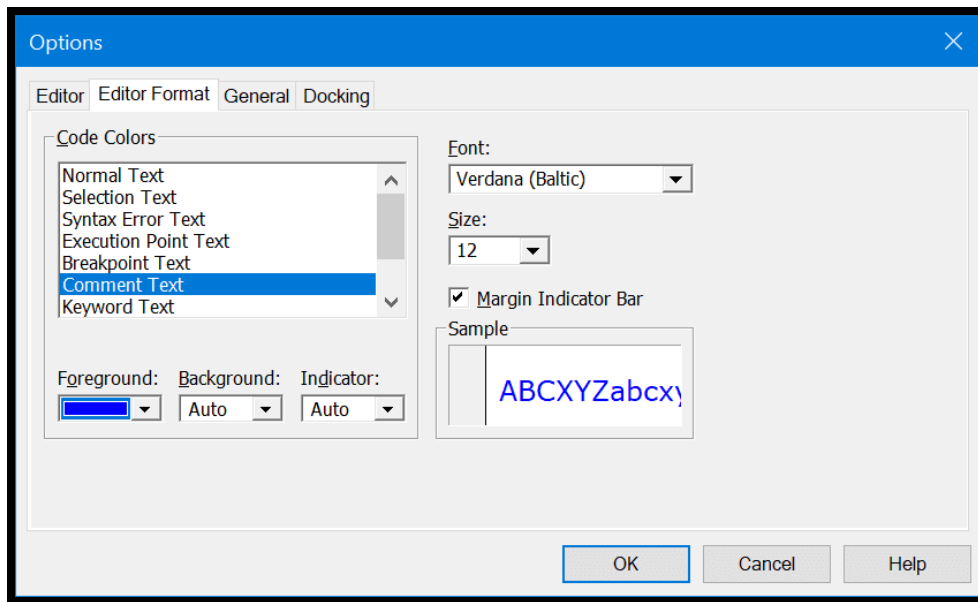


Now you can convert a line into a comment by using the shortcut key Alt + C.

And if you want to create a shortcut key for the uncomment button you can simply use the above steps to add the uncomment button to the toolbar and the shortcut key for it will be Alt + U.

### Change the Format of the Comment

VBA gives you an option to change the format of the comment if you want to do so. From the Tools → Options → Editor Format, click on the comment text.



As you can see, I have changed the color of the comment text from green to blue. Now all the comments which I have in the code window are in blue color.

