

## Sample Code

```
' To create a function
Function <function_name> ([arg1[, arg2 ...]]) [As <Type>]
End Function
' To create a subroutine
Sub <sub_name> ([arg1[, arg2 ...]])
End Sub
' Get the value of a cell from the active sheet
Range("A1").Value
' Get the value of a cell from a particular sheet
Sheets("Sheet Name").Range("A1").Value
' Get the value of a cell from a particular sheet and wb
Workbooks("Wbk.xlsx").Sheets("Sheet").Range("A1").Value
' Making decisions
If <boolean expression> Then
    <code to execute>
[ElseIf
    <code to execute>]
[Else
    <code to execute>]
End If
' Repeating actions
For var = 1 To 10
    <code to execute>
Next
```

## Shortcut Keys

### Excel Shortcuts

Ctrl-1	Format Cells Dialog
Alt-F8	Run Macro
Alt-F11	Open VBA Editor
Alt,t,o	Open Excel Options

### VBA Editor Shortcuts

Ctrl-G	Open Immediate Window
Ctrl-R	Open Project Explorer Window
F4	Open Properties Window
F5	Run current macro
F8	Execute next line of program
Shift-F8	Execute next line of program (skip sub-functions)

\* Note: items in brackets ("[ ... ]") are optional.

## How can I find out how to do <insert question here> in VBA? (For example, how can I find out how to rename a sheet in VBA?)

If you are trying to do something that you haven't done before (like renaming a sheet with VBA), start off by recording a macro like we did at the beginning of the tutorial. Then hit Alt-F11 to open up the VBA editor and see what code was generated by the macro recorder. This will usually suffice to get you started.

## What are the variable types that I can use when declaring a variable?

The main types we went over include:

- Integer (for numbers with no decimals between -32,768 and 32,767);
- Long (numbers with no decimals between -2,147,483,648 to 2,147,483,647);
- Single (pretty big numbers with decimals);
- Double (really big numbers with decimals);
- Byte (numbers with no decimals between 0 and 255).

In addition to those note above, there are many more. Some of the more popular one's that you may be interested in include:

- String (holding character data like a name or a file path, etc.).
- Date
- Workbook or Worksheet
- Range (could be a single cell or multiple cells)

## What can VBA code do that a macro can not do? Is VBA really that powerful?

VBA code is many order of magnitudes more powerful and versatile than simple macros. We went over two examples in the tutorial that demonstrate this fact. The first was coloring every other row a different color. The second was the mortgage calculator that we created. In both instances, we needed to perform different tasks depending on whether certain things were true or not (e.g., did we color the last row red? If so, color this one blue). We also needed to repeat some task a certain number of times, but that number of times could vary (e.g., are we going to amortize the loan over 3 years or 10 years?). In both scenarios, we were faced with problems that macros simply could not handle - and yet this was only the tip of the iceberg in terms of what you can do with VBA.

## How can I learn more about VBA? Are there other resources or books you would recommend?

There are an enormous number of books that you can buy to help you expand your knowledge of VBA. Those by [John Walkenbach](#) are generally pretty good. And while the [O'Reilly book](#) is fairly dated, their content is typically top-notch. In addition to books, you can check out additional videos at our site - we publish a new one every three to four weeks. [Click here](#) for more details.