# I'm Sick And Tired Of Doing This! (Macros)

HelpMeExcel.com

James Tobin Consulting, LLC

## A Few Things Before We Get Started…

Q: What is a Macro?
A: A set of instructions, that when commanded, causes Excel to perform specific tasks.

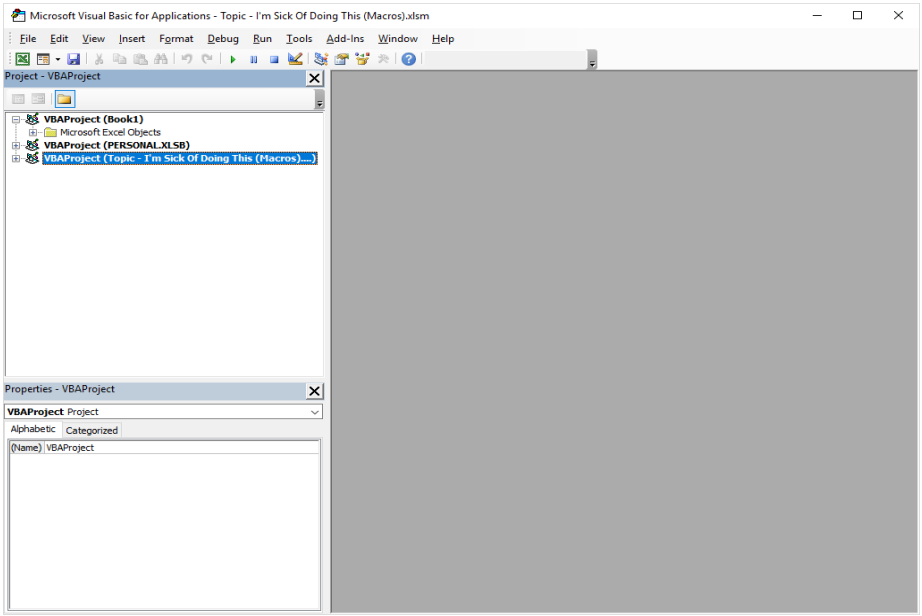
Q: Why do I need a Macro?
A: To minimize processing time and errors, to standardize output and to make yourself more valuable to the team (and your next employer!).
A: To save your life when you're sick and tired of doing something over and over again! Click one button and voilà, you're done!

Q: What is VBA?
A: VBA stands for Visual Basic for Applications and it's the programming language used in Excel macros.

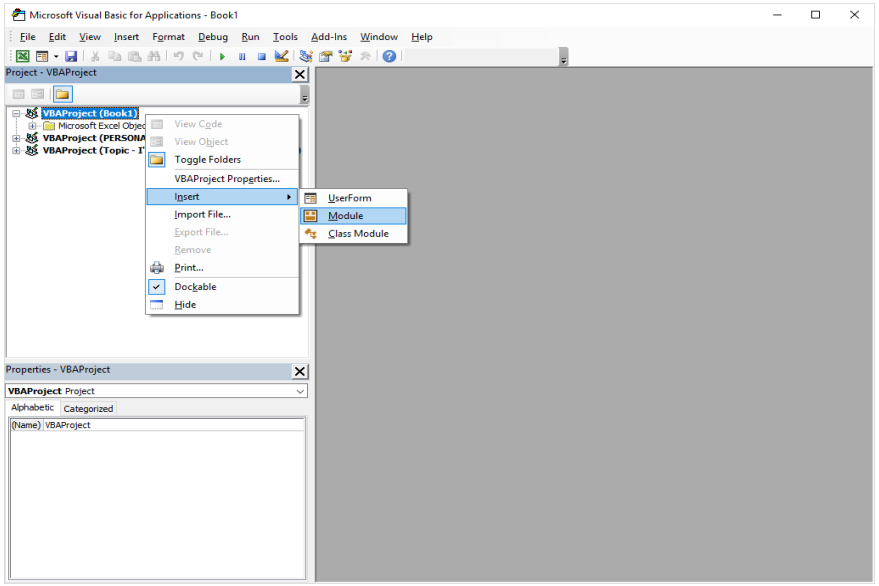
Q: Is it tough to learn?
A: Well… don't be intimidated. You can start with the Macro Recorder and grow into a full-fledged propeller head!

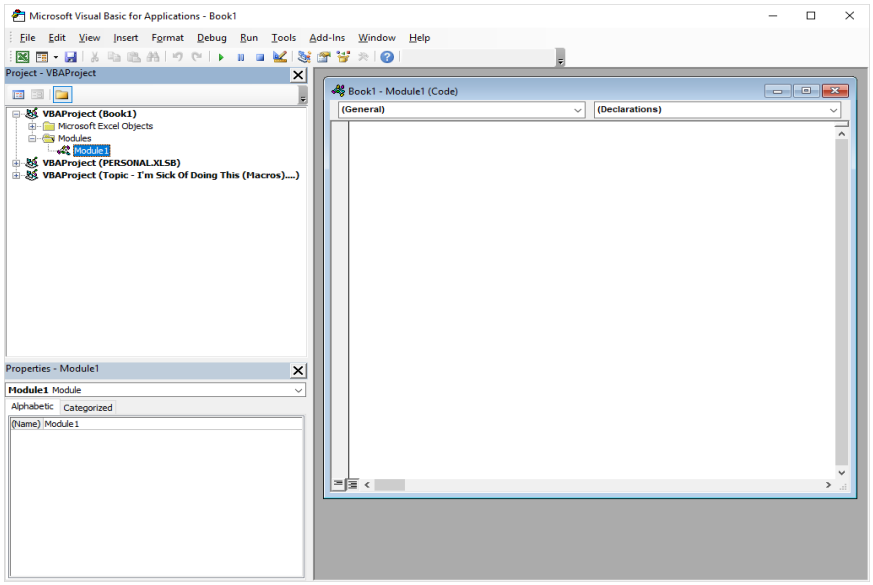
Q: How do I get started?
A: Save the current file as an xlsm (Macro Enabled) file. The Macro Enabled (xlsm) file ensures that any Modules containing VBA code will remain attached to the file.

Q: Now what…?
A: We'll record a macro later in this lesson, so for now, lets go to the VBA Editor to get started. The fastest way to get to the VBA Editor is to use Alt+F11



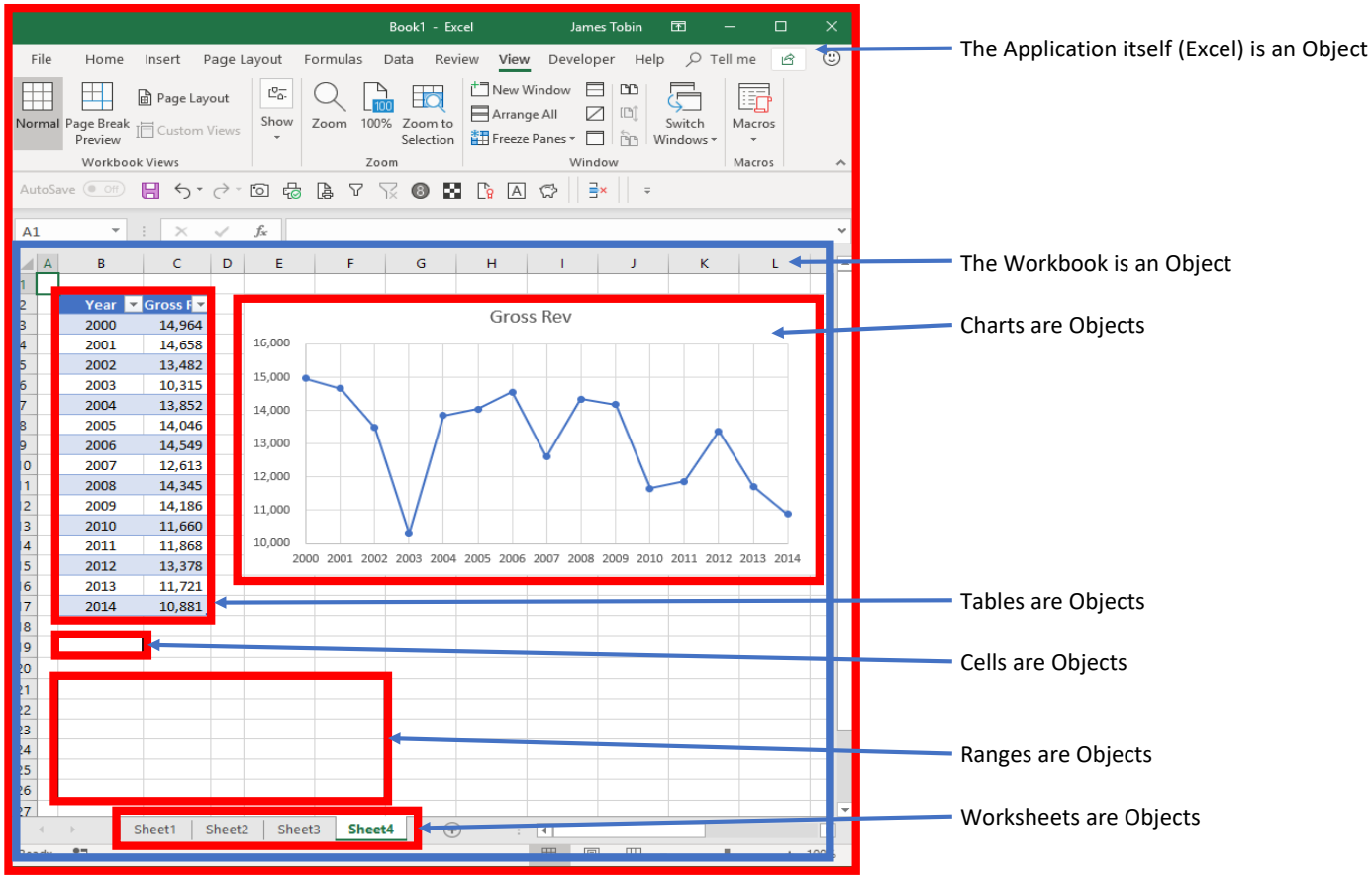
Q: What do I do next?
A: If you look at Book1, in the Project Window, you'll see that there is a sub-category for Objects. To create a macro you'll need to install a Module. To do that, right-click on Book1 and select Insert, then select Module. The new Module appears in the Code Window section of the VB Editor.



## Objects, Methods and Properties

Q: What is an Object?
A: Almost everything is an Object…




Q: What is a Method?
A: Typically, a Method is what you do to the object. For example, Clear, Cut, Copy, Paste, Count, Move, Select, Fill, Save, etc..

Q: What is a Property?
A: Typically, a Property is an attribute of the Object. You can either set the property or change the property. For example, Column, Row, Width, Value, etc..

Q: Is there an order of usage?
A: Typically, you'll test and/or change an attribute (Property) and then take action (Method) on/in the Object. For example, you'll test to see if the column width equals 10, if so, you'll ignore the method and exit the macro. If not, using a method, you'll resize the column width to 10, then exit the macro.

Basically, in Excel, we have Objects that we program with Properties and Methods.

## Let's Write a Macro…

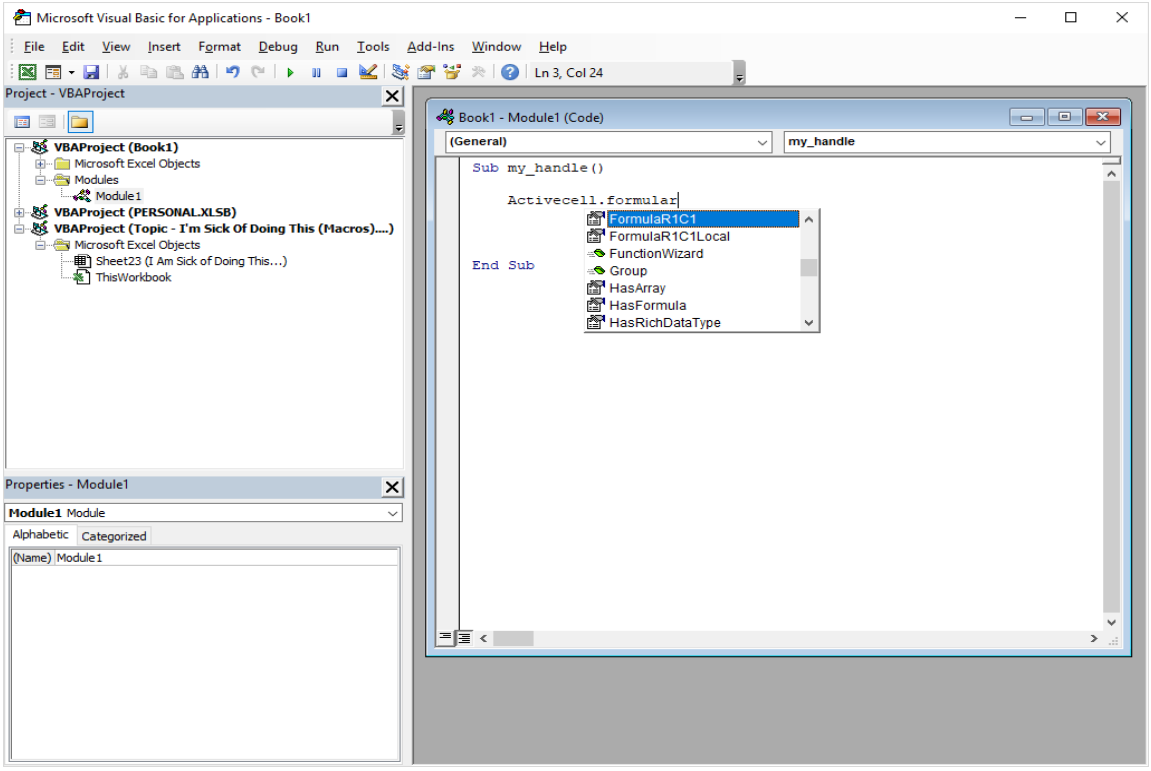
Let's jump right in, the water is perfect!

1 With the Module open in the Code Window, type "sub" and the macro's name, then hit enter. The name cannot have spaces or special characters (e.g. !@#$%^&*,.). You'll notice that the Visual Basic Editor placed parenthesis at the end of the Sub line and the words "End Sub" two rows beneath the Sub line.

2 Place a few hard returns (hitting the Enter key) after the sub line. In one of the newly created rows, after hitting the tab key, type Activecell.FormulaR1C1="My Name"

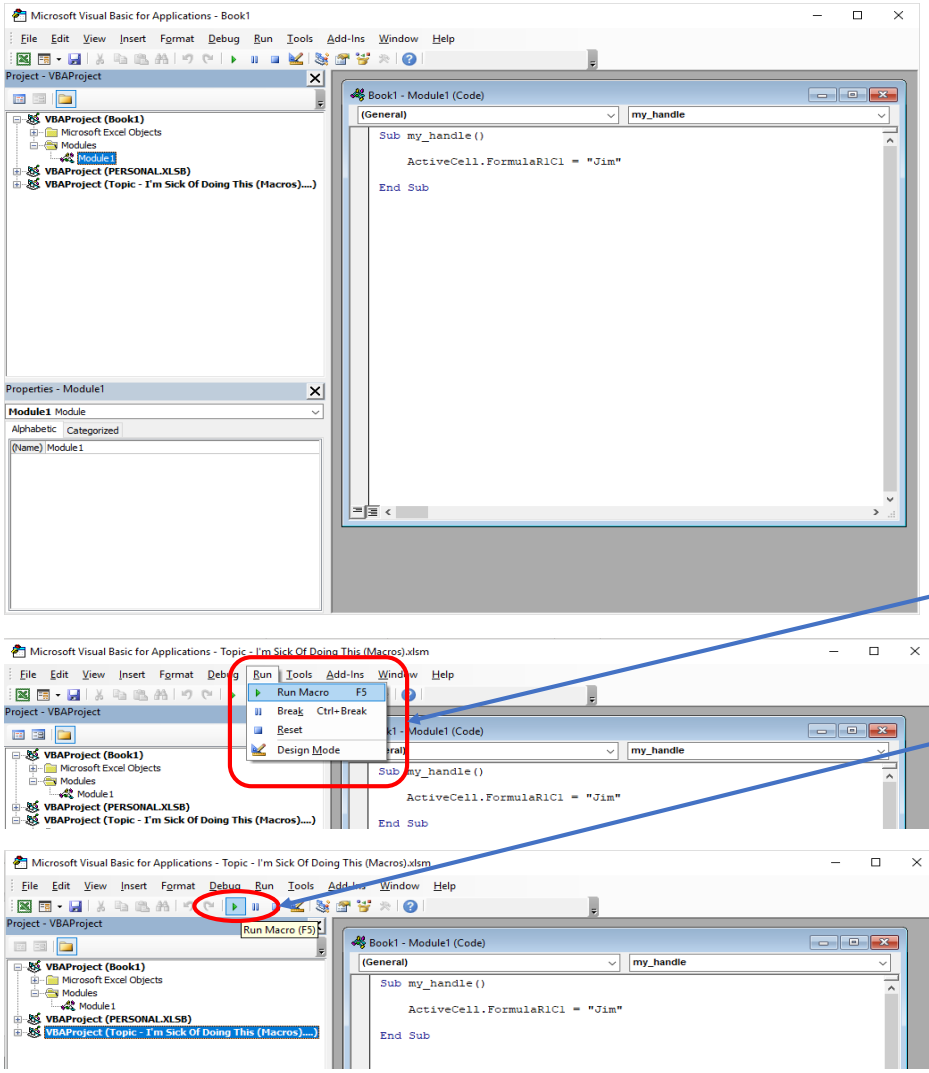
You probably noticed that upon placing the period after Activecell, the Visual Basic Editor opened a box with options. That box is called Intellisense and it allows you to choose the method/property you are allowed to perform for the specific object you've selected in that row of code. In this case, the object is the Activecell. There are many methods/properties you can select. As you begin typing a method/property, Intellisense displays those that are allowed and once you've identified the method/property you need, hit either Tab or Enter and the Visual Basic Editor will fill in the rest.

You'll notice there are icons to the left of the methods/properties in the Intellisense box. The hand pointing at, what looks to me to be, a list is a Property and the flying green brick is a Method.



3 The final product!

To run the macro you can hit the F5 key or you can select "Run" from the tabs at the top of the Visual Basic Editor then select "Run Macro" or you can click the green triangle/pointer on the toolbar.



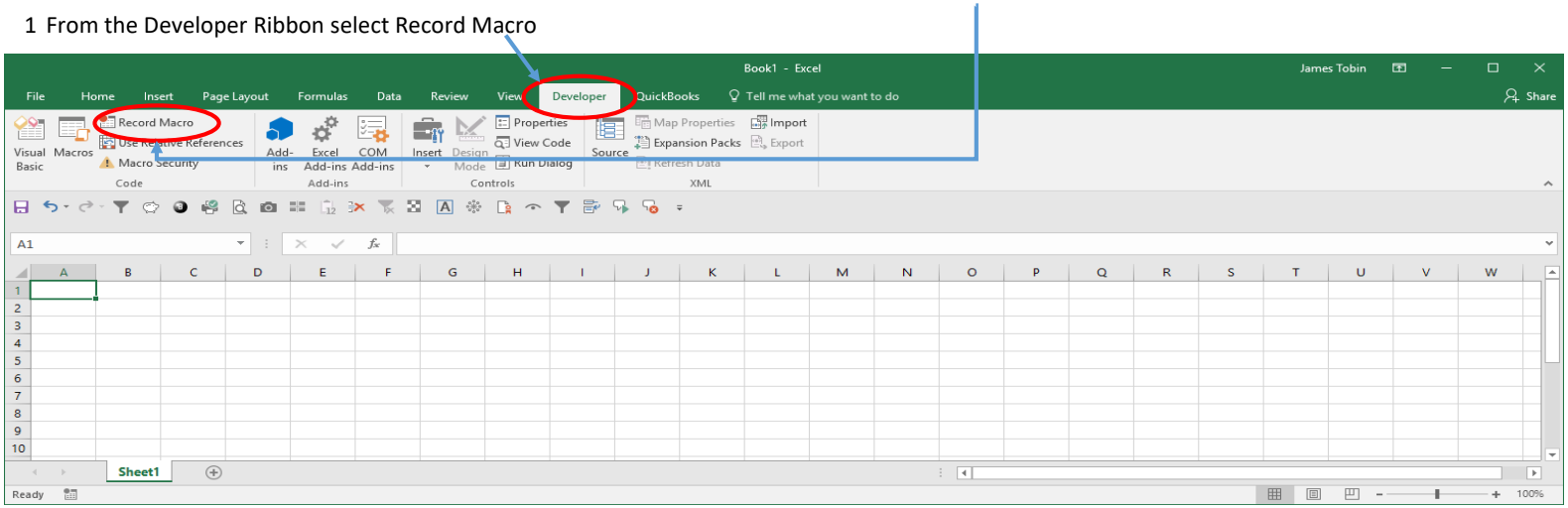

**Save Your Life Tip!!!**

This macro will write what you tell it to write, wherever your cell indicator is on a worksheet, in a workbook. In other words, the Visual Basic Editor assumes the Workbook (Object), the Worksheet (Object), the Cell (Object), unless you specifically tell it otherwise.

## Let's Record a Macro…

IMNTBHO, the fastest and simplest way to begin learning to write macros, is to record one.

1 From the Developer Ribbon select Record Macro




2 A dialogue box will appear asking for a name and description.

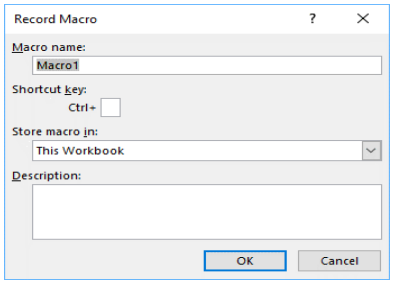
i. In the "Macro name:" field, you can leave the macro name as Macro# or you can assign a new name to the macro you are about to record.
The name cannot have spaces or special characters (e.g. !@#$%^&*,.)

ii. In the "Shortcut key:" field, you can assign a shortcut key, but I'd recommend holding off until you're more familiar with the macro writing process.

iii. In the "Store macro in:" field, you can choose where you want the macro stored. The choices are: "This Workbook", "New Workbook" or "Your Personal Macro Workbook". Unless your sure about where you want to store the macro, store the macro in "This Workbook".

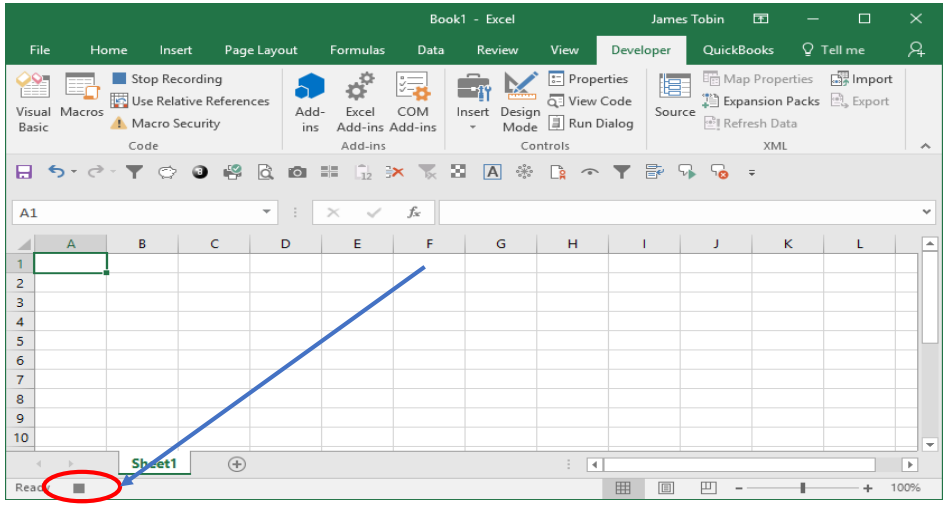
iv. In the "Description:" field, insert a brief description of the macro's purpose and intent. This is not a required field, but can be helpful.

v. Hit OK.

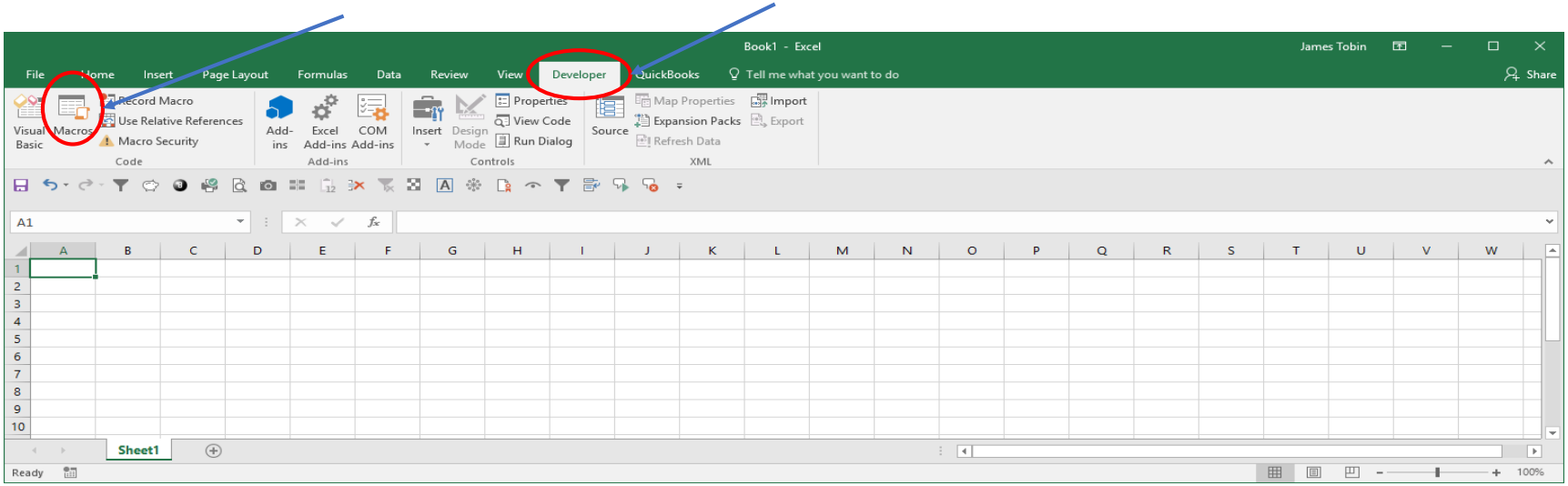



3 Perform the task you're sick of doing. Load it all in there (e.g. cursor movements, formula writing, formatting, copying, pasting, inserting, deleting, whatever…).

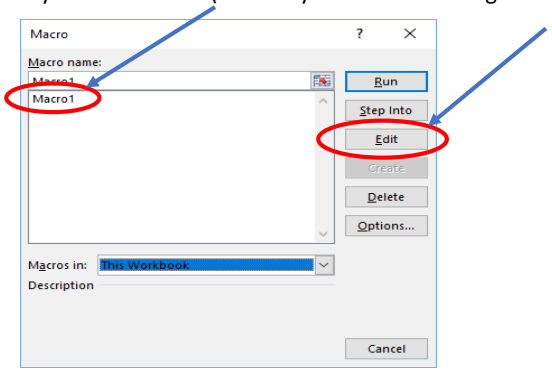
4 When you're finished recording the task you're sick of doing, click on the Stop Recording Button.



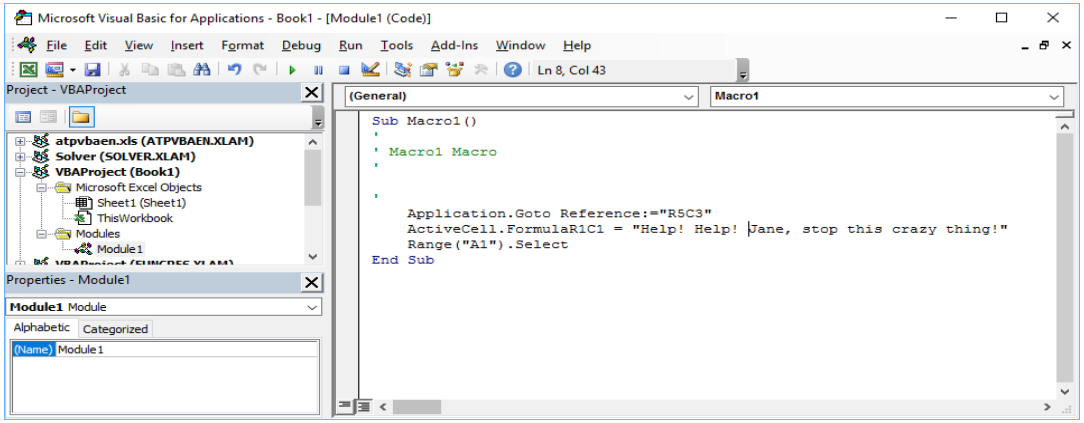

5 Go back to the Developer Ribbon and select Macros (or select Alt +F8 )



Highlight the newly created macro (click only once or it will begin to run/execute).
Select Edit.




7 Your newly created macro will appear.
The language may or may not mean something to you depending upon your experience using Excel and VBA.




If you're satisfied, close the Visual Basic Editor.

If you wish to make a change, you can edit the code (be sure to save via the Save icon on the toolbar) or you can re-record the macro to include the changes you wish to include.

## "Moving" Macros

One of the most common elements of a macro is moving the cell indicator from one place to another to perform some task (e.g. Test a property, or effect a change on an Object etc.).

Below, you'll find a few (NOT ALL…) examples of code to move the cell indicator.

1 Using "Range" to select a single cell
```
Range("b5").Select
```

2 Using "Range" to select a range of cells
```
Range("c6:c30").Select
```

3 Using "Range" to select a named range
```
Range("LiveNamedRange").Select
```

4 Selecting whole columns
```
Columns(2).Select          <==== Selecting a single column, just use the column number
Columns("c:g").Select      <==== Selecting a range of contiguous columns, treat as range w/letters btwn ""s
```

5 Selecting whole rows
```
Rows(5).Select             <==== Selecting a single row, just use the row number
Rows("4:8").Select         <==== Selecting a range of contiguous rows, treat as range w/""s
```

6 Using "Range" to select non-contiguous columns
```
Range("C:C,E:E,I:I,L:N,P:P").Select
```

7 Using "Range" to select non-contiguous rows
```
Range("5:5,9:9,14:17,22:22").Select
```

8 Using "Range" to select non-contiguous cells
```
Range("B4,D13,H5,I13,K6,J22,A24").Select
```

9 Using "Application.Goto Reference:="; note the R1C1 usage
```
Application.Goto reference:="r12c1"
```

10 Selecting all the cells in a worksheet
```
Cells.Select
```

11 How to find out what your active cell address is
```
MsgBox ActiveCell.Address  <==== Introducing "The Message Box" !!! Very Useful!!!
```

## "Moving" Macros cont.

12 How to find out what your active range is

```
MsgBox Selection.Address
```

13 Moving to a cell in another worksheet

Note that this is a 2-step process. If you try to keep it in one line it will cause an error

```
Sheets("Sheet2").Select
Range("A1").Select
```

14 Moving based on current location

```
ActiveCell.Offset(2, 5).Select
```

15 Moving to the bottom of a column of contiguous data

i. Select the top (uppermost) data element

ii. Run the following code

```
Selection.End(xlDown).Select
```

iii. Note that this is the same as End then the Down Arrow as a Keyboard Combo

16 Highlighting/Selecting a column of contiguous data

i. Select the top (uppermost) data element

ii. Run the following code

```
Range(Selection, Selection.End(xlDown)).Select
```

iii. Note that this is the same as holding down the Shift + End keys then the Down Arrow as a Keyboard Combo

My experience is that the last two are accompanied by the Activecell.Offset line

## Copy/Paste Macros

A good number of elementary macros are copy/paste macros; copying from one cell then pasting to another cell or range of cells. This section provides a few (NOT ALL...) ways to copy/paste using macros.

```
Sub SelectRangeCopyRangePasteRange()

   Range("c6").Select                    <==== Selects the source range to copy from.  In this case, cell C6.
   Selection.Copy                        <==== Copies C6
   Range("c7:c30").Select                <==== Selects the targeted paste range.  In this case, the range of C7 thru C30.
   ActiveSheet.Paste                     <==== Pastes to the range C7 thru C30.
   Application.CutCopyMode = False       <==== Turns off Copy/Paste mode in Excel

End Sub
```

```
Sub CopyRangePasteRange()

                                         Selects the source range to copy from.  In this case, cell C6.
   Range("c6").Copy Range("c7:c30")
                                         Selects the targeted paste range.  In this case, the range of C7 thru C30.
   Application.CutCopyMode = False       <==== Turns off Copy/Paste mode in Excel

End Sub
```

```
Sub NoCopyNoPaste()

   Range("C6:C30").Select                <==== Selects the targeted range.  In this case, the range of C6 thru C30.
   Selection.FormulaR1C1 = "=RC[-2]+RC[-1]"   <==== Places the formula in cells C6 thru C30, similar to CTRL+Enter.

End Sub
```

## Three Free Macros…

Here are 3 macros that you can copy and paste into your Personal Workbook and perhaps place a corresponding icon in your Quick Access Toolbar.

1 My pet peeve… I hate getting Excel files that do not print the name of the workbook and where it is stored (saved) on the reports. I was sick and tired of manually installing these in almost every report I received. So I recorded a macro to install the workbook path, workbook name, worksheet, the date, the time, the page number and the total number of pages in the report.

```
Sub FileNameAndPath()
'Macro to insert filename with path, date and time and the page number of the total pages in print-out.
'
   Application.PrintCommunication = False
   With ActiveSheet.PageSetup
      .RightFooter = "&""Arial,Italic""&6&Z&F  &A" & Chr(10) & "&D  &T" & Chr(10) & "Page  &P  of  &N"
      .ScaleWithDocHeaderFooter = True
      .AlignMarginsHeaderFooter = True
   End With
   Application.PrintCommunication = True

End Sub
```

2 Another pet peeve... I frequently would receive Excel files that had a mixture of numeric fonts, in some cases giving the report an almost Picaso-esque appearance. It drove me nuts and I was not going to forward such slop to my customers. I was sick and tired of manually fixing the numeric formats, so I recorded a macro...

```
Sub SetNumbersFormatCommaTwo()
'Macro to set the number format to "Numbers" with comma and two decimal places
'
   Selection.NumberFormat = "#,##0.00_);[Red](#,##0.00)"

End Sub
```

3 This macro has the potential to speed up your spreadsheets. It reduces the number of calculations Excel performs by ensuring that only the range actually used is targeted for calculation.

```
Sub ResetRange()
'Macro to reset/re-establish the usable/calculating range
'
   ActiveSheet.UsedRange

End Sub
```