1. Start Excel. A new, empty spreadsheet with the title Book1 will appear in the Excel window. It will have 3 “sheets” with the default names Sheet1, Sheet2 and Sheet3. Everything we do in this exercise will be done on Sheet1.

2. Click on the **Developer** tab, then click on **Visual Basic** on the developer toolbar.

   A new window should appear with “Microsoft Visual Basic” on its title bar. If the window is completely empty, pull down the **View** menu and select **Project Explorer**, then pull the **View** menu down again and select **Properties Window**.

   This should add two windows inside of the Microsoft Visual Basic window. The top window, the project explorer window, should be displaying a VBA project (representing your Excel spreadsheet file) and within this project you should see components named Sheet1, Sheet2, Sheet3 and ThisWorkbook.

3. Go back to your Excel spreadsheet window. Click on the **Developer** tab, then click on the **Insert** icon. From the subsequent popup window, select (by clicking) a **control button**. Then move your mouse onto the spreadsheet and, while holding down the left mouse button, create a button of the desired size by dragging the mouse.

4. As soon as you are done placing the button on your spreadsheet, an **Assign Macro** window will appear. This offers you the chance to associate a certain macro (which you have not yet created) with that button. The default offering for a button control is a button_click macro. You can choose either to write a **new** macro all by yourself or you can choose to **record** a macro.

   Click on **record**, then go to the Excel spreadsheet window and enter
   
   3 in cell A1
   4 in cell A2
   5 in cell A3
   the formula “=Sum(A1:A3)” in cell B1 (without the quotation marks)

   Click the **Developer** tab and choose **Stop Recording**
5. Return to your Microsoft Visual Basic editor (which we will henceforth call VBE). In the project explorer window, double-click on module1 to make its code window appear. Look for a subroutine named Button1_Click(). This is VBE’s translation of your spreadsheet actions into Visual Basic code. Some macros are easily written by using this record feature. Other, macros can be written from scratch by the spreadsheet programmer. Recording a macro gives you a quick and easy way to create the code that corresponds to certain actions. You will need to get used to the names and properties of the spreadsheet cells, rows, columns, ranges of cells, formulas, etc. before you can become a proficient Excel macro programmer.

6. Return to your Excel spreadsheet. Delete the contents of cells A1, A2, A3 and B1. Click your control button and note what happens. The contents of those cells are restored because your macro stores the instructions necessary to create those contents.

7. Delete the contents of cells A1, A2, A3 and B1 again. Now, while you are still looking at your Excel spreadsheet, press Alt-F8 (or click on the Developer tab, then click Macro on the toolbar). A macro window will pop up showing the names of your macros and offer you the chance to run, step into, edit or delete your macro.

   Click on step into. This will start the execution of your macro, but will pause right after the macro begins execution. A yellow arrow shows the statement that will be executed next if you resume execution. You can now single-step through your macro by pressing F8 repeatedly. Each press of the F8 key will execute one more instruction in your macro. Peek back at Sheet1 in your Excel spreadsheet occasionally, then return to the macro window to press F8. You will see the contents of cells A1, A2, A3 and B1 restored, one by one, as your macro executes.

   Watch the yellow arrow in the macro window carefully. If you keep pressing F8 after your macro completes all its instructions, it will restart at the beginning and go through the same instructions another time.

   This single stepping of macro code comes in handy if you want to experiment with some code or if you are trying to track down a bug in some code you have previously written.

8. Go back to the Visual Basic Editor and pull down the Help menu. The first item listed is Microsoft Visual Basic Help, which you can access by pressing F1 or by selecting this item from the pull down help menu. Open the Help window.

   Under the Concepts entry in the Help Table of Contents, work you way through the “Workbooks and Worksheets”, “Cells and Ranges” and “Controls, Dialog Boxes, and Forms” sections. Read and try to understand this material. With this as starting knowledge, you will be prepared to start writing your own spreadsheet macros.
VBA information sheet

An Excel spreadsheet is regarded as an array of Worksheets
  worksheets(1), worksheets(2), ...
 worksheets("sheet1")

A worksheets contains cells. You can reference individual cells and ranges of cells. You can reference rows and columns as well.
  Rows is the name of the array of all rows on a worksheet
    Rows(1), Rows(2), ...
  Columns is the name of the array of all columns on a worksheet
    columns(1), Columns(2), ...
    Columns("A"), Columns("B"), ...
  Range is a VBA data type.
    Dim mycells as Range
    Set mycells = Union(Rows(1), Rows(3), Rows(8))
    Set mycells = Worksheets(1).Range("A1:D5")
    Range("1:5") ' rows 1..5
    Range("A:C") ' columns A..C
    Range("C5:D9, G9:H16")
    Range("1:1, 3:3, 8:8") ' rows 1, 3, 8
  Cells is the name for the 2-dimensional array of all Cells on a worksheet
    Cells(1,1), Cells(1,2), ... Cells(2,1), ...Cells(row,column)

Cells have attributes and methods.
  Worksheets(1).Cells(1,2).Formula="=AVERAGE(D1:D10)"
  Worksheets(1).Cells(1,1).Value=10
  mycells.Formula="=RAND" ' enter formula in each cell of the range
  Cells(3,3).ClearContents
  Cells(4,4).Activate ' make this the active cell
  ActiveCell.Value=20 ' assign value to the active cell
  Cells(2,2).Font.Bold=True ' display cell contents in bold
  Cells(1,3).Interior.ColorIndex = 27 ' cell background color is set to yellow
  mycells.Font.Bold=True ' all cells in range display contents in bold face font

You have, of course, all the looping and decision constructs of VB at your disposal.
  Dim counter As Integer
  For counter = 1 To 20
      Worksheets(1).Cells(counter,3).Value = counter
  Next counter

  MsgBox “Acknowledge this message”

You also have the power of your Excel spreadsheet functions at your disposal
  x = Application.WorkSheetFunction.Min(mycells)