VBA Introduction

Created by:	Marek Prazak	Reviewed by:	Marek Prazak	Create Date:	29 March 2019

1. Purpose

- Purpose of this lesson is to explain what is a macro, how to set up Excel to be able to use the macro, how to use basic commands and how to work with Record macro function.
- The lesson should take approximately 1 hour to finish.
- As these lessons are so short, there is not nearly enough time to go over each separate line in the coding. You might see some explanation throughout, as it will be necessary for the lessons, but take the initiative and each time you see some part of the code that you do not understand use the internet to find explanation. As you'll see below by using function Record macro you might be able to understand logic of some parts of the VBA.

2. Definition

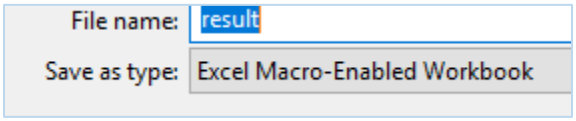
- **Macro** - If you have tasks in **Microsoft Excel** that you do repeatedly, you can record a macro to automate those tasks. A macro is an action or a set of actions that you can run as many times as you want. When you create a macro, you are recording your mouse clicks and keystrokes.
- **VBA** - VBA (Visual Basic for Applications) is the programming language of Excel and other Office programs
- **Form/Active X controls** - Form controls are built in to Excel whereas ActiveX controls are loaded separately.
    - Generally, you'll use Forms controls, they're simpler. ActiveX controls allow for more flexible design and should be used when the job just can't be done with a basic Forms control. In this lesson you'll see basic use of Active X, in later lessons you'll be using Form controls.

3. Content

3.1. Create a Macro

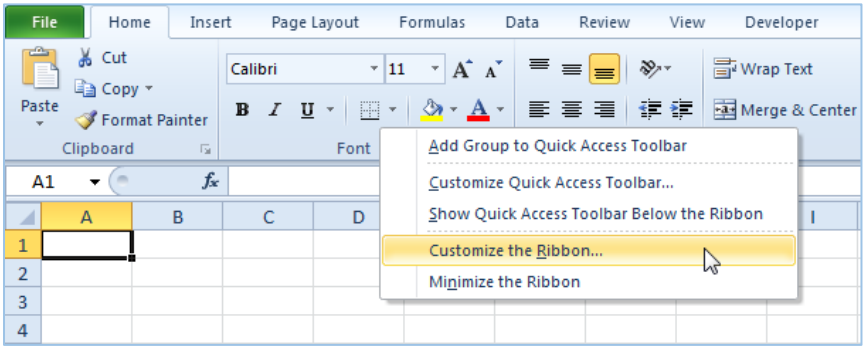
3.1.1. Save Macro-Excel

- Always make sure that you **save your project** as **Macro-Enabled file** otherwise you will lose all your progress.

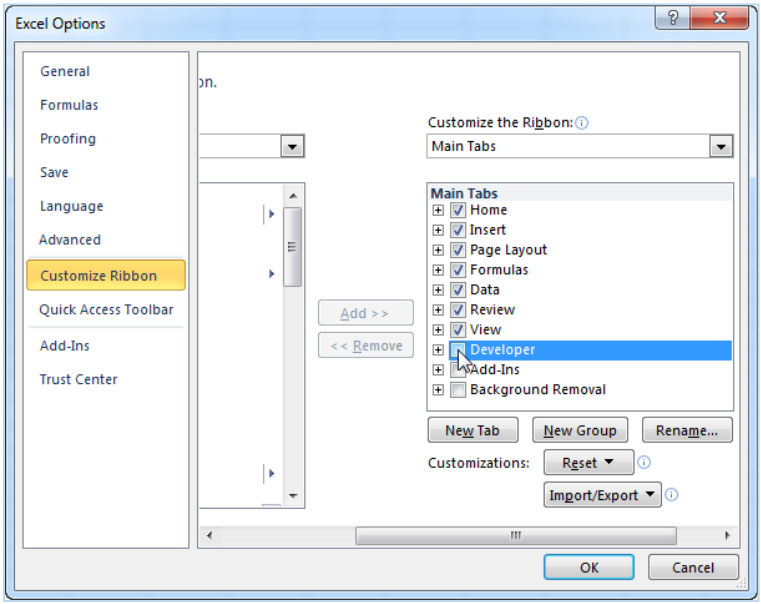


3.1.2. Developer Tab

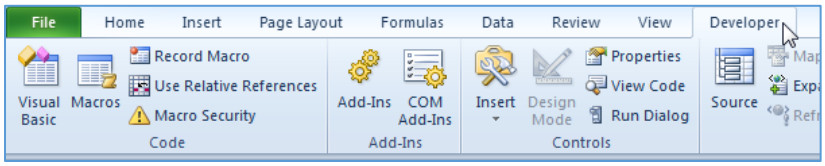
- First check that you have Developer Tab switched on in your Excel. If you don't see the developer tab, proceed with below steps to have it set up.
  - Right click anywhere on the ribbon, and then click Customize the Ribbon.



  - Under Customize the Ribbon, on the right side of the dialog box, select Main tabs (if necessary).
  - Check the Developer check box and click OK.

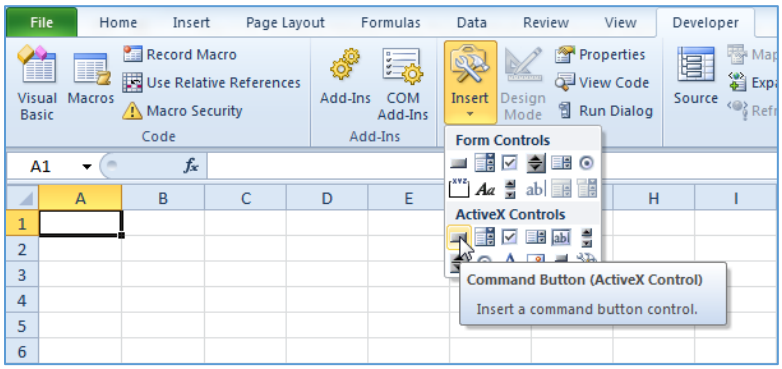


- You can find the Developer tab next to the View tab.



3.1.3. Command Button

- On the **Developer tab**, click **Insert**
- In the ActiveX Controls group, click **Command Button**.

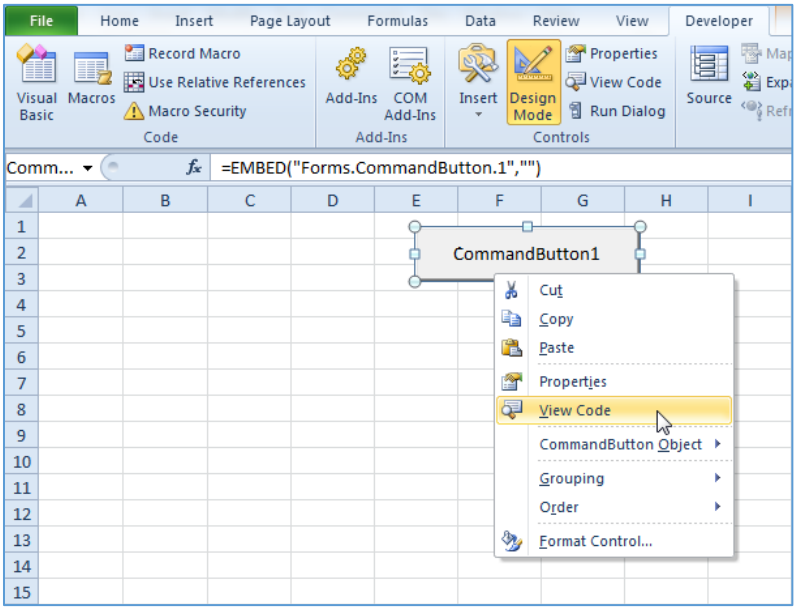


- Drag a command button on your worksheet.

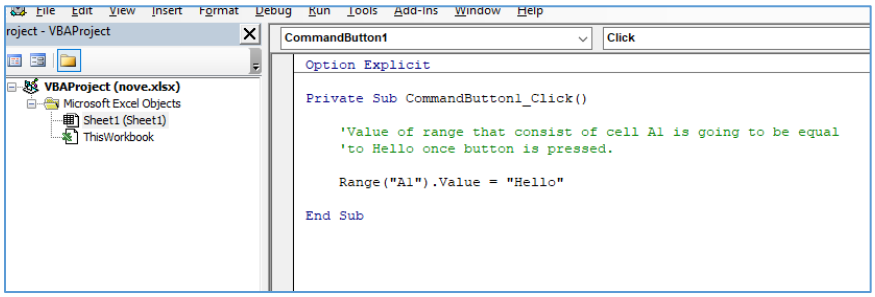
3.1.4. Assign a Macro

- Right click CommandButton1 (make sure **Design Mode is selected**).

- Click on View Code

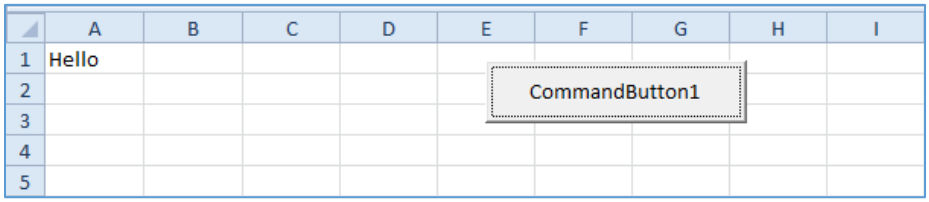


- The Visual Basic Editor appears.
- Place your cursor between **Private Sub CommandButton1_Click()** and **End Sub**
- Add the code line shown below.



Note: *the window on the left with the names Sheet1, Sheet2 and Sheet3 is called the Project Explorer. If the Project Explorer is not visible, click View, Project Explorer. To add the Code window for the first sheet, click Sheet1 (Sheet1).*

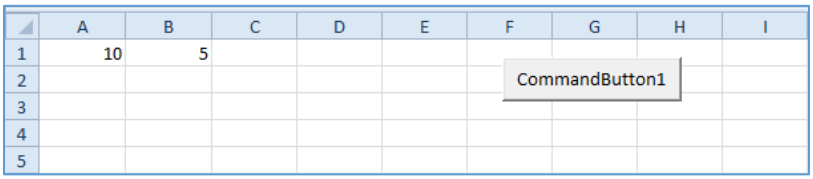
- Close the Visual Basic Editor.
- Click the command button on the sheet (make sure Design Mode is deselected).



- Notice that the Macro code for this button has been placed in sheet1 view in the VBA editor, as such the **macro will work only from this sheet**.

3.1.5. Swap Values

- This example teaches you how to **swap two values** in **Excel VBA**. You will often need this structure in more complicated programs as we are going to see later.
  - Two values on your worksheet.



- Place a command button on your worksheet and add the following code lines.
- First, we **declare a variable** called **temp** of type **Double**. (variable will be explained in later lessons Variables will be explained in later lesson in more detail, but to summarize it in few words, variables are "code" words or names with specific formatting or type. Below variable Temp is double, which means that it can only contain numbers, on the other hand for example type String contains text)

```
Dim temp As Double
```

- We initialize the variable temp with the value of cell A1. (**temp** will be **equal** to the **value** in **cell/range A1**)

```
temp = Range("A1").Value
```

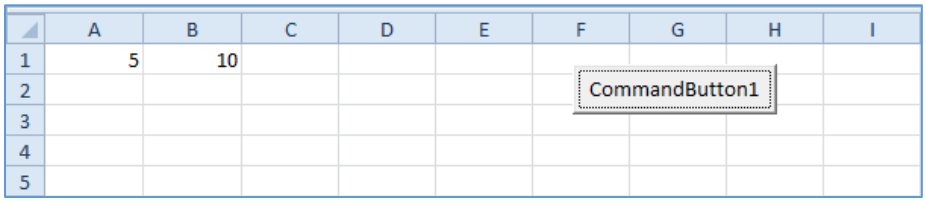
- Now we can safely write the value of cell B1 to cell A1 (**value** of cell/range **A1** is **equal** to **value** of cell/range **B1**

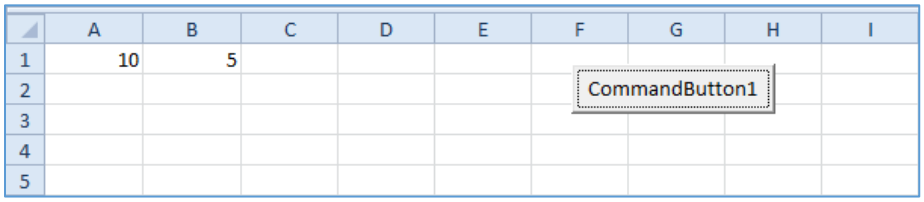
```
Range("A1").Value = Range("B1").Value
```

- Finally, we write the value of cell A1 (written to temp) to cell B1.

```
Range("B1").Value = temp
```

- Click the command button two times.



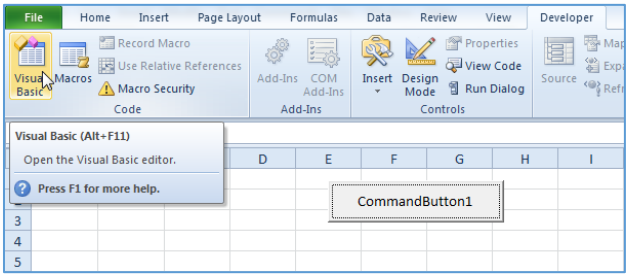



```
Private Sub CommandButton1_Click()

    Dim temp As Double

    temp = Range("A1").Value

    Range("A1").Value = Range("B1").Value

    Range("B1").Value = temp


End Sub
```

3.1.6. Visual Basic Editor

- **The Visual Basic Editor** is not exactly the same as Excel. It is actually a separate application, opened through excel, that allows you to write VBA code on which the macros run. If you'd like to learn more in detail all the useful parts of VBA editor, you can find further source in Additional Knowledge at the end of the lesson.

  - To open the Visual Basic Editor, on the Developer tab, click Visual Basic or press shortcut ALT + F11



- VBA is always written in sequence, meaning whatever is first will run first. Keep this in mind as we go further.
- To help you understand the coding used in these lessons we are taking advantage of comment out symbol, as on example below.

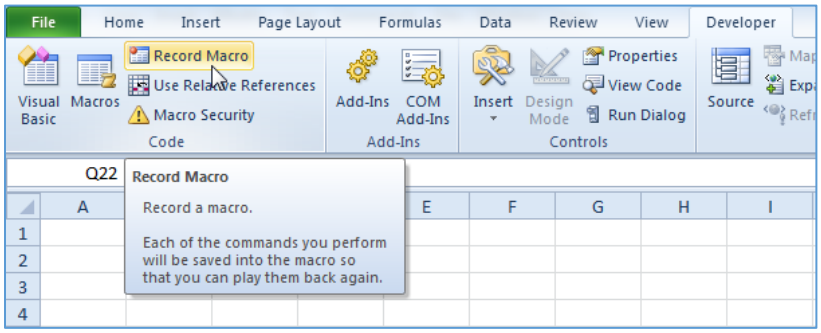
```
'Range("A1").Value = "Hello" This is only comment not actual code, below is your code.

Range("A1").Value = "Hello"
```

3.2. Record macro

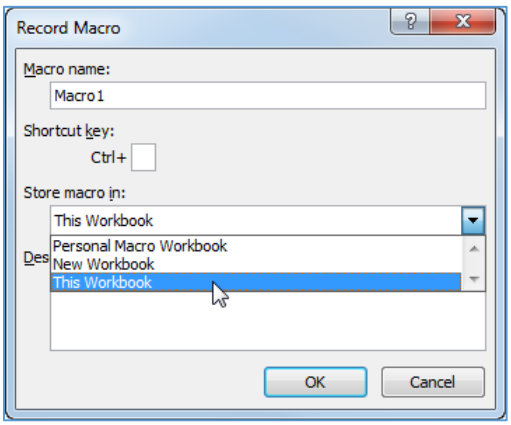
- Record Macro function, records **every** task you perform with Excel. All you have to do is record a specific task once. Next, you can execute the task over and over with the click of a button. The Macro Recorder is also a great help when you don't know how to program a specific task in Excel VBA. Simply open the Visual Basic Editor, after recording the task, to see how it can be programmed.

- Unfortunately, there are a lot of things you cannot do with the Macro Recorder. For example, you cannot loop through a range of data with the Macro Recorder. Moreover, the Macro Recorder uses a lot more code than is necessary, which can slow your program down.

    - On the **Developer tab**, click **Record Macro**.

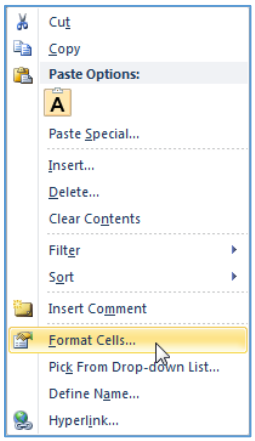


- Enter a name.
- Select This Workbook from the drop-down list. As a result, the macro will only be available in the current workbook. Click OK

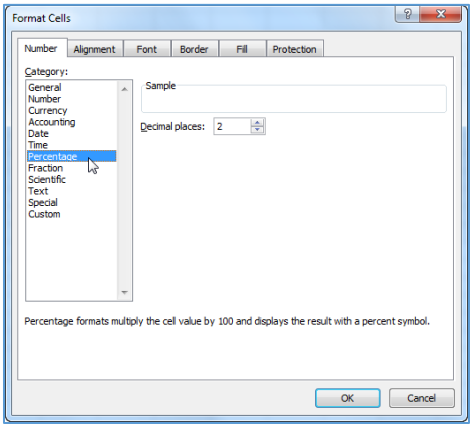


Note: *if you store your macro in Personal Macro Workbook, the macro will be available to all your workbooks (Excel files). This is possible because Excel stores your macro in a hidden workbook that opens automatically when Excel starts. If you store your macro in New Workbook, the macro will be available only in the new workbook that you have created.*

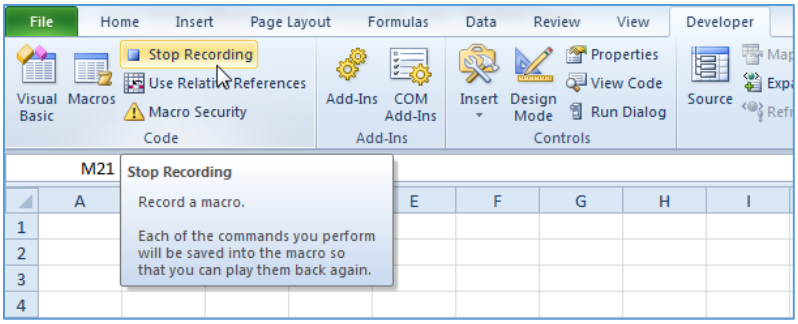
- Right mouse click on the active cell (selected cell). Be sure not to select any other cell! Next, click Format Cells.



- Select Percentage and click OK.

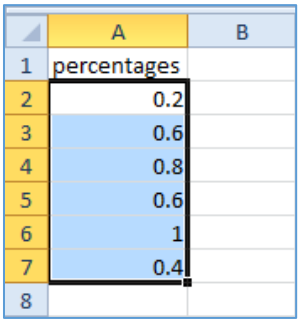


- Finally, click Stop Recording.

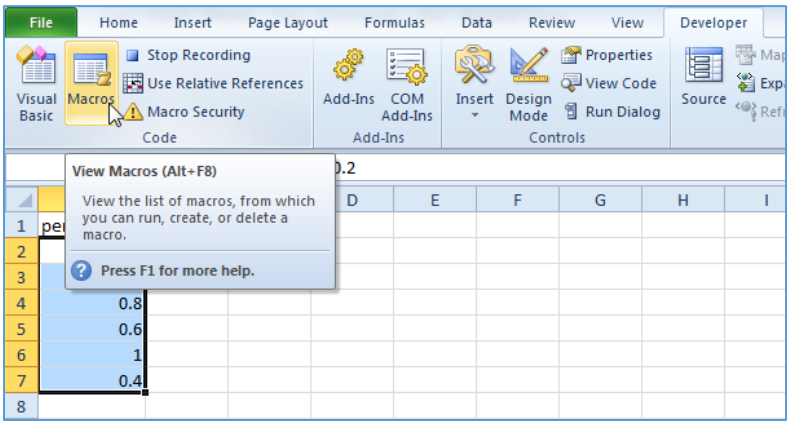


You've just recorded a macro with the Macro Recorder!

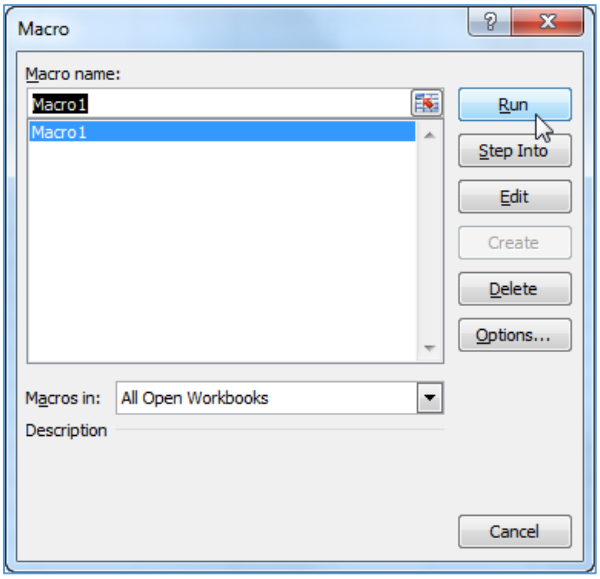
- To Run a Recorded Macro and see that macro converts to percentage
- Enter some numbers between 0 and 1.
- Select the numbers.



- On the Developer tab, click Macros or press ALT + F8.



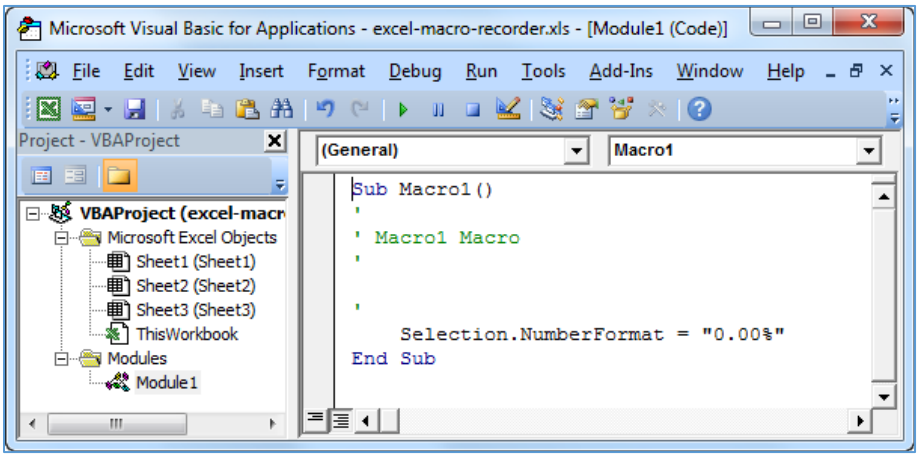
- Click Run.



Result

	A	B
1	percentages	
2	20.00%	
3	60.00%	
4	80.00%	
5	60.00%	
6	100.00%	
7	40.00%	
8		

- Let's look at the code.

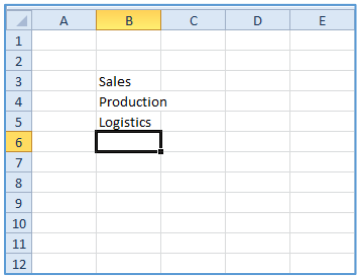


- Explanation: Range/Cell that is selected/highlited, when macro is used, will have its formatting changed

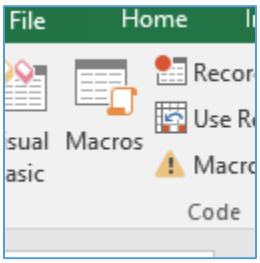
Note: *the macro has been placed into a module called Module1. Code placed into a module is available to the whole workbook. That means, you can select Sheet2 or Sheet3 and change the number format of cells on these sheets as well. Remember, code placed on a sheet (assigned to a command button) is only available for that particular sheet.*

- **Recording in Absolute Mode** - A macro recorded in absolute mode **always** produces **the same result**.

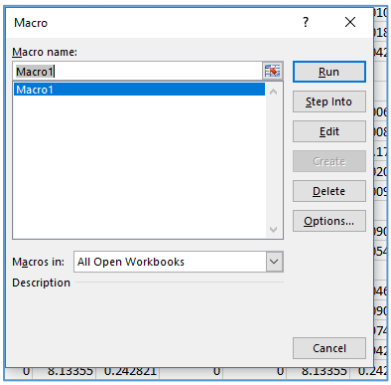
- Click **Record Macro** button.
- Select cell B3. Type Sales and press enter, which selects below cell.
- Type Production and press enter, which selects below cell.
- Type Logistics and press enter, which selects below cell.



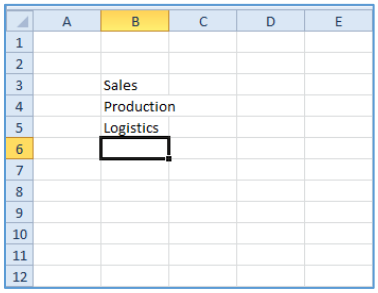
- Click Stop Recording
- Empty Range("B3:B5")
- To run the macro
  - Select any random cell
  - Press Macros button



  - Select Macro1 and press Run



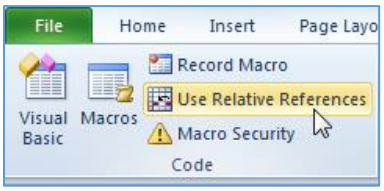
  - Result of your action is that macro will mimic your action and regardless of which cell you've selected it will fill in the recorded cells.



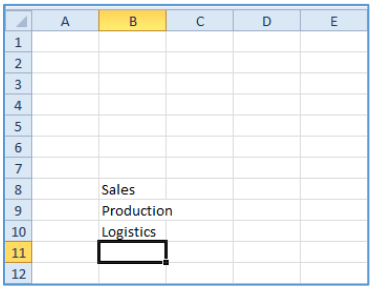
- Let's look at the code now.

```
'E8 is the cell that we've selected
    Range("E8").Select
'active/selected cells value(formula, more later in the lesson) equals to sales
    ActiveCell.FormulaR1C1 = "sales"
'after pressing enter, cell E9 has been selected
    Range("E9").Select
    ActiveCell.FormulaR1C1 = "production"
    Range("E10").Select
    ActiveCell.FormulaR1C1 = "logistic"
    Range("E11").Select
```

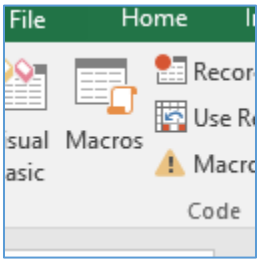
- **Recording in Relative Mode** - A macro recorded in relative mode **can be applied to different cells.**
  - Select "Use Relative References".



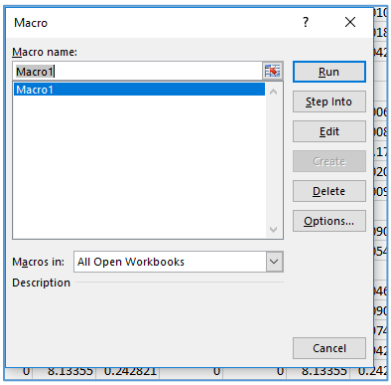
  - Let's start by selecting any single cell (in out example we've clicked on cell B8)
  - Click Record Macro button.
  - Now Type Sales in the cell(B8) and press enter, this will select the cell below.
  - Type Production in the cell(B9) and press enter.
  - Type Logistics in the cell(B10) and press enter.



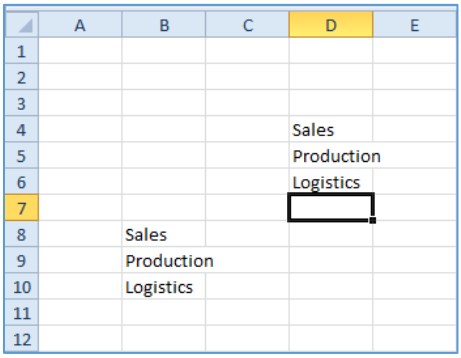
- Click Stop Recording Button.
- To run the macro
  - Select any random cell (in our case, cell D4)
  - Press Macros button



  - Select Macro1 and press Run



  - Result of your action is that macro will mimic the three typed in words starting in the cell that you've selected.



- Let's look at the code now.

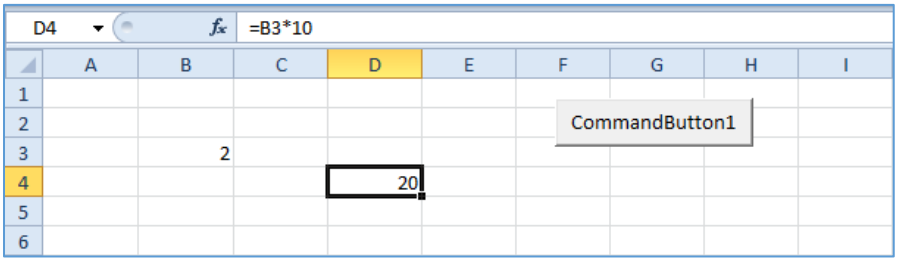
```
'Active cell is considered the cell that was highlited, sometimes also called Selected cell
'Activecell is equal to the word(formula, explained later on) Production
    ActiveCell.FormulaR1C1 = "Production"

'When you've pressed enter, the cell below has been selected.
'In VBA this translates to You have selected a cell that is offset from the originaly active cell by 1 row down and 0 columns to the right.
'This cell is now considered active as it is highleted
    ActiveCell.Offset(1, 0).Select

'New active cell is equal to sales
    ActiveCell.FormulaR1C1 = "Sales"

'Same logic as above applies to other cells
    ActiveCell.Offset(1, 0).Select
    ActiveCell.FormulaR1C1 = "logistic"
    ActiveCell.Offset(1, 0).Select
```

  - As you might have noticed Record macro function will record almost every action that you do in excel (scroll up, down, select, etc.) which is useful for learning and quickly understanding certain commands, but is not very useful for more complex macro, due to the fact that it will record every action performed which slows down the macro and makes the code hard to navigate.

3.3. Formula R1C1

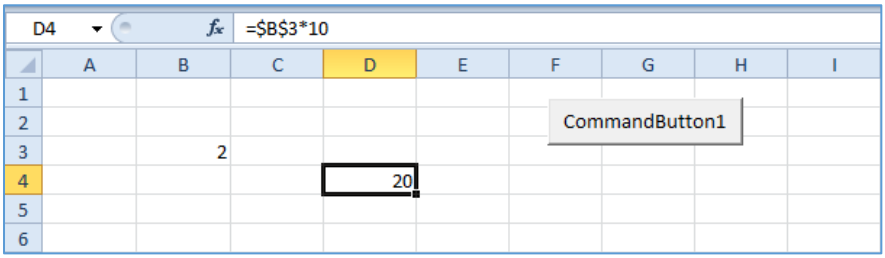
- The example illustrates the difference between **A1**, **R1C1** and **R[1]C[1] style** in **Excel VBA**, which allow you to work with excel formulas.
  - **A1 style**
    - Place a command button on your worksheet and add the following code into Code of the button.

```
Range("D4").Formula = "=B3*10"
```



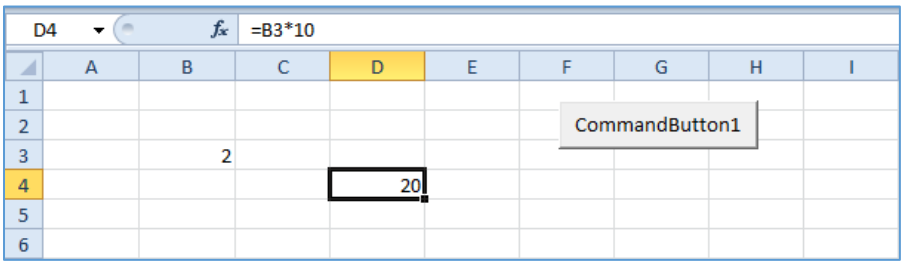
- **R1C1 style**
  - Add the following code line.

```
Range("D4").FormulaR1C1 = "=R3C2*10"
```



  - Cell D4 references cell B3 (R3C2 = row 3, column 2). This is an absolute reference ($ symbol in front of the row number and column letter).
- **R[1]C[1] style**
  - Add the following code line.

```
Range("D4").FormulaR1C1 = "=R[-1]C[-2]*10"
```



- Cell D4 references cell B3 (one row above and 2 columns to the left). This is a relative reference. This code line gives the exact same result as the code line used at step 1.

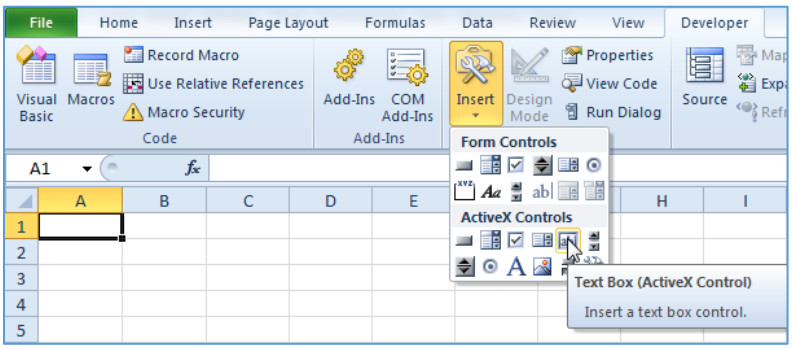
- Why learning about this? Because the Macro Recorder uses the FormulaR1C1 property (R[1]C[1] style). The Macro Recorder creates the following code lines if you enter the formula =B3*10 into cell D4.

```
Sub Macro1()
'
' Macro1 Macro
'

'
    Range("D4").Select
    ActiveCell.FormulaR1C1 = "=R[-1]C[-2]*10"
    Range("D5").Select
End Sub
```

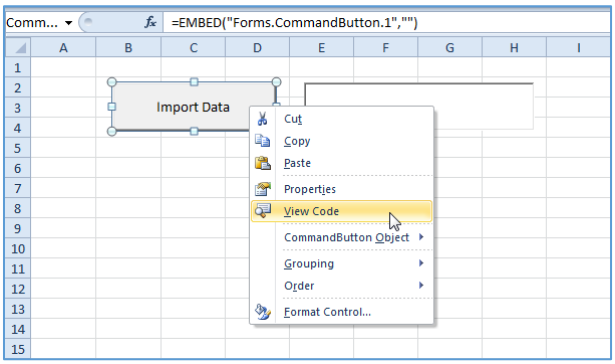
3.4. ActiveX controls

Although in some situations it can be useful to directly place a controls on your worksheet, they are particularly useful when placed on a Userform.

- **Command button** – We've already seen an example of this control above.
- **ActiveX controls** are considered objects in VBA(part of object are considered also outlook functions, form controls, etc.) More on objects see Additional Knowledge. For now think of objects as objects which have many properties that can be called out.
- **Text Box** - A text box is an empty field where a user can fill in a piece of text.
  - On the Developer tab, click Insert.
  - In the ActiveX Controls group, click Text Box.



- Drag a command button and a text box on your worksheet.
- Right click the command button (make sure Design Mode is selected).
- Click View Code.



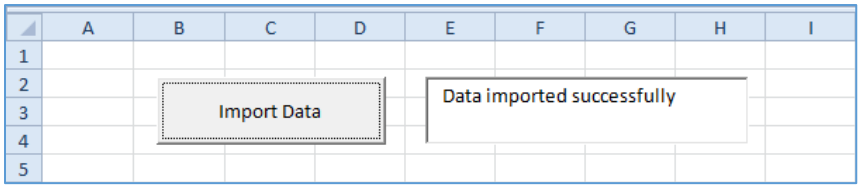
Note: *you can change the caption and name of a control by right clicking on the control (make sure Design Mode is selected) and then clicking on Properties. Change the caption of the command button to Import Data. For now, we will leave TextBox1 as the name of the text box.*

- Add the following code line:

```
TextBox1.Text = "Data imported successfully"
```

Note: *For now don't worry too much about the structure of the syntax as you'll get hang of that as you progress. For now think of* ***objectName.property's*** *object as "object that has something being done to it", property is that something.*

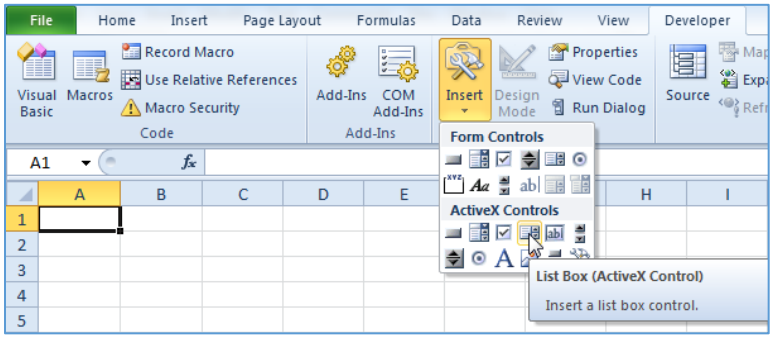
- Click the command button on the sheet (make sure Design Mode is deselected).



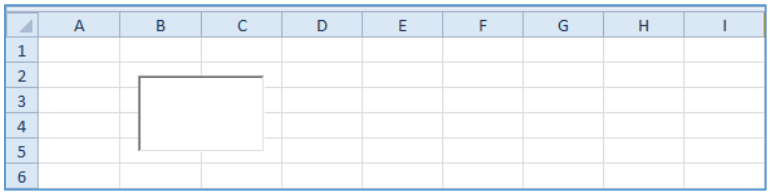
- To clear a text box, use the following code line:

```
TextBox1.Value = ""
```

- **List Box -** A list box is a list from where a user can select an item.
  - In the ActiveX Controls group, click List Box.

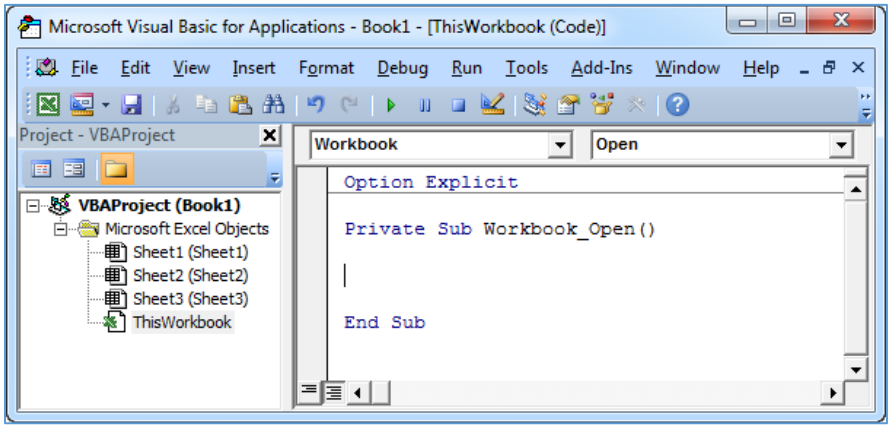


  - Drag a list box on your worksheet.



Create a **Workbook Open Event**(more on how this works will be explained in later lessons). Code added to the Workbook Open Event will be executed by Excel VBA when you open the workbook.

- Open the Visual Basic Editor. ALT+F11
- Double click on ThisWorkbook in the Project Explorer.
- Choose Workbook from the left drop-down list and choose Open from the right drop-down list.



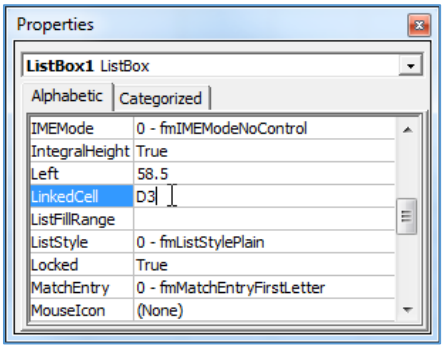
- To add items to the list box, add the following code lines to the Workbook Open Event:

```
With Sheet1.ListBox1
    .AddItem "Paris"
    .AddItem "New York"
    .AddItem "London"
End With
```

Note: *use Sheet2 if your list box is located on the second worksheet, Sheet3 if your list box is located on the third worksheet, etc. If you use these code lines outside the Workbook Open event, you might want to add the following code line before these code lines. This code line clears the list box. This way your items won't be added multiple times if you execute your code more than once.*

```
ListBox1.Clear
```

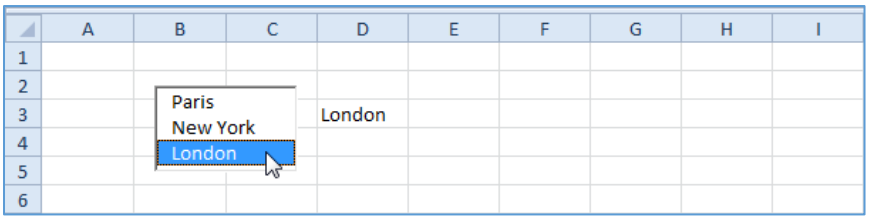
- To link this list box to a cell, right click on the list box (make sure design mode is selected) and click on Properties. Fill in D3 for LinkedCell.



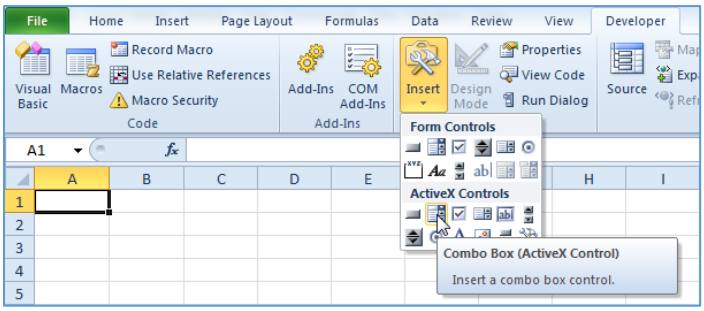
Note: *also see the ListFillRange property to fill a list box with a range of cells*

- Save, close and reopen the Excel file.

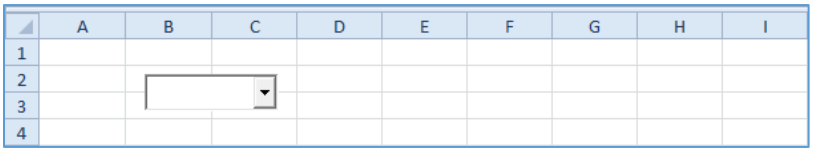
Result



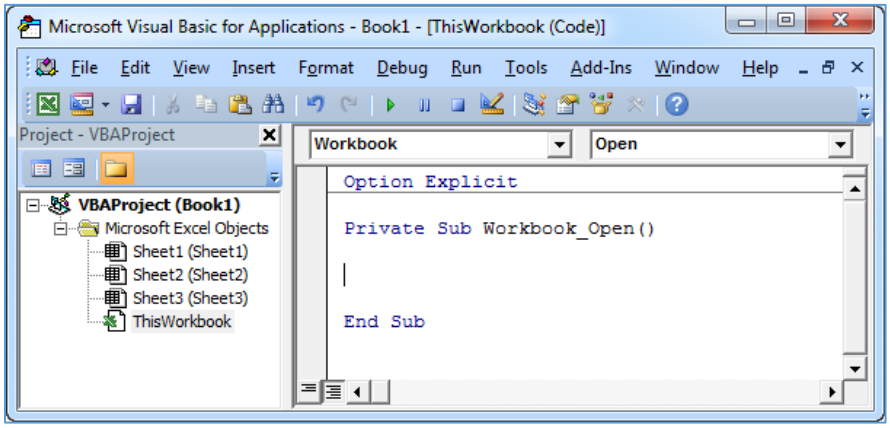
- **Combo box -** A combo box is a drop-down list from where a user can select an item or fill in his/her own choice.
  - In the ActiveX Controls group, click Combo Box.



  - Drag a combo box on your worksheet.



  - Open the Visual Basic Editor.
  - Double click on This Workbook in the Project Explorer.
  - Choose Workbook from the left drop-down list and choose Open from the right drop-down list.



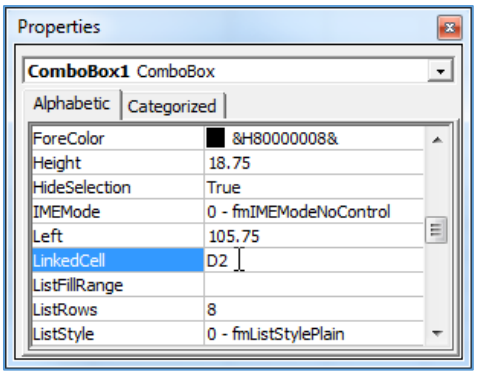
- To add items to the combo box, add the following code lines to the Workbook Open Event:

```
With Sheet1.ComboBox1
    .AddItem "Paris"
    .AddItem "New York"
    .AddItem "London"
End With
```

Note: *use Sheet2 if your combo box is located on the second worksheet, Sheet3 if your combo box is located on the third worksheet, etc. If you use these code lines outside the Workbook Open event, you might want to add the code lines below before these code lines. The first code line clears the combo box. This way your items won't be added multiple times if you execute your code more than once. The second code line clears your own choice.*

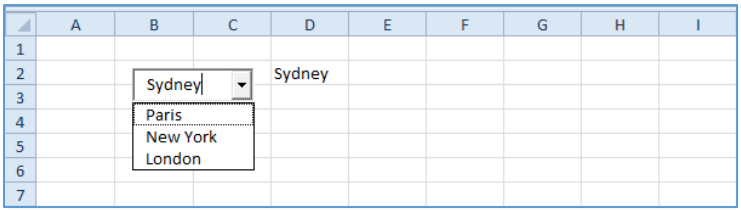
```
ComboBox1.Clear
ComboBox1.Value = ""
```

- To link this combo box to a cell, right click on the combo box (make sure design mode is selected) and click on Properties. Fill in D2 for **LinkedCell**.

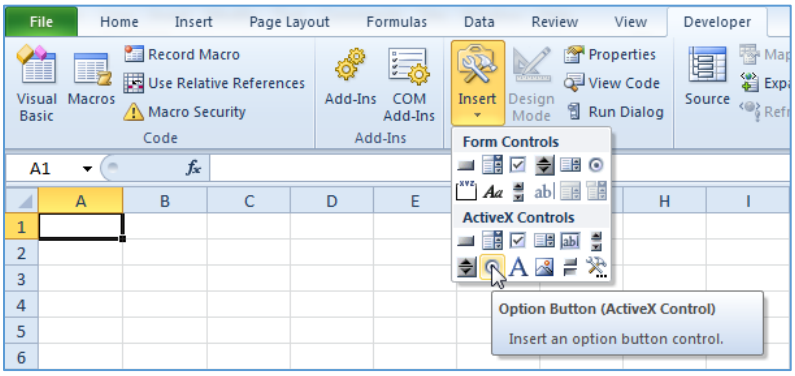


Note: *also see the **ListFillRange** property to fill a combo box with a range of cells.*

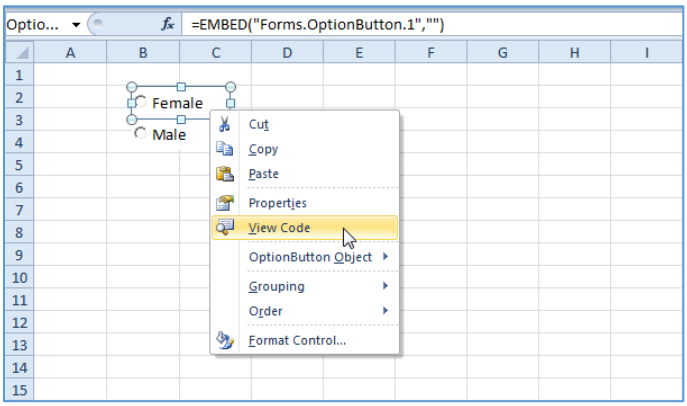
- Save, close and reopen the Excel file.



- **Option Buttons**
  - In the ActiveX Controls group, click Option Button.



- Drag two Option buttons on your worksheet.
- Right click the first Option button.



Note: *you can change the caption and name of a control by right clicking on the control (make sure Design Mode is selected) and then clicking on Properties. Change the captions of the option buttons to Female and Male. For now, we will leave OptionButton1 and OptionButton2 as the names of the option buttons.*

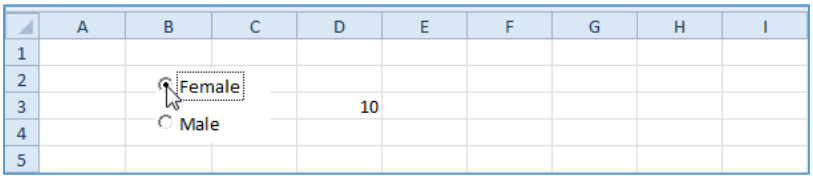
- Add the following code line:

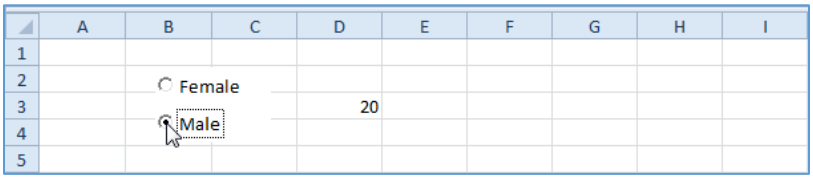
```
If OptionButton1.Value = True Then Range("D3").Value = 10
```

- Right click the second Option button (make sure Design Mode is selected).
- Click View Code.
- Add the following code line:

```
If OptionButton2.Value = True Then Range("D3").Value = 20
```

- lick the option buttons on the sheet (make sure Design Mode is deselected).





4. Closing Exercise

1) Use excel named source to get you started with the lay out
2) Create list box that will contain values GBR, POR, CZE that will be written out into cell F5
3) Create textbox (later used for confirmation)
4) Create formula If, that will decide value based on chosen country into cell G5, conditions as per below
- GBR = 2
- POR = 3
- CZE = 4
5) Create command button, name it Start.
6) Button will start code that does following
- Creates formula in cell J4 that multiplies Result of formula by 10
- Creates formula in cell J5 that multiplies Result of formula by 20
- Creates formula in cell J6 that multiplies Result of formula by 30
- Copies cells J4, J5, J6 and pastes them as values (to remove formulas) into sheet Result and into the range/table named Result
- Switches back into Sheet1
- Writes “Done” into textbox
7) Adjust the Sheet1 to make Grindlines and Headings disappear

5. Additional Knoledge

VBA editor in detail - https://powerspreadsheets.com/excel-visual-basic-editor/

VBA Objects - https://excelmacromastery.com/vba-objects/