

Pop Up Message Boxes

Created by:	Hari Krish	Reviewed by:	Marek Prazak	Create Date:	29/03/2019
--------------------	------------	---------------------	--------------	---------------------	------------

1. Purpose.....	2
2. Definition	2
3. Content	2
3.1 POP MESSAGE BOX	2
3.1.1 SYNTAX	2
3.1.2 TYPES OF MSG BOXES	2
3.1.3 USE OF MSGBOX	2
3.2 Excel Functions in VBA.....	9
3.2.1 USE OF FORMULAS IN VBA	9
4. CLOSING EXERCISE	11
5. ADDITIONAL KNOWLEDGE	11

1.0 Purpose

- This lesson will show you how to create Pop Messages in VBA which are used to inform the user of steps in the macro or for receiving input. We're as well going to explain how to code the basic formulas in Excel using VBA by which you'll be able to avoid unnecessary use of excel formulas that would otherwise slow down your workbook.
- Don't despair if you won't understand all from just reading. We highly suggest to test out the content by trying to use it as you progress through the lesson.
- The lesson should take approximately 1 hour for completion.

2.0 Definition

- The Message Box is a pop up box in Excel VBA, through which you can inform the users of your program or intake strings into it.

3.0 Content

3.1 POP MESSAGE BOX

3.1.1 SYNTAX

- Below you can see a structure (syntax) of a basic message box. Desired text has to be placed inside the quotation marks.

MsgBox "ENTER YOUR REQUIRED TEXT HERE"

3.1.2 TYPES OF MSG BOXES

- Message box can be divided into two types, **MsgBox** and **InputBox**.

MsgBox - Shows a message with some information to the user.

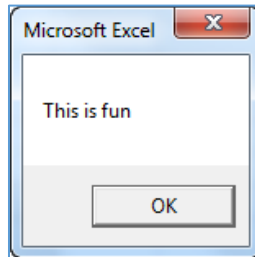
InputBox - This is an input form that allows the user to enter data.

3.1.3 USE OF MSGBOX

- To try the **MsgBox**, create a new **Commandbutton** and write in code as per below examples.

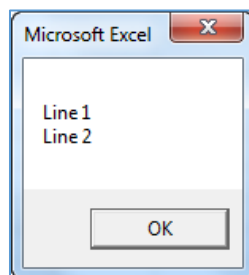
- **Single Line Text** – when you click the command button, a simple pop up box with custom message will appear, to close this, click OK or close the window.

```
Private Sub CommandButton1_Click()  
    MsgBox "This is fun"  
End Sub
```



- **Multi Line Text** – similar to the previous example, a simple message will appear. With use of **vbNewLine** you get a new line in the message. As **vbNewLine** is a code you have to put the text of the message (all lines) between quotation marks as well as place **&** symbols in front and behind the **vbNewLine** code. See the example below

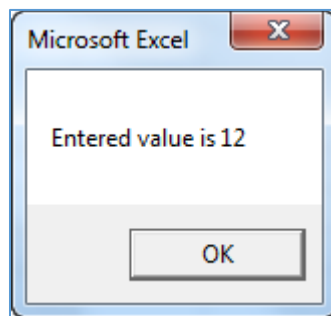
```
Private Sub CommandButton1_Click()  
    MsgBox "Line 1" & vbNewLine & "Line 2"  
End Sub
```



- **Reading a Cell Value and Displaying** – when you place value into the cell A1 it will be read by the macro

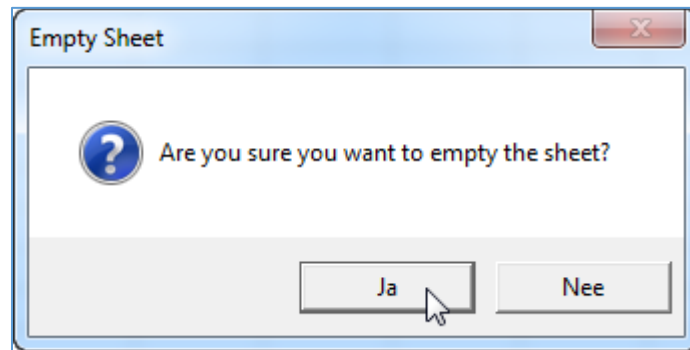
```
Private Sub CommandButton1_Click()  
    MsgBox "Entered value is " & Range("A1").Value  
End Sub
```

	A	B
1	12	
2		
3		



- **User choice**
 - There are many different versions of **User choice** MsgBox, below you can see just a few of them.
 - **Yes / No**

```
Private Sub CommandButton1_Click()  
    MsgBox "Are you sure you want to empty the sheet?", vbYesNo + vbQuestion, "Empty Sheet"  
End Sub
```

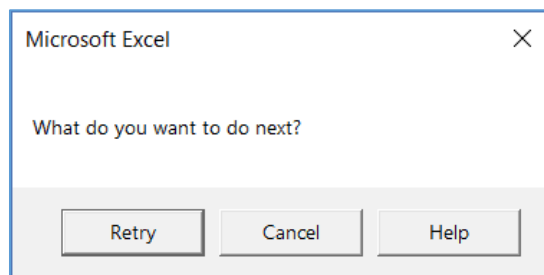


- **Retry/ Cancel/ Help**

```
Private Sub CommandButton1_Click()

    MsgBox "What do you want to do next?", vbRetryCancel + vbMsgBoxHelpButton

End Sub
```



Note: in this code, we have combined two different button constants (**vbRetryCancel** + **vbMsgBoxHelpButton**). The first part shows the **Retry and Cancel** buttons and the second part shows the **Help** button.

Button Constant	Description
vbOKOnly	Shows only the OK button
vbOKCancel	Shows the OK and Cancel buttons
vbAbortRetryIgnore	Shows the Abort, Retry, and Ignore buttons
vbYesNo	Shows the Yes and No buttons
vbYesNoCancel	Shows the Yes, No, and Cancel buttons
vbRetryCancel	Shows the Retry and Cancel buttons
vbMsgBoxHelpButton	Shows the Help button. For this to work, you need to use the help and context arguments in the MsgBox function
vbDefaultButton1	Makes the first button default. You can change the number to change the default button. For example, vbDefaultButton2 makes the second button as the default

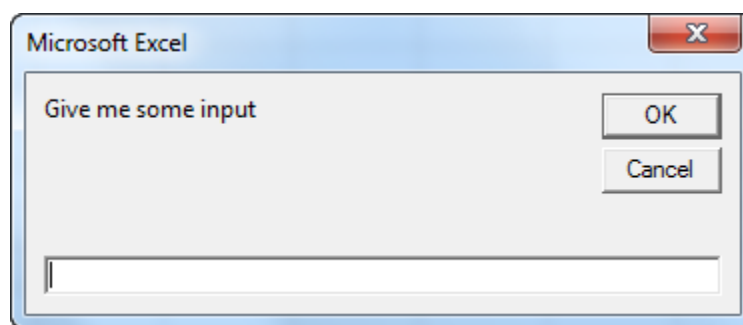
Note: While going through the examples of creating different buttons, you may wonder what the point of having all these buttons is if it doesn't have any impact on the code.”.

It does! Based on the selection, you can code what you want the code to perform as a result. For example, if you select OK, the code should continue, and if you click Cancel, the code should stop. This can be done by using variables and assigning the value of the Message Box to a variable. We will cover this in later sections of this tutorial.

- **InputBox**

- You can use the **InputBox function** in **Excel VBA** to prompt the user to enter a value
- Place a command button on your worksheet and add the below code lines.

```
Private Sub CommandButton1_Click()  
  
    Dim myValue As Variant  
  
    myValue = InputBox("Give me some input")  
  
    Range("A1").Value = myValue  
  
End Sub
```



Result is when the user enters the value 5 and clicks the OK button, value will be filled in cell A1.

	A	B	C	D	E	F	G	H	I
1	5								
2							CommandButton1		
3									
4									
5									

Note: we use a variable of type *Variant* here because a *Variant* variable can hold any type of value. This way the user can enter text, numbers, etc. (more on variables later).

- **MsgBox Icon Constants**

- Apart from the buttons, you can also **customize** the **icons** that are displayed in the MsgBox dialog box. For example, you can have a red critical icon or a blue informational icon.
- Below is a table that lists the codes for their corresponding icons.
- Example of such use you've already seen with MsgBox utilizing user choice.

Icon Constant	Description
vbCritical	Shows the critical message icon
vbQuestion	Shows the question icon
vbExclamation	Shows the warning message icon
vbInformation	Shows the information icon

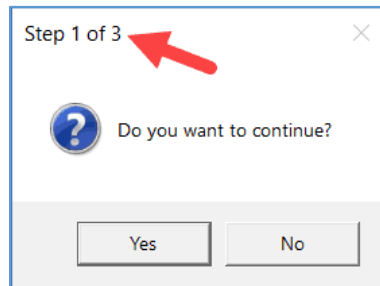
- **Customizing Title in the MsgBox**

- When using MsgBox, you can customize the title and the prompt messages.
- In case you don't specify the title argument, MsgBox automatically uses the title of the application (which has been Microsoft Excel in this case).
- You can customize the title by specifying it in the code.

```
Private Sub CommandButton1_Click()

    MsgBox "Do you want to continue?", vbYesNo + vbQuestion, "Step 1 of 3"

End Sub
```



- **MsgBox Value to a Variable**

- With MsgBox function in Excel, you can decide what you want to do when a user clicks a specific button. This is possible as every button has a value associated to it.
- If you click on the **Yes** button, the MsgBox function returns a value (**6** or the constant **vbYes**) which you can use in the code. Similarly, if the user selects the **No** button, it returns a different value (**7** or the constant **vbNo**) that can be used in the code.

You don't need to memorize these, just be aware of it and that you can use the constants.

Button Clicked	Constant	Value
Ok	vbOk	1
Cancel	vbCancel	2
Abort	vbAbort	3
Retry	vbRetry	4
Ignore	vbIgnore	5
Yes	vbYes	6
No	vbNo	7

- Now let's see how we can control the VBA macro code based on what button a user clicks.
- In the below code, if the user clicks Yes, it displays the message "You Clicked Yes", and if the user clicks No, it displays, "You clicked No".

NOTE: Don't worry about what each part of the code does for now as you'll understand **IF** in next lesson.

```
Private Sub CommandButton1_Click()

    Result = MsgBox("Do you want to continue?", vbYesNo + vbQuestion)

    If Result = vbYes Then
        MsgBox "You clicked Yes"
    Else
        MsgBox "You clicked No"
    End If

End Sub
```

- In the above code, we have assigned the value of the MsgBox function to the Result variable. When you click Yes button, the Result variable gets the vbYes constant (or the number 6) and when you click No, the Result variable gets the vbNo constant (or the number 7).
- Then we used the If_Then_Else construct to check if the Result variable holds the value **vbYes**. If **it does**, it shows the prompt **"You Clicked Yes"**, **else** it shows **"You clicked No"**.
- You can use the same concept to run a code, if the user clicks No exit the sub.

3.2 Excel Functions in VBA

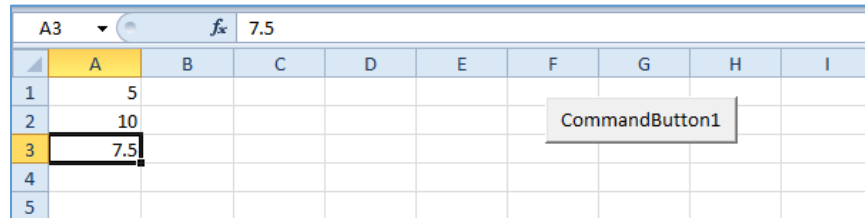
- Through use of VBA you can use Excel formulas. The Formulas used in VBA are acting in same fashion as regular Excel formulas. Most common ones are **SUM, SUMIF, SUMIFS, COUNT, COUNTA, COUNTIF, COUNTIFS, AVERAGE, AVERAGEIF, VLOOKUP**. You might ask, why not just implement regular excel formula. Benefit of calculating and using the formulas in the code is that your macro file won't be so large and excel itself will not be as slow as if you were to use thousands of formulas.

3.2.1 USE OF FORMULAS IN VBA

- Place a command button on your worksheet and add the below code line.

```
Private Sub CommandButton1_Click()  
    Range("A3").Value = Application.WorksheetFunction.Average(Range("A1:A2"))  
End Sub
```

- When you click the command button on the worksheet, Excel VBA calculates the average of the values in cell A1 and cell A2 and places the result into cell A3.



	A	B	C	D	E	F	G	H	I
1	5								
2	10								
3	7.5								
4									
5									

Note: If you look at the formula bar, you can see that the formula itself is not inserted into cell A3. To insert the formula itself into cell A3, use the following code line **Range("A3").Value = "=AVERAGE(A1:A2)"**

- In the future you might see that it is possible to simplify your code by leaving out the word Application, as the Application object is always assumed, but as it is best practice we would recommend to keep using it as it helps with readability of the code.

4.0 Closing Exercise

Complete practice exercise using below steps.

- 1) Create command button
- 2) Write a code to have an Inputbox1 popup that would write a value into cell A1.
- 3) Once Inputbox1 is confirmed have Inputbox2 pop up and have it write value into cell A2.
- 4) Once confirmed display Msg Box "Thank you for input." and on the next line "Do you want to continue?"
Give the user option to choose between Yes/No/Cancel, use question icon.
- 5) Try to use If/Else/EndIf in your code to get result messages.
when YES – use formula SUM to sum of A1 and A2 and write out results into cell A3 and give out pop up message with the result and header of this pop up box is "Result"
when NO – give out pop up box saying "I will not count anything"
when CANCEL – do nothing and close the questions

5.0 Additional knowledge

functions - <https://exceljet.net/excel-functions>

formulas - <https://exceljet.net/formulas>

keyboard shortcuts - <https://exceljet.net/keyboard-shortcuts>