# **** VBA, visual basic

## Indhold

## Debugging

https://www.geeksforgeeks.org/debugging-vba-code-in-excel/

# MsgBox "value in A1 is " & val  kan placeres i koden

Debug.Print vbCrLf  & "Sub MaxNumberInCellA20_down_Click()"  udskrives I imidiate vinduet, vbCrLf  giver linjeskift ifht forrige udskrift
Kan ikke få step into I debugging til at virke

# Finding All Pivot Tables In A Workbook

To easily find all the **pivot tables in your Excel workbook**, you can use two straightforward methods: creating a VBA Macro or using Excel's Find feature. The VBA Macro is the best solution as it automatically extracts the name and location for each pivot table.

Creating A VBA Macro

A VBA Macro is the best solution for listing all pivot tables in your workbook. You can create a custom script that extracts the name and location for each pivot table.

Press Alt+F11 to open the Microsoft Visual Basic for Applications window.

Go to "Insert," then click "Module."

Copy and paste the following code in the module window:

```
Sub ListAllPivotTables()

   Dim ws As Worksheet
   Dim pt As PivotTable
   Dim i As Integer

   i = 1

   For Each ws In ActiveWorkbook.Worksheets

      For Each pt In ws.PivotTables

         Debug.Print i & ". " & pt.Name & " on " & ws.Name
```

```
        i = i + 1

    Next pt

  Next ws

End Sub
```
Press F5 to execute the macro.

## Create a Macro med VBA

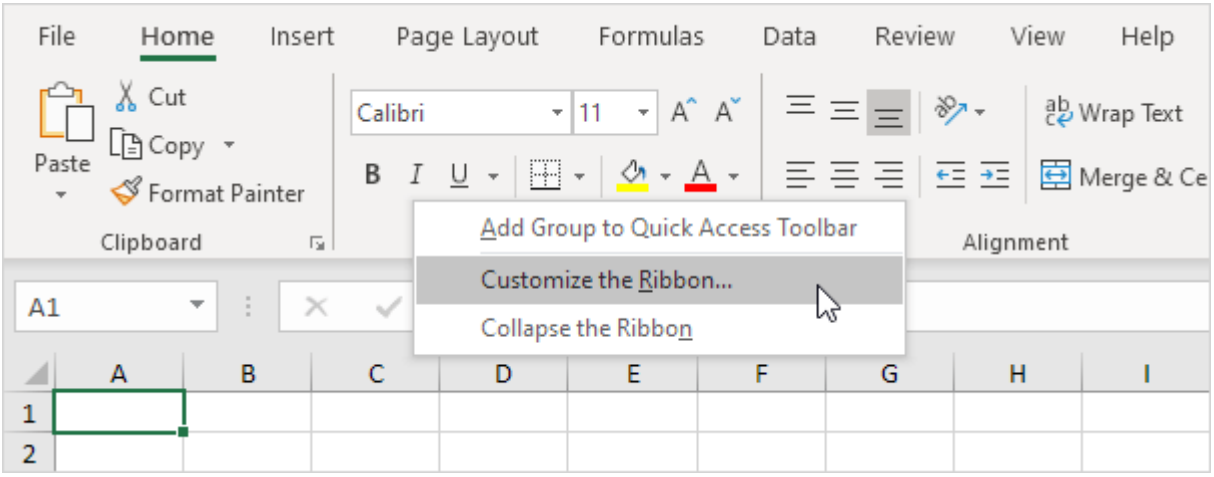
https://www.excel-easy.com/vba/create-a-macro.html

With Excel VBA you can automate tasks in Excel by writing so-called macros. In this chapter, learn how to create a simple macro which will be executed after clicking on a command button. First, turn on the Developer tab.

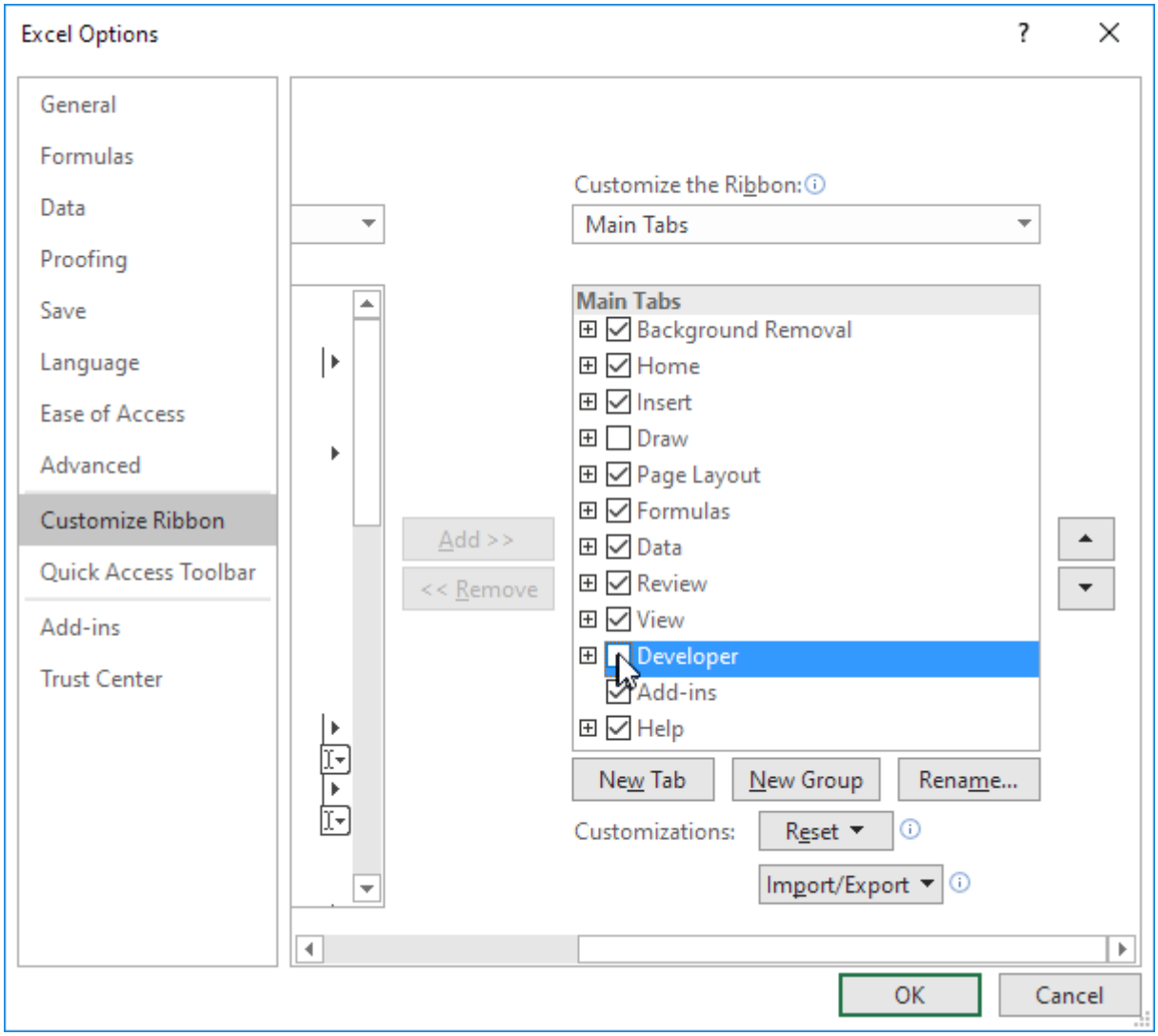
Developer Tab

**To turn on the Developer tab, execute the following steps.**

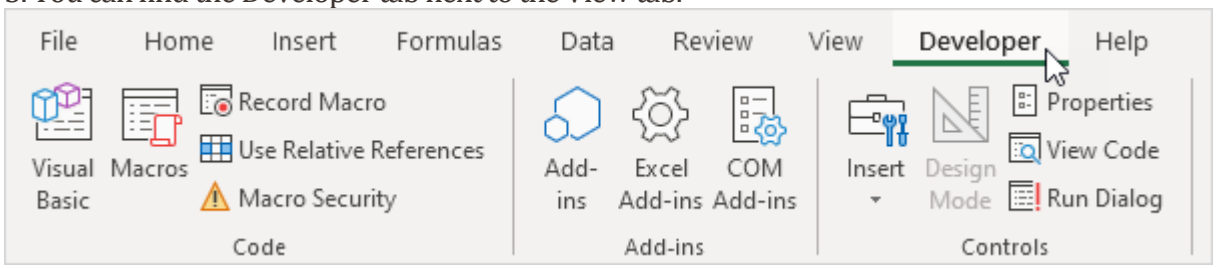
1. Right click anywhere on the ribbon, and then click Customize the Ribbon.




2. Under Customize the Ribbon, on the right side of the dialog box, select Main tabs (if necessary).
3. Check the Developer check box.




4. Click OK.
5. You can find the Developer tab next to the View tab.




# Command Button laves

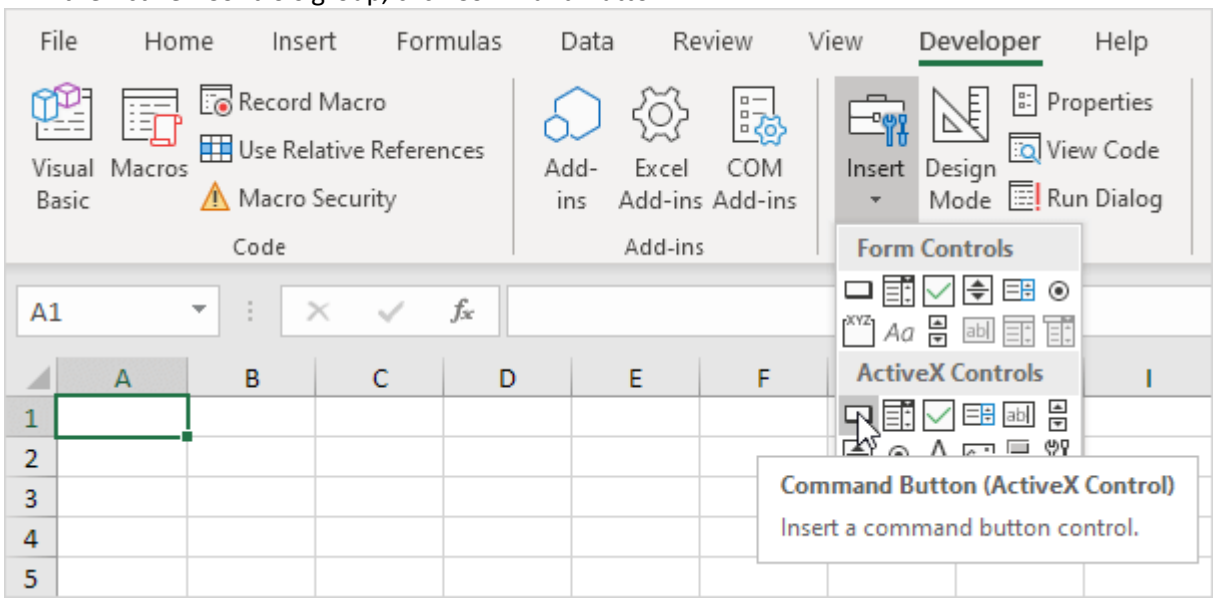
aktiver designeren, indsæt button/objekt, klik objektet, så åbnes udkast til eventhandler som sub

Save filen, luk design-vinduet

To place a command button on your worksheet, execute the following steps.

1. On the Developer tab, click Insert.

2. In the ActiveX Controls group, click Command Button.



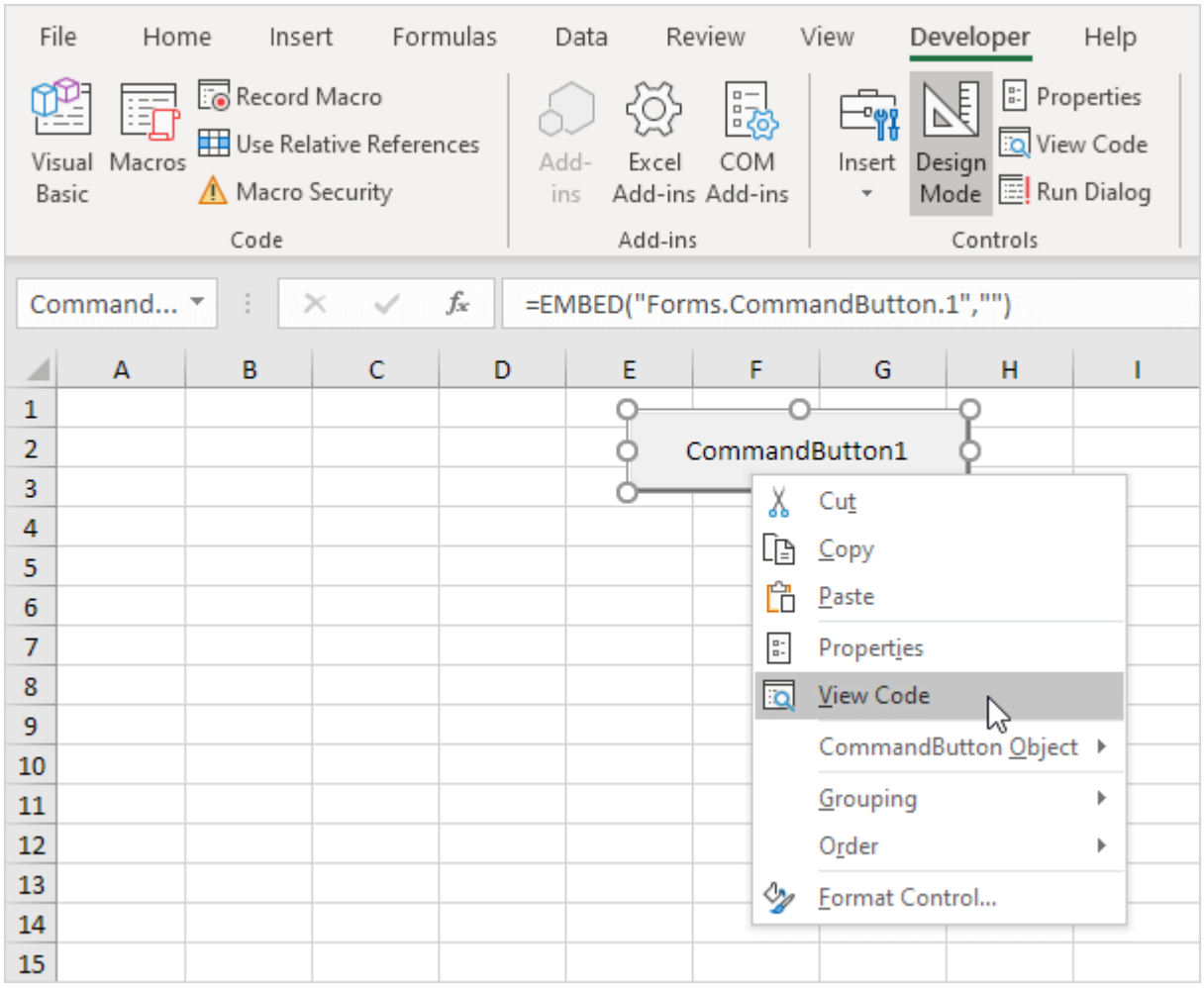

3. Drag a command button on your worksheet.

## Assign a Macro

To assign a macro (one or more code lines) to the command button, execute the following steps.

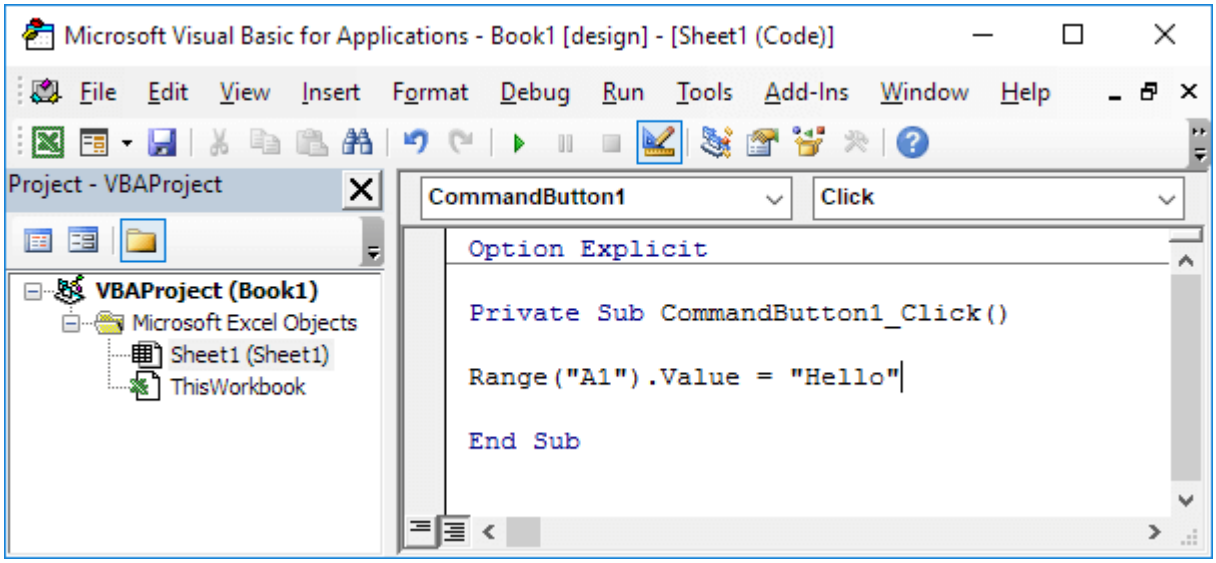
1. Right click CommandButton1 (make sure Design Mode is selected).

2. Click View Code.




The Visual Basic Editor appears.

3. Place your cursor between Private Sub CommandButton1_Click() and End Sub.
4. Add the code line shown below.



# ' en kommentarlinje

# Private Sub: så ses den ikke i makrooversigten

, og man kan gemme i xlsx fil,

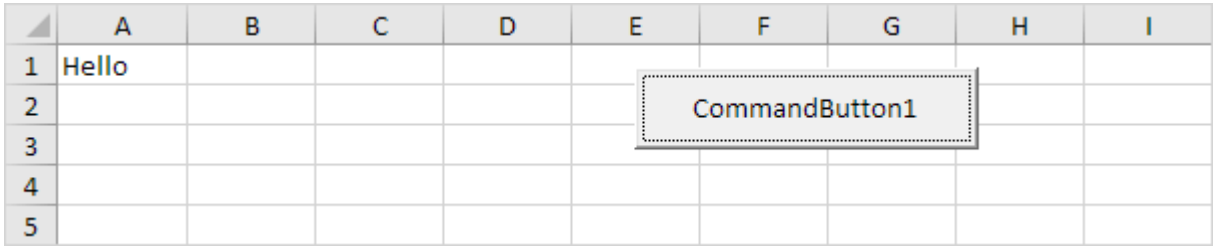
Uden private er den public, ses udefra, man kan kun gemme filen som .xlsm fil,

Microsoft Excel-regneark med aktiverede makroer (.xlsm)

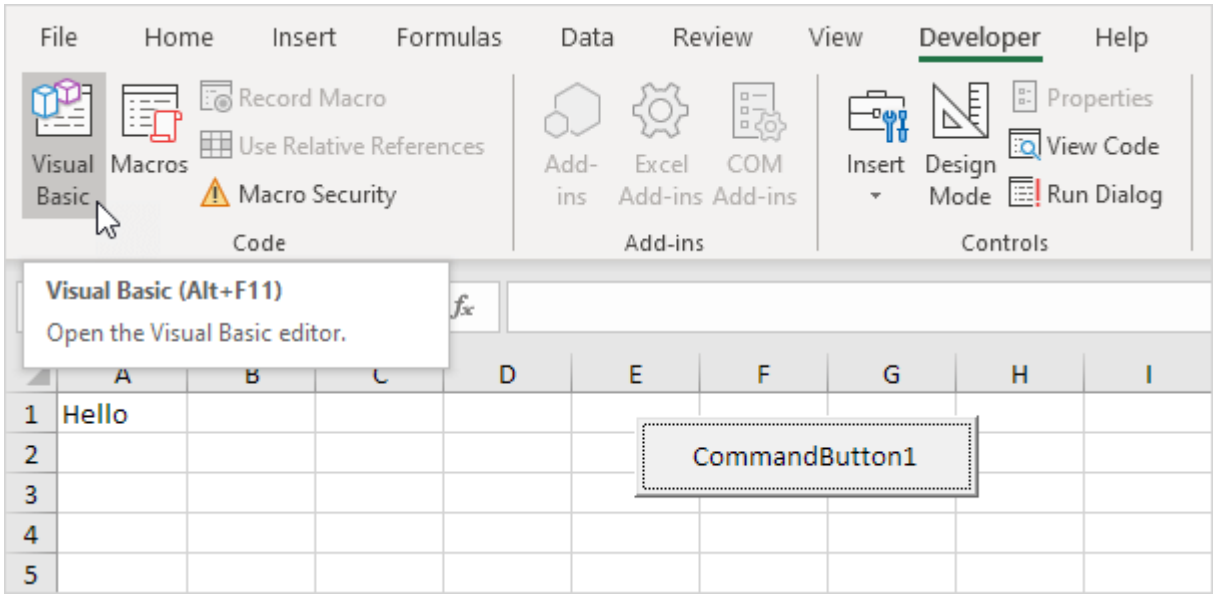
Note: the window on the left with the names Sheet1 (Sheet1) and ThisWorkbook is called the Project Explorer. If the Project Explorer is not visible, click View, Project Expl

# Microsoft Excel-regneark med aktiverede makroer

(.xlsm)orer. If the Code window for Sheet1 is not visible, click Sheet1 (Sheet1). You can ignore the Option Explicit statement for now.
5. Close the Visual Basic Editor.
6. Click the command button on the sheet (make sure Design Mode is deselected).
Result:




Congratulations. You've just created a macro in Excel!
Visual Basic Editor:
To open the Visual Basic Editor, on the Developer tab, click Visual Basic.

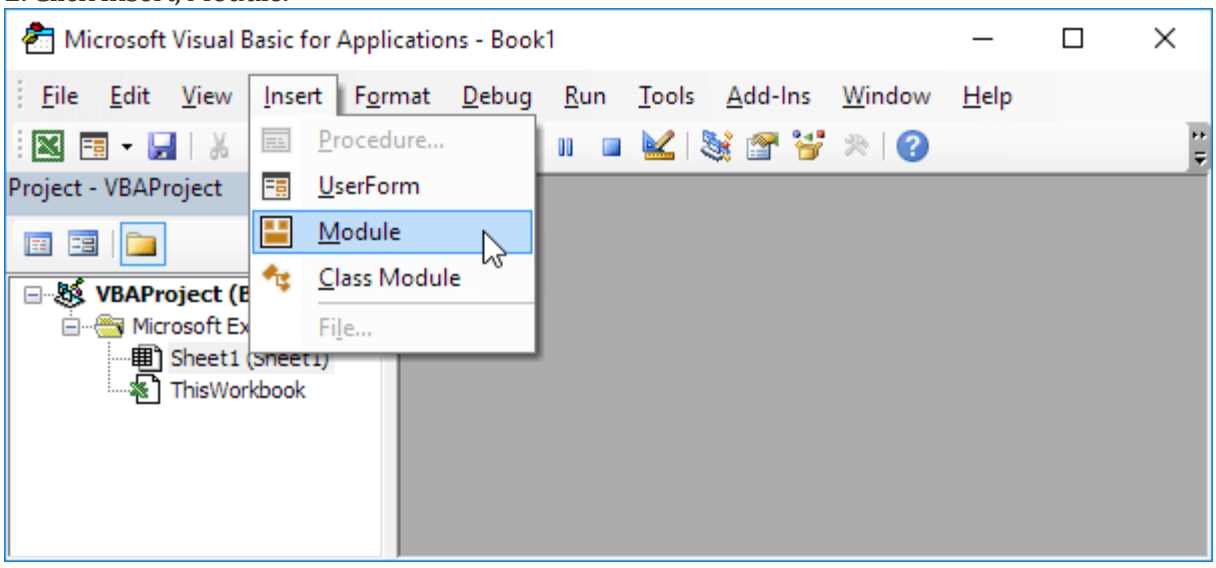



The Visual Basic Editor appears.

# Run Code from a Module

you might find it difficult to decide where to put your VBA code. The Create a Macro chapter illustrates how to run code by clicking on a command button. This example teaches you how to run code from a module.

1. Open the Visual Basic Editor.
2. Click Insert, Module.



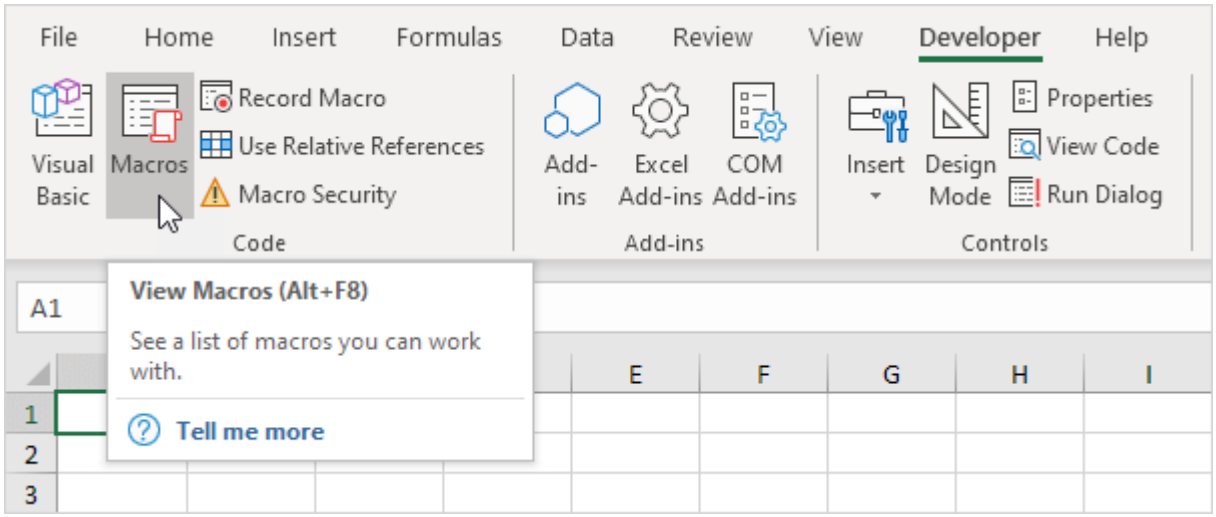
3. Create a procedure (macro) called Cyan, som skifter baggrundsfarve ved kørsel

```
Sub Cyan()


End Sub
```

a procedure is either a sub or a function. Learn more about functions and subs here, if you like.
The sub changes the background color of your worksheet to cyan. To achieve this, få skrevet:

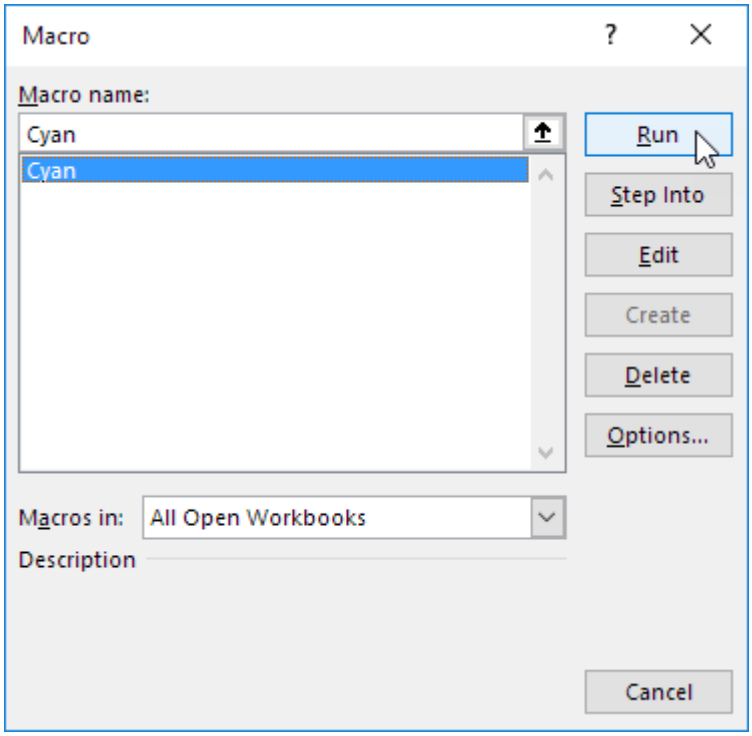
```
Sub Cyan()
Cells.Interior.ColorIndex = 28
End Sub
```

Note: instead of ColorIndex number 28 (cyan), you can use any ColorIndex number.
To run the procedure, execute the following steps.
5. Click Macros.

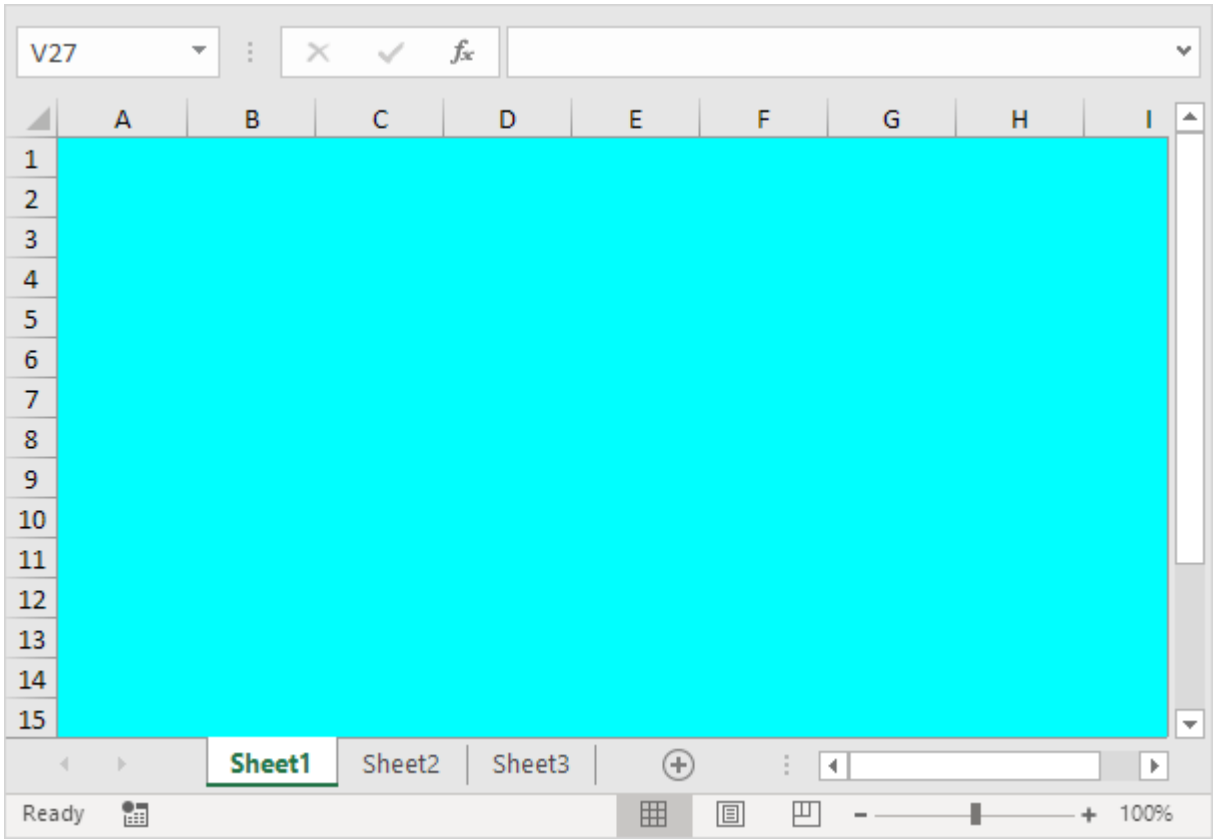



1. Select Cyan and click Run.

# Forrige sub var private, derfor ses den ikke i makrooversigten




Result:



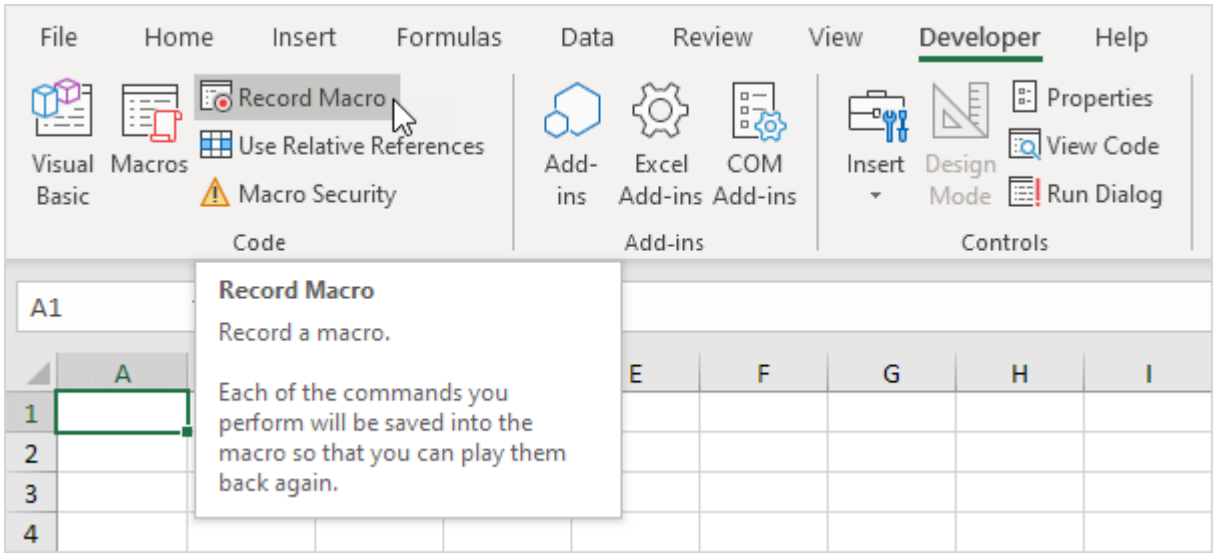

Note: code placed into a module is available to the whole workbook. That means you can select Sheet2 or Sheet3 and change the background color of these sheets as well. The Add a Macro to the Toolbar program illustrates how to make a macro available to all your workbooks (Excel files). Remember, code placed on a sheet (assigned to a command button) is only available for that particular sheet.

## Macro Recorder

The Macro Recorder, a very useful tool included in Excel VBA, 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.

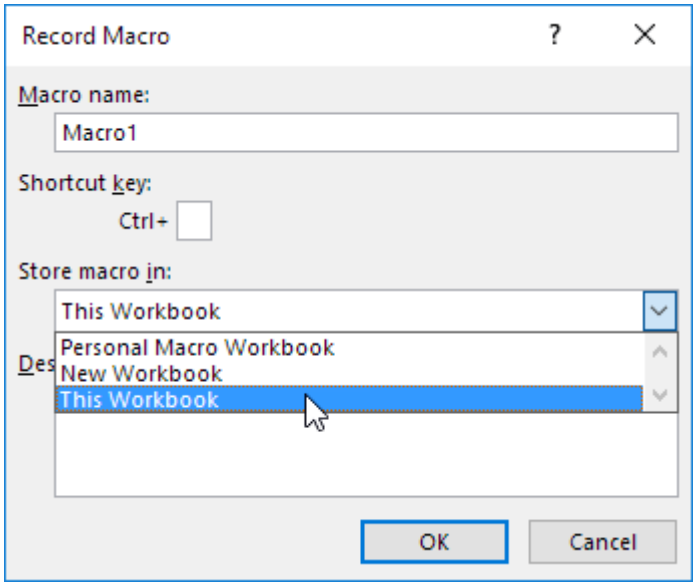
Record a Macro

1. On the Developer tab, click Record Macro.




2. Enter a name.

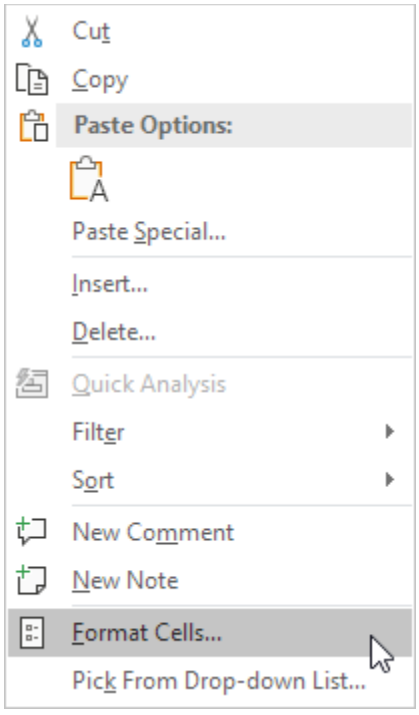
3. Select This Workbook from the drop-down list. As a result, the macro will only be available in the current workbook.



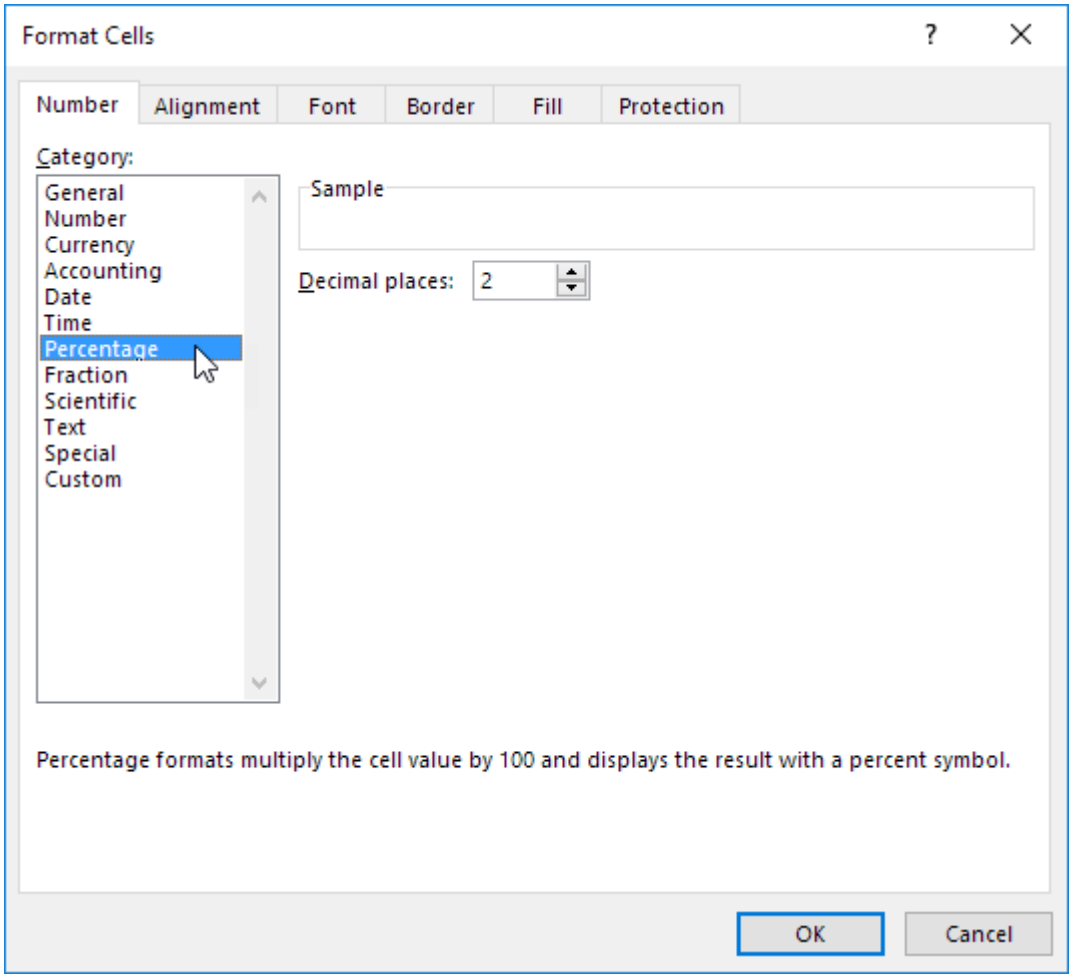

Note: if you store your macro in Personal Macro Workbook, the macro will be available to all your workbooks (Excel files). This is possible because Excel stores your macro in a hidden workbook that opens automatically when Excel starts. If you store your macro in New Workbook, the macro will only be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

4. Click OK.

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

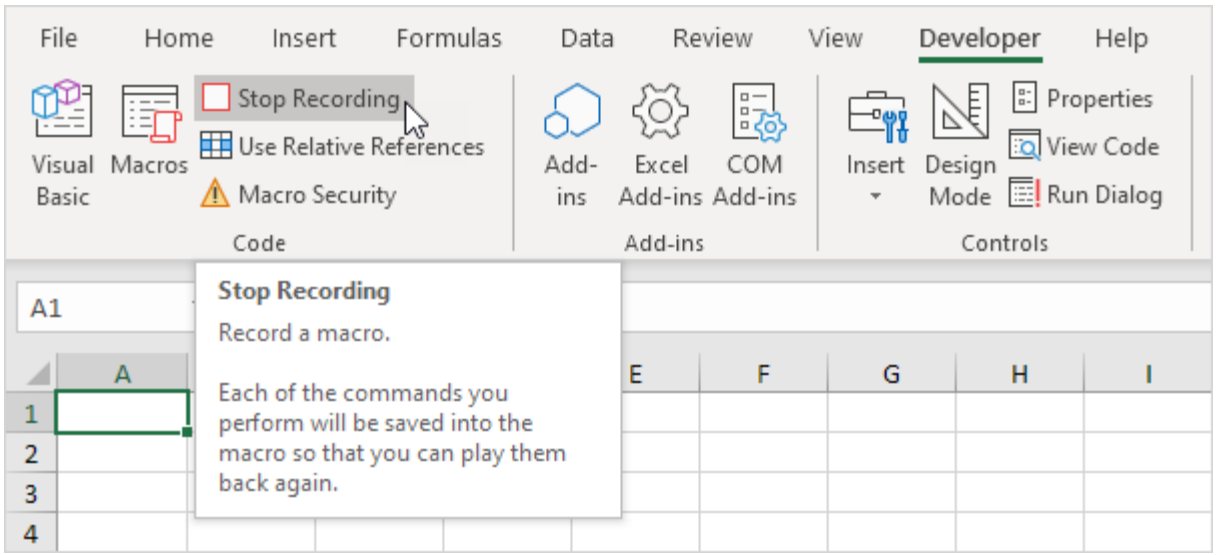



6. Select Percentage.




7. Click OK.

8. Finally, click Stop Recording.




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

Run a Recorded Macro

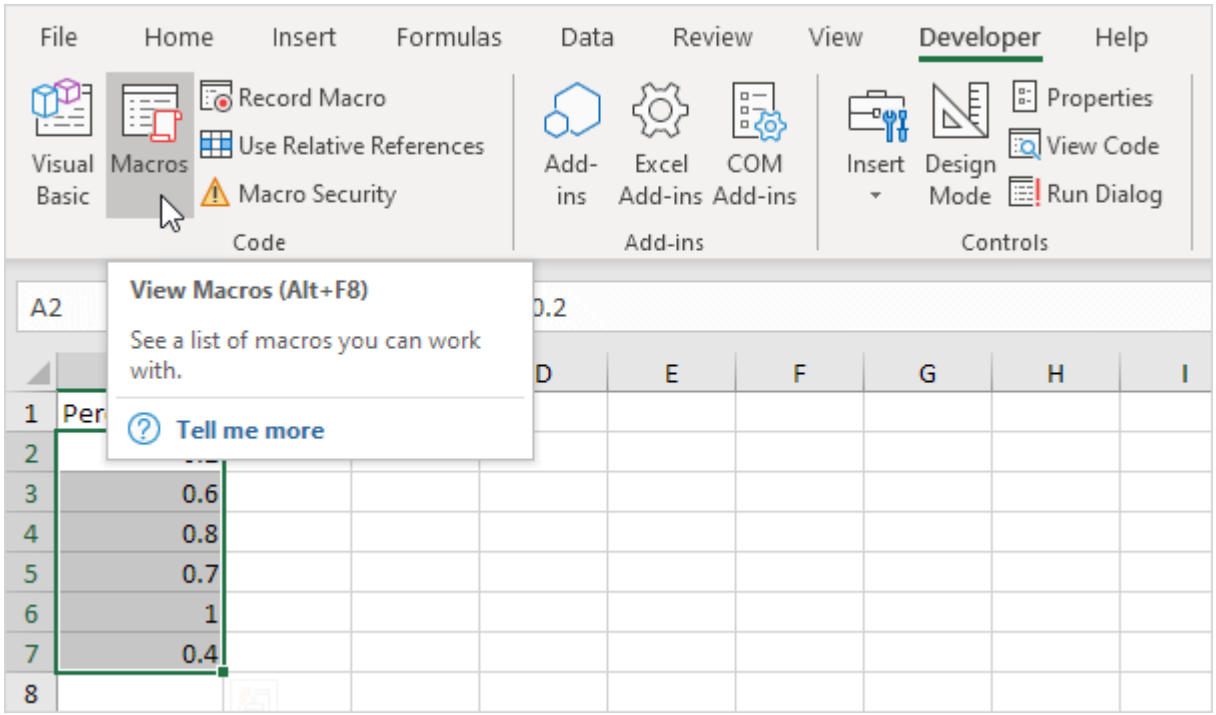
Now we'll test the macro to see if it can change the number format to Percentage.

1. Enter some numbers between 0 and 1.

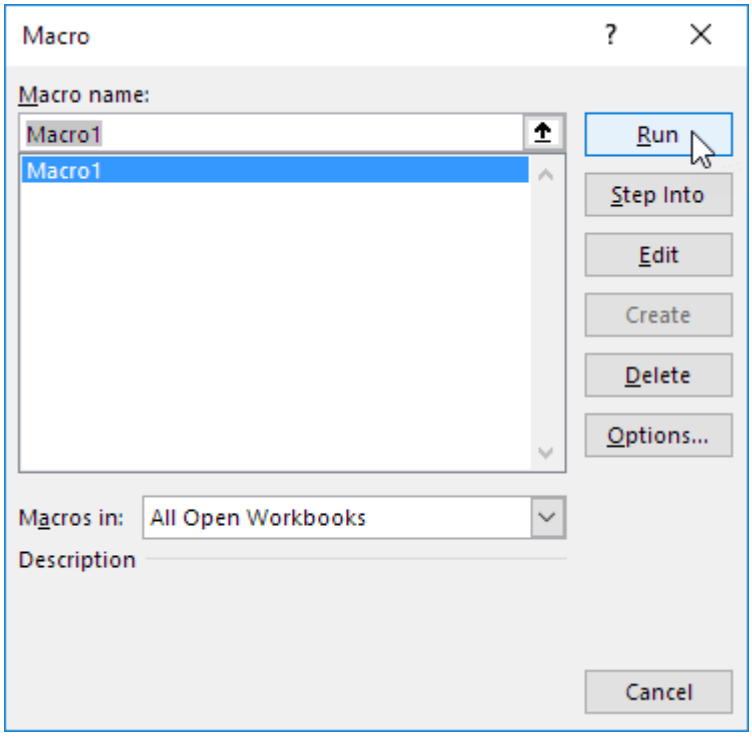
2. Select the numbers.

| | A | B |
|---|---|---|
| 1 | Percentages | |
| 2 | 0.2 | |
| 3 | 0.6 | |
| 4 | 0.8 | |
| 5 | 0.7 | |
| 6 | 1 | |
| 7 | 0.4 | |
| 8 | | |

3. On the Developer tab, click Macros.




4. Click Run.

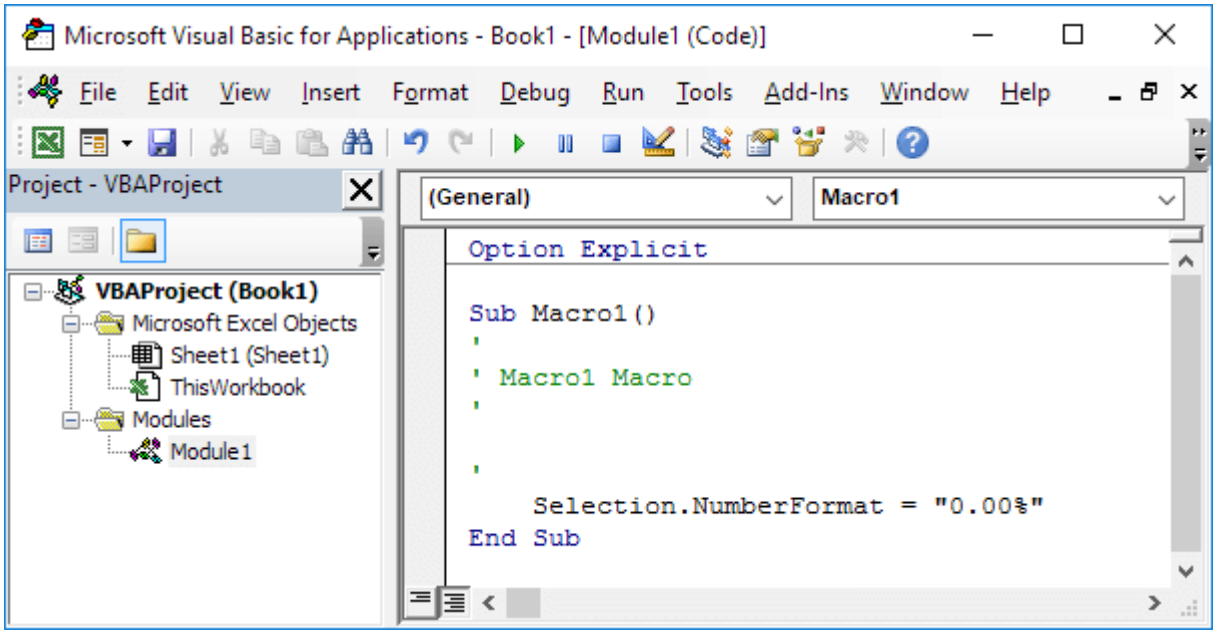



Result:

| | A | B |
|---|---|---|
| 1 | Percentages | |
| 2 | 20.00% | |
| 3 | 60.00% | |
| 4 | 80.00% | |
| 5 | 70.00% | |
| 6 | 100.00% | |
| 7 | 40.00% | |
| 8 | | |

# See the Macro

To take a look at the macro, open the Visual Basic Editor.




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 change the number format of cells on other sheets as well. Remember, code placed on a sheet (assigned to a command button) is only available for that particular sheet. You can ignore the Option Explicit statement for now.

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 required, which can slow your process down.

4/9 Completed! Learn much more about creating macros >

# Use Relative Cell-References

By default, Excel records macros in absolute mode. However, sometimes it is useful to record macros in relative mode. This program teaches you how to do this. If you don't know how to record a macro, we highly recommend you to read this example first.

**Recording in Absolute Mode, med faste cellereferencer**

To record a macro in absolute mode, execute the following steps.

1. First, click Record Macro.

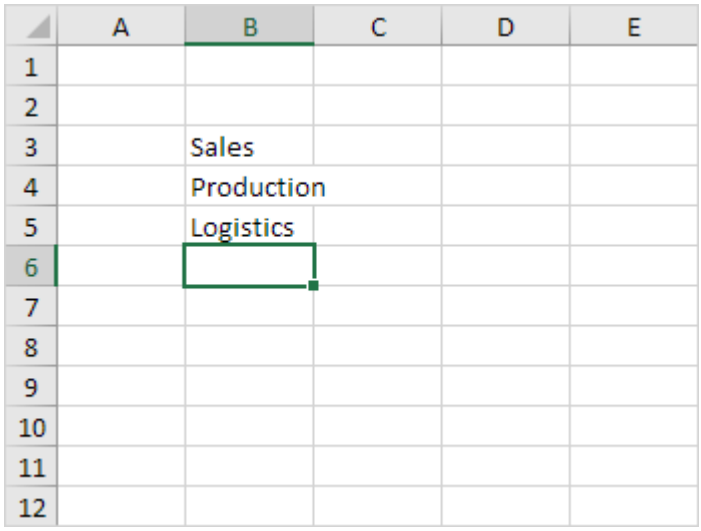
2. Next, select cell B3. Type Sales and press enter.

3. Type Production and press enter.

4. Type Logistics and press enter.7

Sales

Production
Logistics
Result:



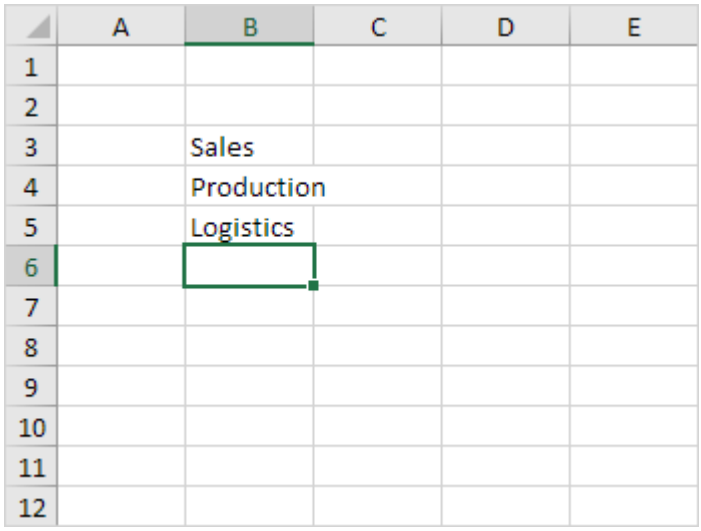
| | A | B | C | D | E |
|---|---|---|---|---|---|
| 1 | | | | | |
| 2 | | | | | |
| 3 | | Sales | | | |
| 4 | | Production | | | |
| 5 | | Logistics | | | |
| 6 | | | | | |
| 7 | | | | | |
| 8 | | | | | |
| 9 | | | | | |
| 10 | | | | | |
| 11 | | | | | |
| 12 | | | | | |

5. Click Stop Recording.

6. Empty Range("B3:B5").

7. Select any cell on the sheet and run the recorded macro.

Result:



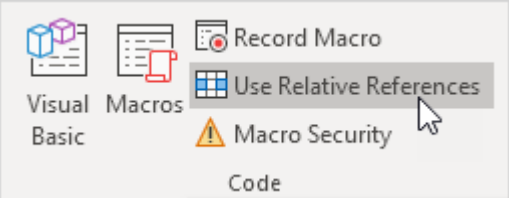
| | A | B | C | D | E |
|---|---|---|---|---|---|
| 1 | | | | | |
| 2 | | | | | |
| 3 | | Sales | | | |
| 4 | | Production | | | |
| 5 | | Logistics | | | |
| 6 | | | | | |
| 7 | | | | | |
| 8 | | | | | |
| 9 | | | | | |
| 10 | | | | | |
| 11 | | | | | |
| 12 | | | | | |

A macro recorded in absolute mode always produces the same result, i same celler

**Recording in Relative Mode**

Wouldn't it be nice to place these words anywhere on the sheet automatically? Not just Range("B3:B5"). This would make the macro much more flexible. Solution: record the macro in relative mode.

1. Select "Use Relative References".




2. First, select any single cell (for example, cell B8).

3. Next, click Record Macro.

4. Type Sales and press enter.

5. Type Production and press enter.

6. Type Logistics and press enter.

Result:

| | A | B | C | D | E |
|---|---|---|---|---|---|
| 1 | | | | | |
| 2 | | | | | |
| 3 | | | | | |
| 4 | | | | | |
| 5 | | | | | |
| 6 | | | | | |
| 7 | | | | | |
| 8 | | Sales | | | |
| 9 | | Production | | | |
| 10 | | Logistics | | | |
| 11 | | | | | |
| 12 | | | | | |

7. Click Stop Recording.

8. Select any other cell (for example, cell D4) and run the recorded macro.

Result:

| | A | B | C | D | E |
|---|---|---|---|---|---|
| 1 | | | | | |
| 2 | | | | | |
| 3 | | | | | |
| 4 | | | | Sales | |
| 5 | | | | Production | |
| 6 | | | | Logistics | |
| 7 | | | | | |
| 8 | | Sales | | | |
| 9 | | Production | | | |
| 10 | | Logistics | | | |
| 11 | | | | | |
| 12 | | | | | |

Excel places the words relative to the initial selected cell. That's why it's called recording in relative mode.

# 5/9 Completed! Learn much more about creating macros >

Go to Next Chapter: MsgBox

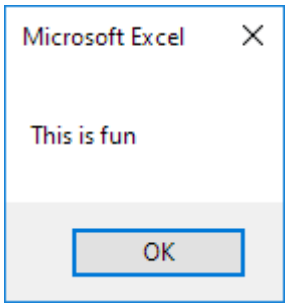
## MsgBox

The MsgBox is a dialog box in Excel VBA you can use to inform the users of your program. Place a command button on your worksheet and add the following code lines:

1. A simple message.

```
MsgBox "This is fun"
```

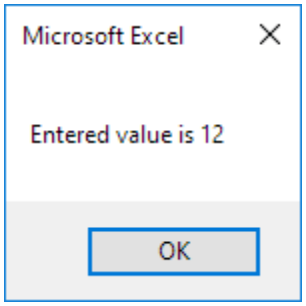
Result when you click the command button on the sheet:




2. A little more advanced message. First, enter a number into cell A1.

```
MsgBox "Entered value is " & Range("A1").Value
```

Result when you click the command button on the sheet:

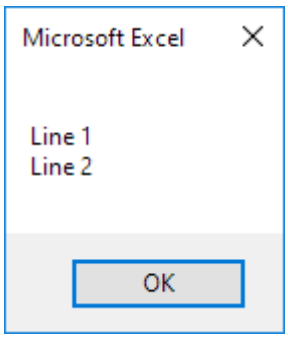



Note: use the & operator to concatenate (join) two strings. Although Range("A1").Value is not a string, it works here.

3. To start a new line in a message, use vbNewLine.

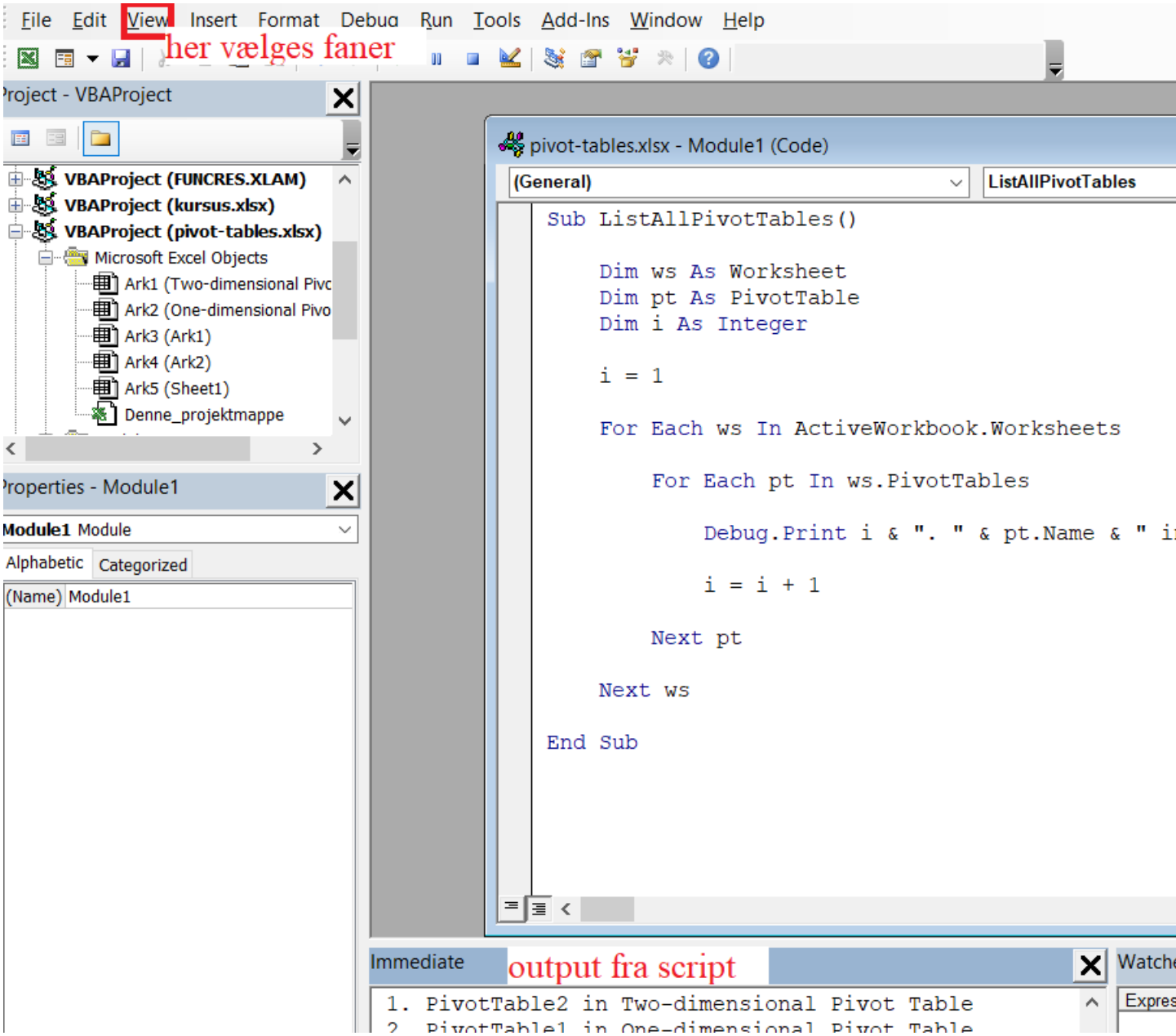
MsgBox "Line 1" & vbNewLine & "Line 2"

Result when you click the command button on the sheet:




Join over 1 million Excel learners by following Excel Easy on Facebook, X, and LinkedIn.

1/3 Completed! Learn much more about msgboxes >




Excel VBA (Visual Basic for Applications) is the name of the programming language of Excel.

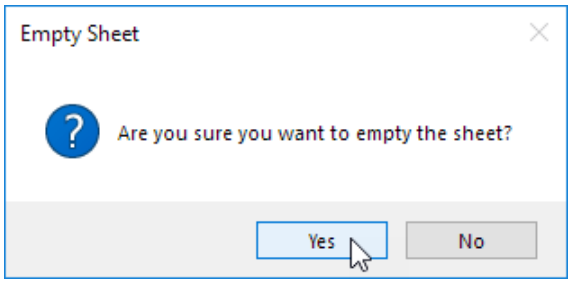
# The MsgBox function in Excel VBA can return a result while a simple MsgBox cannot.

Situation:

| | A | B | C | D | E | F | G | H | I |
|---|---|---|---|---|---|---|---|---|---|
| 1 | | | | | | | | | |
| 2 | | Empty Sheet | | | | | | | |
| 3 | 5 | | | | | | Month | | |
| 4 | | | | | -2000 | | | | |
| 5 | | | | | | | | | |
| 6 | | | Sales | | | | | | |
| 7 | | | | 10 | | 466 | | | |
| 8 | | | | | | | | | |
| 9 | 6548 | | | | | | | 45454 | |
| 10 | | | | | | | | | |
| 11 | | | | | | | | | |

Place a command button on your worksheet and add the following code lines:

```
Private Sub CommandButton1_Click()
  Dim answer As Integer

   answer = MsgBox("Are you sure you want to empty the sheet?", vbYesNo + vbQuestion, "Empty Sheet")

   If answer = vbYes Then
      Cells.ClearContents
   Else
      'do nothing
   End If
End Sub
```

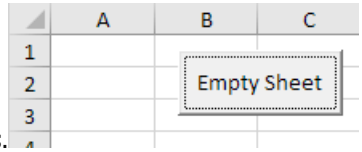



The MsgBox function, when using parentheses, has three arguments. The first part is used for the message in the message box. Use the second part to specify which buttons and icons you want to appear in the message box. The third part is displayed in the title bar of the message box.

answer = MsgBox("Are you sure you want to empty the sheet?", vbYesNo + vbQuestion, "Empty Sheet")

Note: Place your cursor on vbYesNo in the Visual Basic Editor and click F1 to see which other buttons and icons you can use. Instead of the constants vbYesNo and vbQuestion, you can also use the corresponding values 4 and 32.

3. If the user clicks the Yes button, Excel VBA empties the sheet. If the user clicks the No button, nothing happens. Add the following code lines to achieve this.




## ****Workbook and Worksheet Object

Learn more about the Workbook and Worksheet object in Excel VBA.

# Object Hierarchy

In Excel VBA, an object can contain another object, and that object can contain another object, etc. In other words, Excel VBA programming involves working with an object hierarchy. This probably sounds quite confusing, but we will make it clear.

The mother of all objects is Excel itself. We call it the Application object. The application object contains other objects. For example, the Workbook object (Excel file). This can be any workbook you have created. The Workbook object contains other objects, such as the Worksheet object. The Worksheet object contains other objects, such as the Range object.

The Create a Macro chapter illustrates how to run code by clicking on a command button. We used the following code line:

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

but what we really meant was:

```
Application.Workbooks("create-a-macro").Worksheets(1).Range("A1").Value = "Hello"
```

Note: the objects are connected with a dot. Fortunately, we do not have to add a code line this way. That is because we placed our command button in create-a-macro.xlsm, on the first worksheet. Be aware that if you want to change things on different worksheets, you have to include the Worksheet object. Read on.

# Collections

You may have noticed that Workbooks and Worksheets are both plural. That is because they are collections. The Workbooks collection contains all the Workbook objects that are currently open. The Worksheets collection contains all the Worksheet objects in a workbook.


You can refer to a member of the collection, for example, a single Worksheet object, in three ways.

1. Using the worksheet name.

Worksheets("Sales").Range("A1").Value = "Hello"

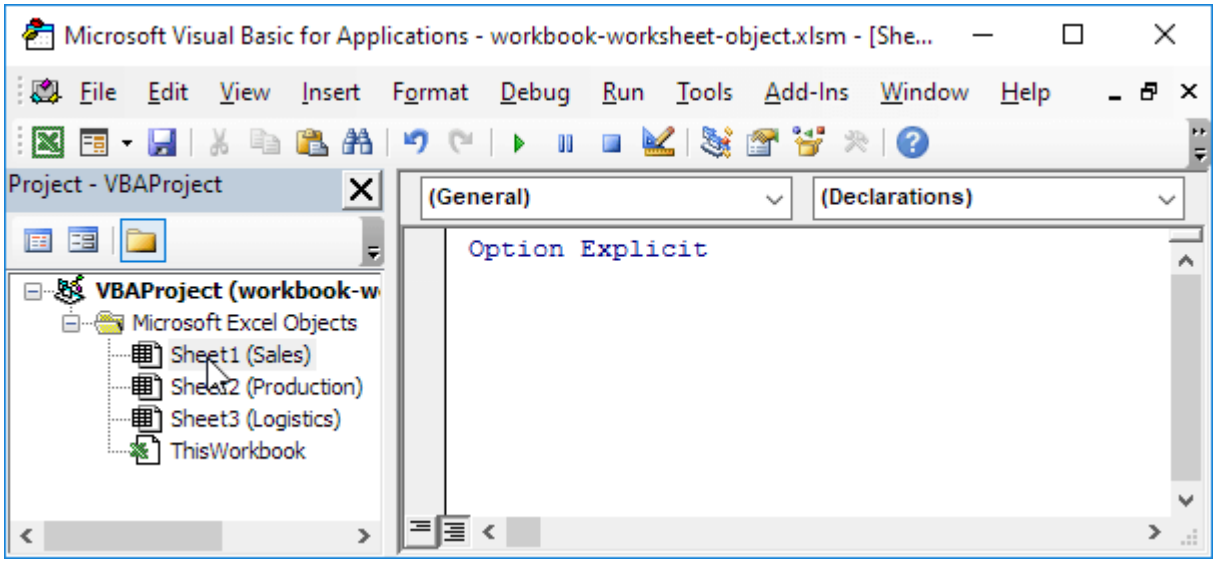
2. Using the index number (1 is the first worksheet starting from the left).

Worksheets(1).Range("A1").Value = "Hello"

3. Using the CodeName.

Sheet1.Range("A1").Value = "Hello"

To see the CodeName of a worksheet, open the Visual Basic Editor. In the Project Explorer, the first name is the CodeName. The second name is the worksheet name (Sales).




Note: the CodeName remains the same if you change the worksheet name or the order of your worksheets so this is the safest way to reference a worksheet. Click View, Properties Window to change the CodeName of a worksheet. There is one disadvantage, you cannot use the CodeName if you reference a worksheet in a different workbook.

# Properties and Methods

Now let's take a look at some properties and methods of the Workbooks and Worksheets collection. Properties are something which a collection has (they describe the collection), while methods do something (they perform an action with a collection).

Place a command button on your worksheet and add the code lines:

1. The Add method of the Workbooks collection creates a new workbook.

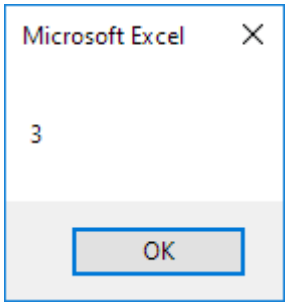
```
Workbooks.Add
```

Note: the Add method of the Worksheets collection creates a new worksheet.

2. The Count property of the Worksheets collection counts the number of worksheets in a workbook.

```
MsgBox Worksheets.Count
```

Result when you click the command button on the sheet:




Note: the Count property of the Workbooks collection counts the number of active workbooks.

1/8 Completed! Learn more about books and sheets >
Go to Next Chapter: Range Object

# Range Object

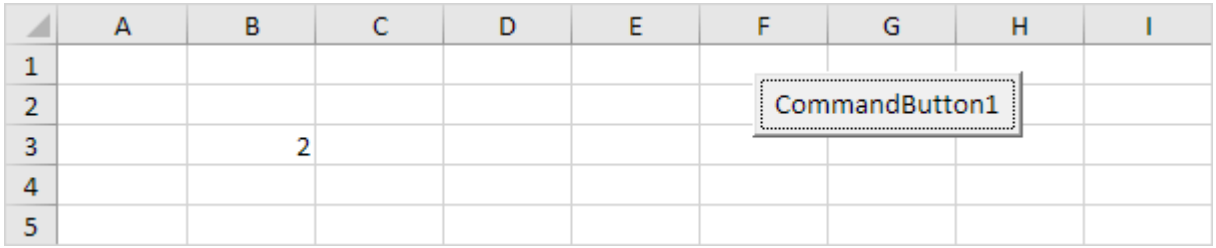
The Range object, which is the representation of a cell (or cells) on your worksheet, is the most important object of Excel VBA. This chapter gives an overview of the properties and methods of the Range object. Properties are something which an object has (they describe the object), while methods do something (they perform an action with an object).

Range Examples

Place a command button on your worksheet and add the following code line:

```
Range("B3").Value = 2
```

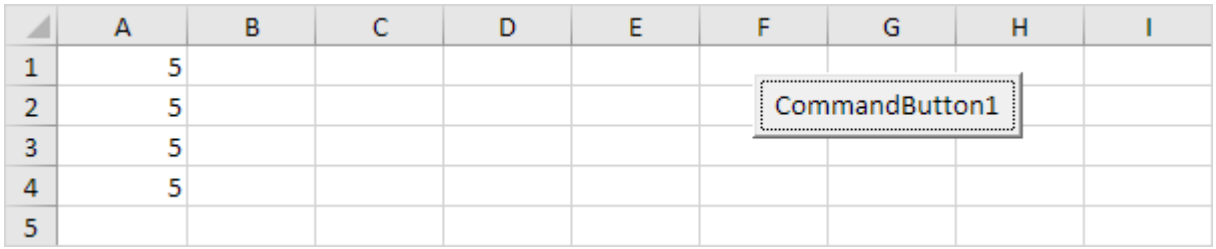
Result when you click the command button on the sheet:



Code:

```
Range("A1:A4").Value = 5
```

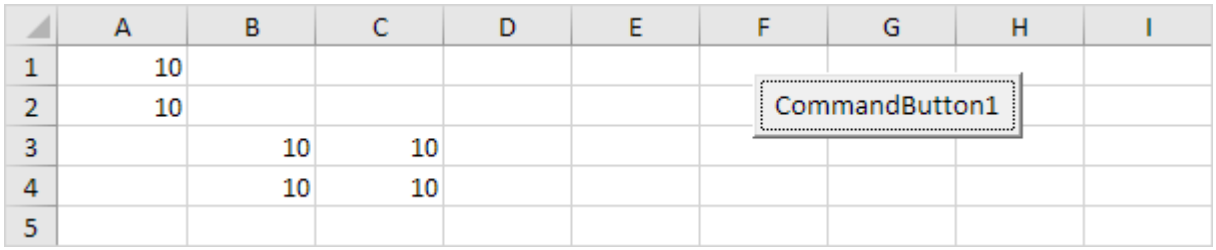
Result:




Code:

```
Range("A1:A2,B3:C4").Value = 10
```

Result:




Note: to refer to a named range in your Excel VBA code, use a code line like this:

```
Range("Prices").Value = 15
```

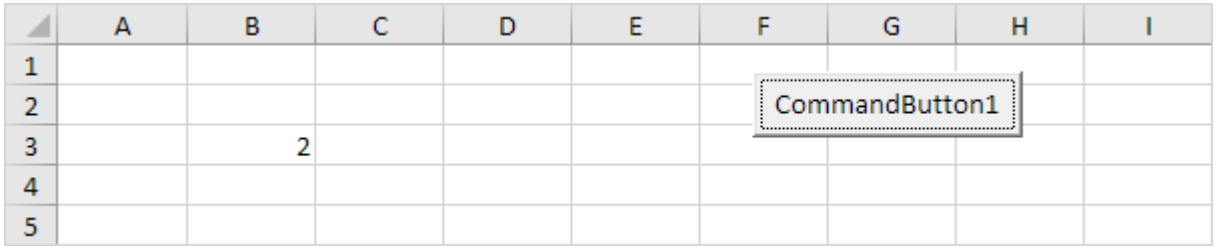
Cells

## Instead of Range, you can also use Cells. Using Cells is particularly useful when you want to loop through ranges.

Code:

```
Cells(3, 2).Value = 2
```

Result:

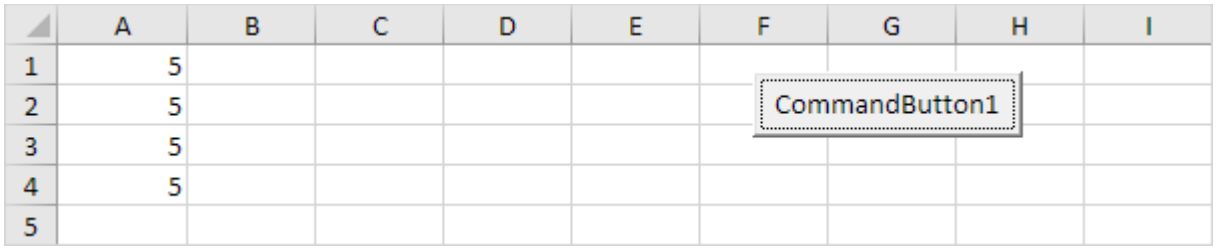



Explanation: Excel VBA enters the value 2 into the cell at the intersection of row 3 and column 2.

Code:

Range(Cells(1, 1), Cells(4, 1)).Value = 5

Result:



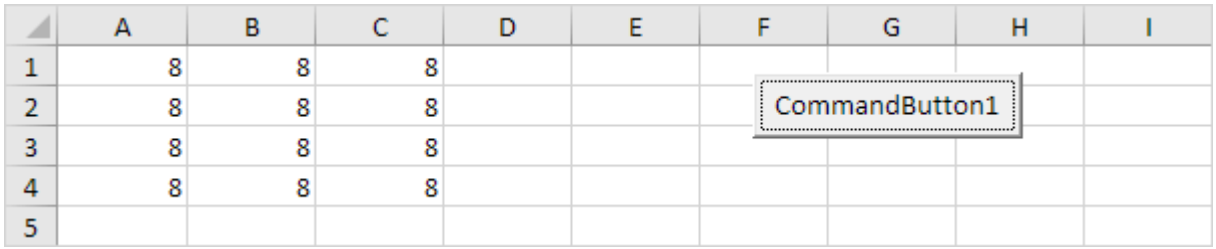

Declare a Range Object

You can declare a Range object by using the keywords Dim and Set.

Code:

```
Dim example As Range
Set example = Range("A1:C4")

example.Value = 8
```

Result:



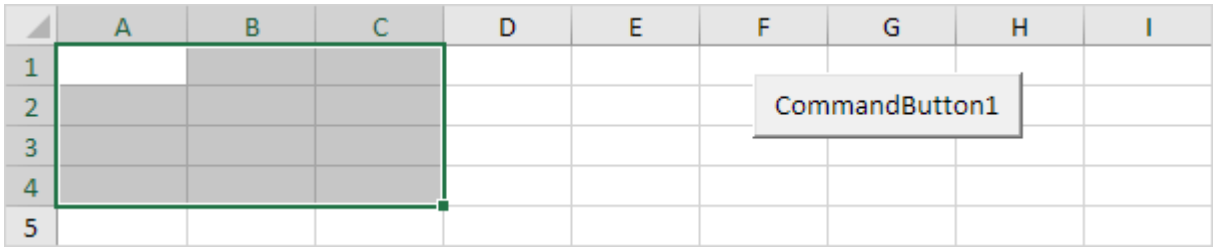

## Select

An important method of the Range object is the Select method. The Select method simply selects a range.

Code:

```
Dim example As Range
Set example = Range("A1:C4")

example.Select
```

Result:



Note: to select cells on a different worksheet, you have to activate this sheet first. For example, the following code lines select cell B7 on the third worksheet from the left.

```
Worksheets(3).Activate
Worksheets(3).Range("B7").Select
```

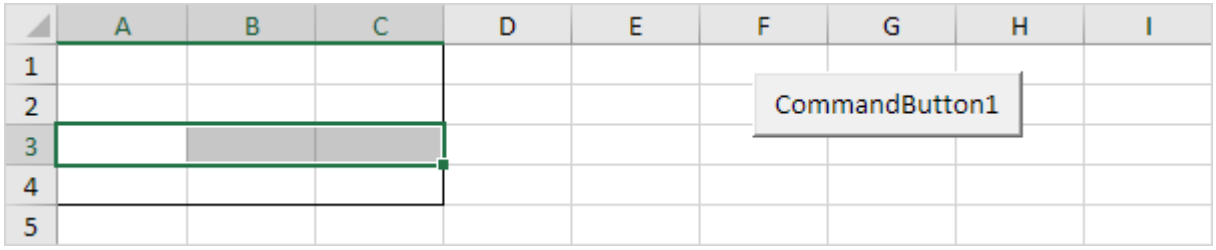
### Rows

The Rows property gives access to a specific row of a range.

Code:

```
Dim example As Range
Set example = Range("A1:C4")

example.Rows(3).Select
```

Result:




Note: borders in the image for illustration only.

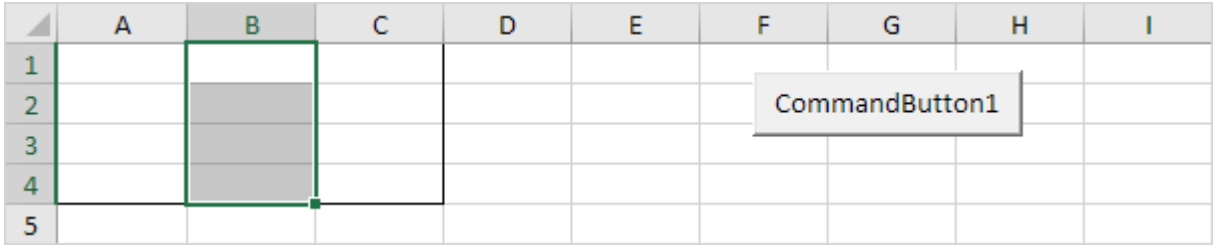
### Columns

The Columns property gives access to a specific column of a range.

Code:

```
Dim example As Range
Set example = Range("A1:C4")

example.Columns(2).Select
```

Result:




Note: borders in the image for illustration only.

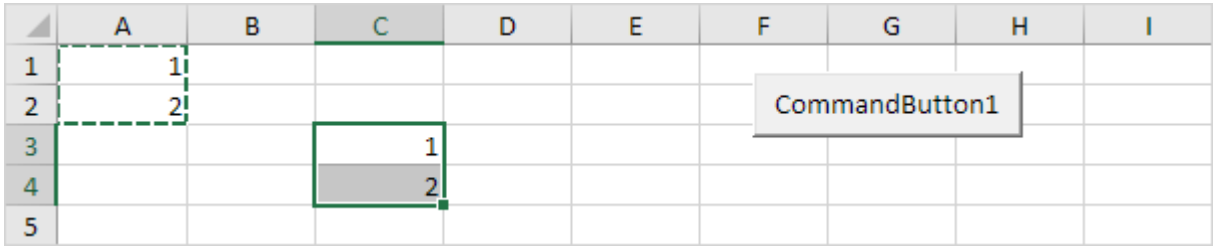
### Copy/Paste

The Copy and Paste methods are used to copy a range and to paste it somewhere else on the worksheet.

Code:

```
Range("A1:A2").Select
Selection.Copy

Range("C3").Select
ActiveSheet.Paste
```

Result:




Although this is allowed in Excel VBA, it is much better to use the code line below, which does exactly the same thing.

```
Range("C3:C4").Value = Range("A1:A2").Value
```

## Clear

To clear the content of an Excel range, you can use the ClearContents method.

```
Range("A1").ClearContents
```

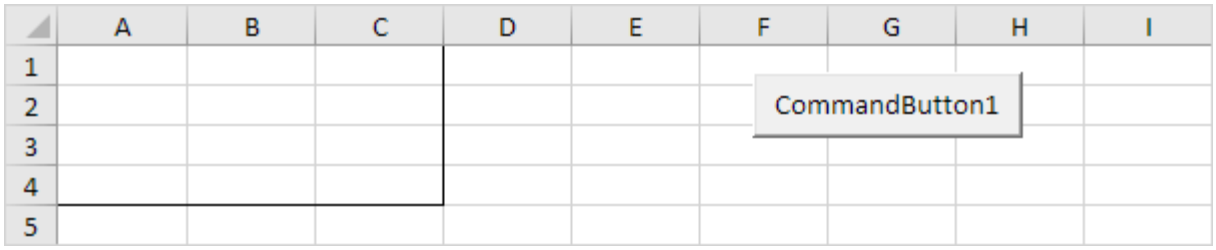
or simply use:

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

Note: use the Clear method to clear the content and format of a range. Use the ClearFormats method to clear the format only.

Count

## With the Count property, you can count the number of cells, rows and columns of a range.

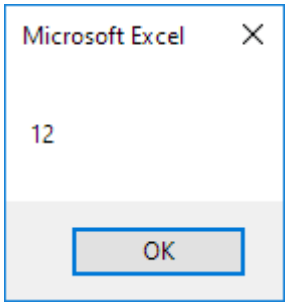



Note: borders in the image for illustration only.

Code:

```
Dim example As Range
Set example = Range("A1:C4")

MsgBox example.Count
```

Result:




Note: in a similar way, you can count the number of columns of a range.

1/14 Completed! Learn more about the range object >

# CurrentRegion

This example illustrates the CurrentRegion property in Excel VBA. The current region is a range bounded by any combination of blank rows and blank columns. Can you find the current region of cell A1?

| | A | B | C | D | E | F | G | H | I |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 4 | | | | | | | | |
| 2 | | 8 | | | | CommandButton1 | | | |
| 3 | 5 | | 6 | | | | | | |
| 4 | | | | | | | | | |
| 5 | | | | | | | | | |

Place a command button on your worksheet and add the following code line:

```
Range("A1").CurrentRegion.Select
```

Result when you click the command button on the sheet:

| | A | B | C | D | E | F | G | H | I |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 4 | | | | | | | | |
| 2 | | 8 | | | | CommandButton1 | | | |
| 3 | 5 | | 6 | | | | | | |
| 4 | | | | | | | | | |
| 5 | | | | | | | | | |

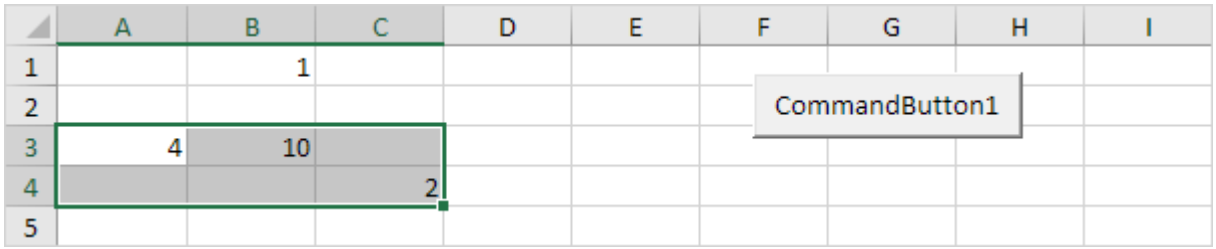
Here is another example. Can you find the current region of cell B3?

| | A | B | C | D | E | F | G | H | I |
|---|---|---|---|---|---|---|---|---|---|
| 1 | | 1 | | | | | | | |
| 2 | | | | | | CommandButton1 | | | |
| 3 | 4 | 10 | | | | | | | |
| 4 | | | 2 | | | | | | |
| 5 | | | | | | | | | |

Code:

```
Range("B3").CurrentRegion.Select
```

Result:




The value 1 in row 1 has no influence on the current region of cell B3. Row 2 is empty!

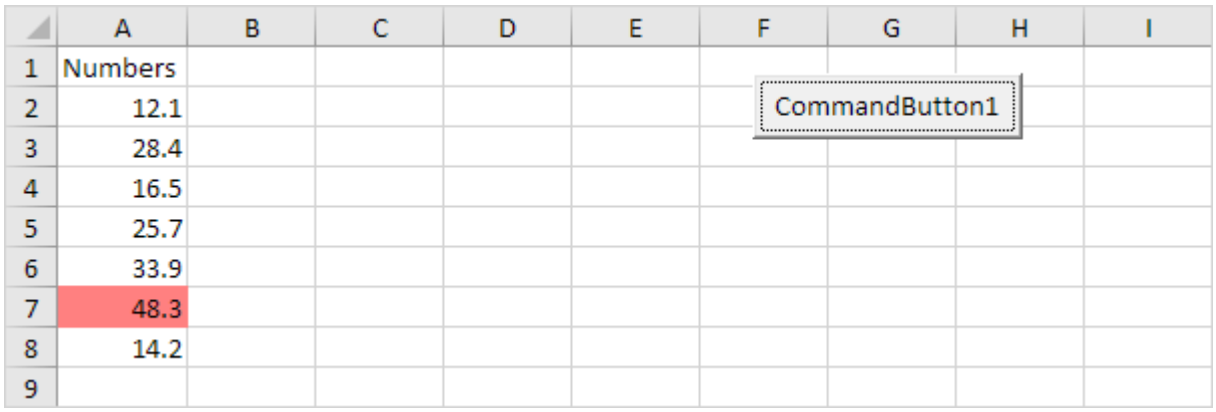
2/14 Completed! Learn more about the range object >

# Dynamic Range

Below we will look at a program in Excel VBA that colors the maximum value of a dynamic range.

Situation:

Each time we add a number and we click the command button, we want Excel VBA to color the maximum value of these numbers.




Place a command button on your worksheet and add the following code lines:

1. First, we declare one variable and two Range objects. One variable of type Double we call maximum. We call the Range objects rng and cell.

```
Dim maximum As Double, rng As Range, cell As Range
```

2. We add the line which changes the background color of all cells to 'No Fill'.

```
Cells.Interior.ColorIndex = 0
```

3. We initialize rng with the numbers. We use the CurrentRegion property for this. CurrentRegion is useful when we don't know the exact boundaries of a range in advance.

Set rng = Range("A1").CurrentRegion

4. We initialize maximum with the maximum value of the numbers. We use the worksheet function Max to find the maximum value.

maximum = WorksheetFunction.Max(rng)

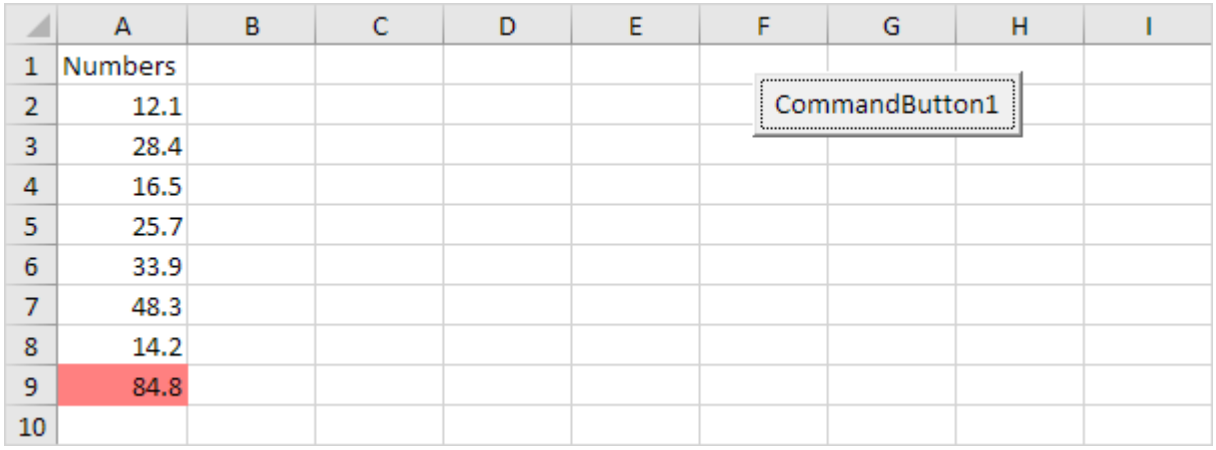
5. Finally, we color the maximum value. We use a For Each Next Loop.

```
For Each cell In rng
   If cell.Value = maximum Then cell.Interior.ColorIndex = 22
Next cell
```

Note: instead of ColorIndex number 22 (red), you can use any ColorIndex number.

6. Add a number.

Result when you click the command button on the sheet:






# The Resize property in Excel VBA resize a range a specific number of rows and columns

The Resize property always takes the top left cell of a range as the starting point.

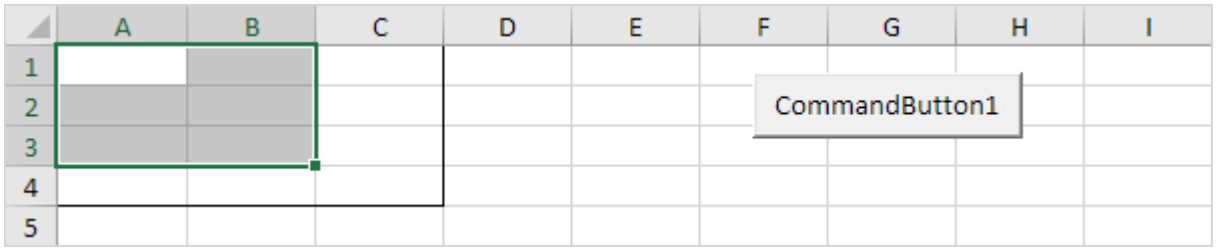
Contract

To contract the size of a range, specify a smaller number of rows or columns than the original range.

Place a command button on your worksheet and add the following code line:

Range("A1:C4").Resize(3, 2).Select

Result:

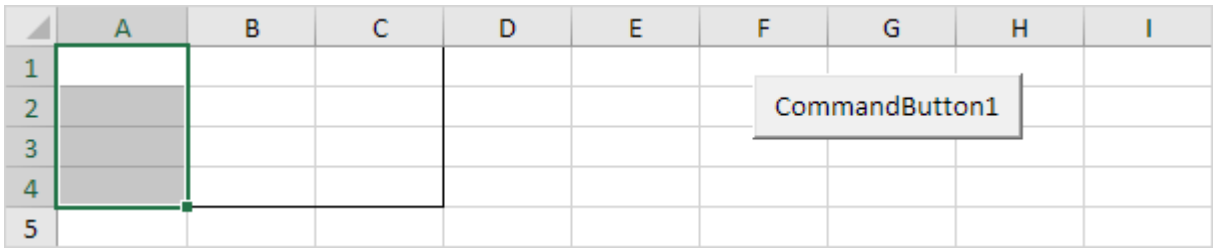



Explanation: this code line resizes Range("A1:C4") to 3 rows and 2 columns and selects this range (borders for illustration only).

Code line:

Range("A1:C4").Resize(, 1).Select

Result:




Explanation: this code line resizes Range("A1:C4") to 1 column (we omit any row specification) and selects this range.

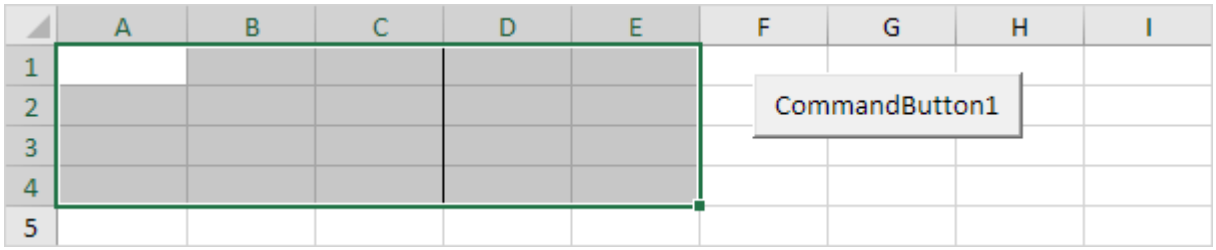
Expand

To expand the size of a range, specify a larger number of rows or columns than the original range.

Code line:

Range("A1:C4").Resize(, 5).Select ‘5columns

Result:

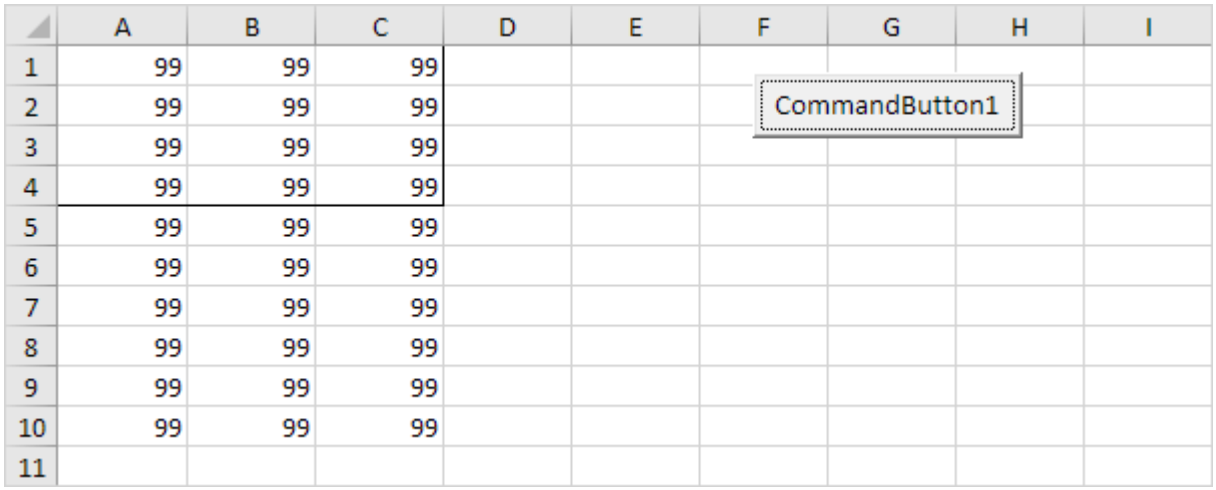



Explanation: this code line resizes Range("A1:C4") to 5 columns. The Select method selects Range("A1:E4").

Code line:

Range("A1:C4").Resize(10) = 99  ‘10rows

Result:




Explanation: this code line resizes Range("A1:C4") to 10 rows. Instead of using the Select method, you can work with the resized range directly. For example, set the values of the cells in the resized range to 99.

Join over 1 million Excel learners by following Excel Easy on Facebook, X, and LinkedIn.

4/14 Completed! Learn more about the range object >

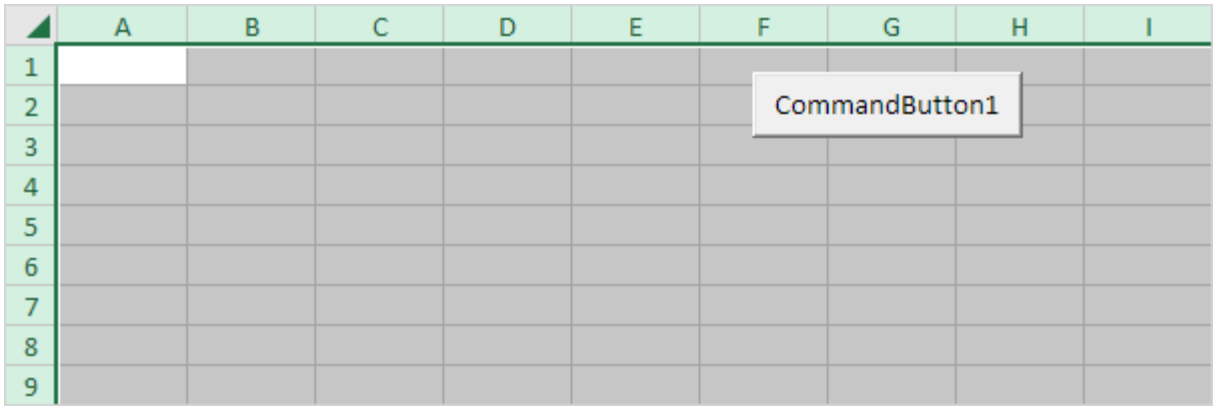
# Entire Rows and Columns

This example teaches you how to select entire rows and columns in Excel VBA. Are you ready?

Place a command button on your worksheet and add the following code lines:

1. The following code line selects the entire sheet.

```
Cells.Select
```

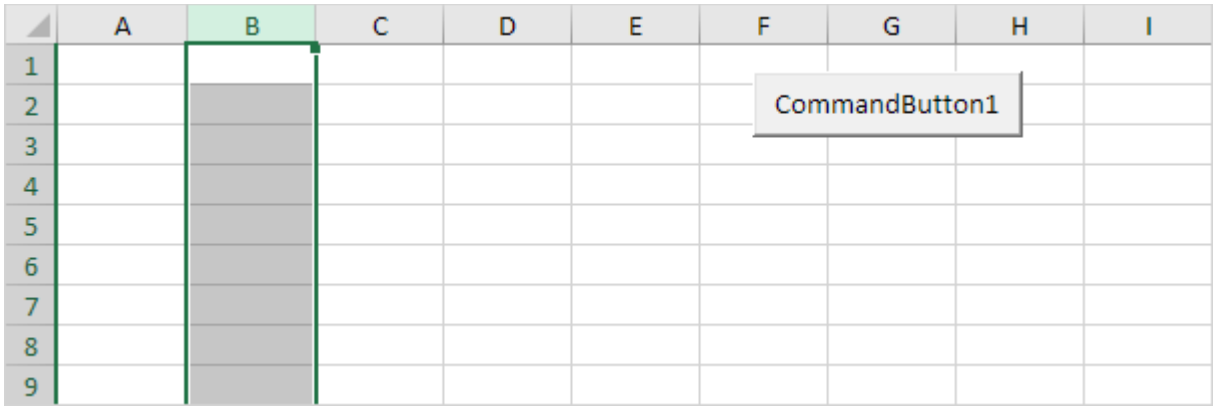



Note: because we placed our command button on the first worksheet, this code line selects the entire first sheet. To select cells on another worksheet, you have to activate this sheet first. For example, the following code lines select the entire second worksheet.

```
Worksheets(2).Activate
Worksheets(2).Cells.Select
```

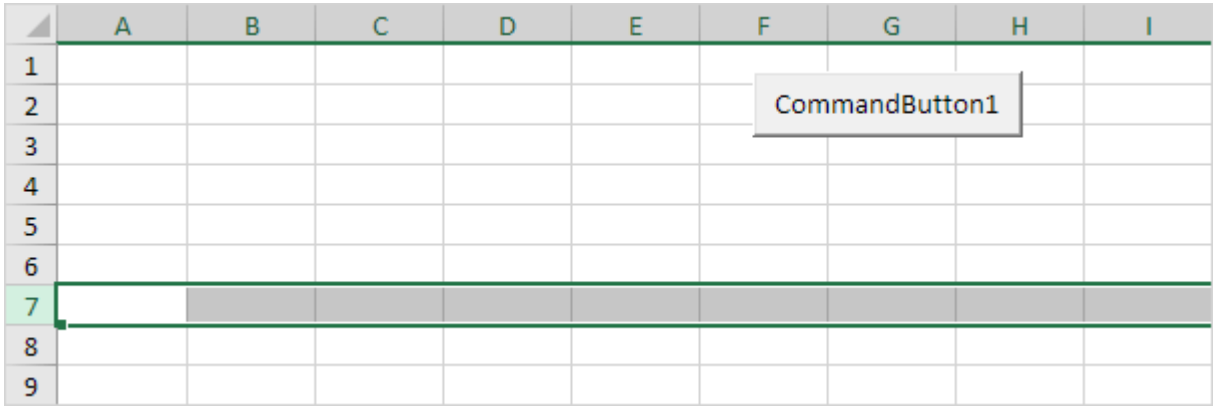
2. The following code line selects the second column.

Columns(2).Select



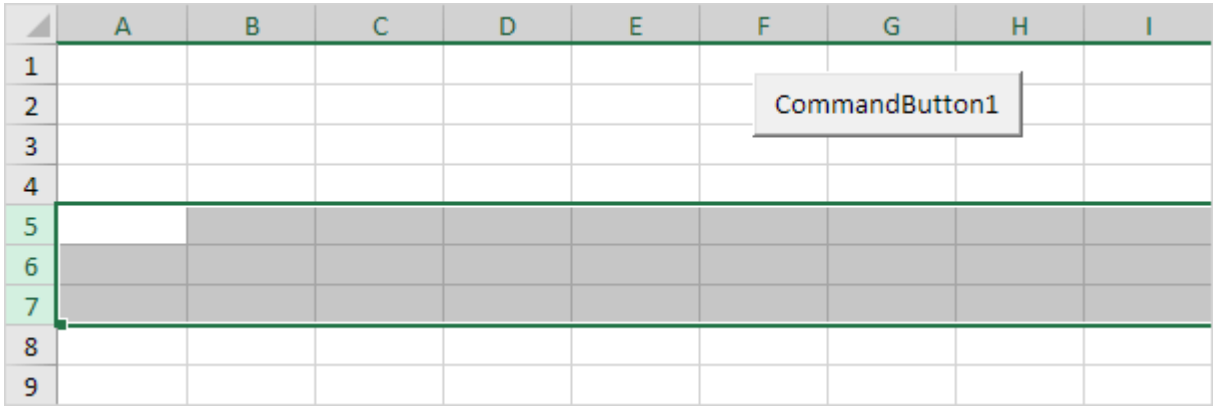

3. The following code line selects the seventh row.

Rows(7).Select



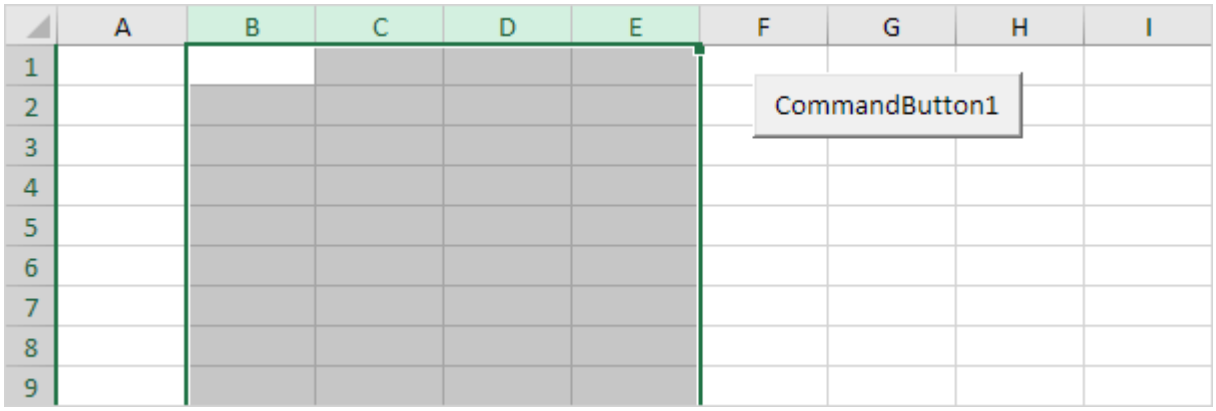

4. To select multiple rows, add a code line like this:

Rows("5:7").Select




5. To select multiple columns, add a code line like this:

Columns("B:E").Select

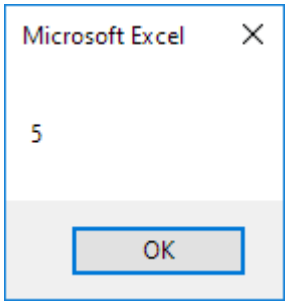



6. Be careful not to mix up the Rows and Columns properties with the Row and Column properties. The Rows and Columns properties return a Range object. The Row and Column properties return a single value.

Code line:

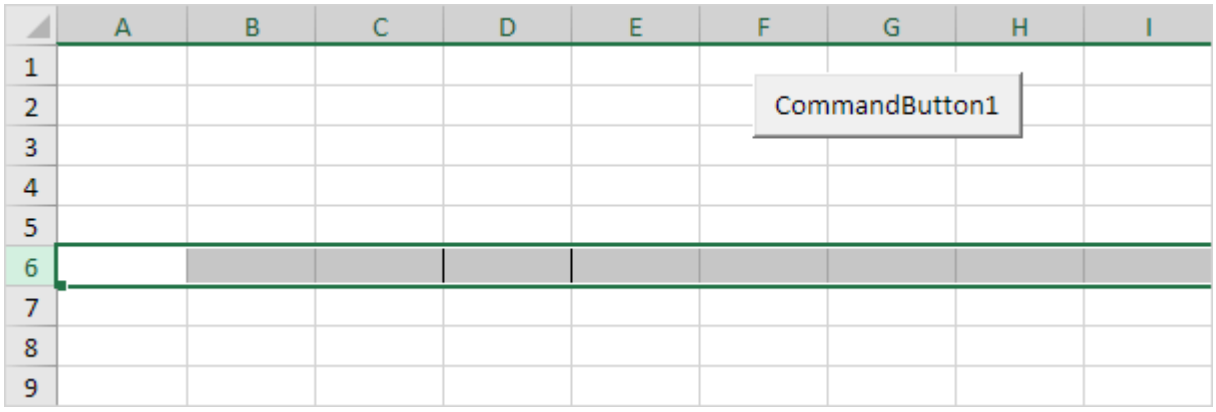
MsgBox Cells(5, 2).Row

Result:




7. Select cell D6. The following code line selects the entire row of the active cell.

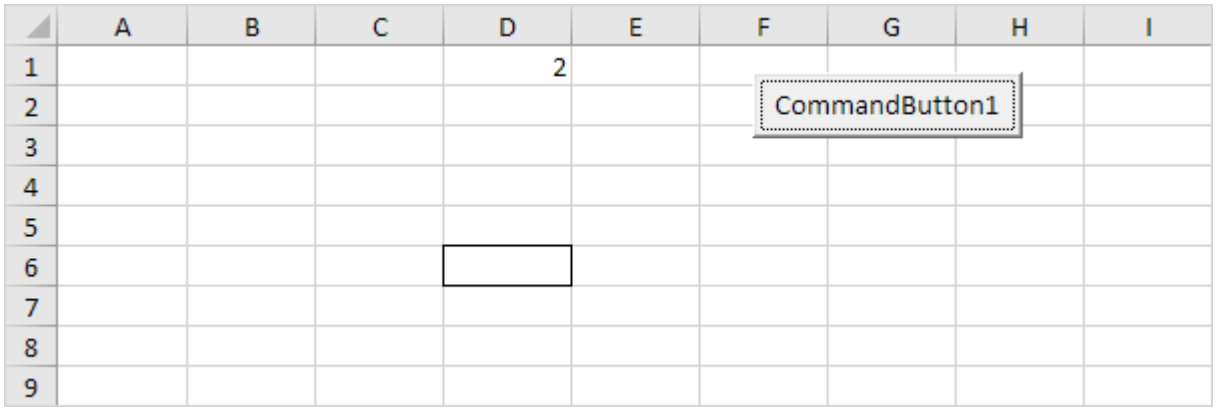
ActiveCell.EntireRow.Select




Note: borders in the image for illustration only.

8. Select cell D6. The following code line enters the value 2 into the first cell of the column that contains the active cell.

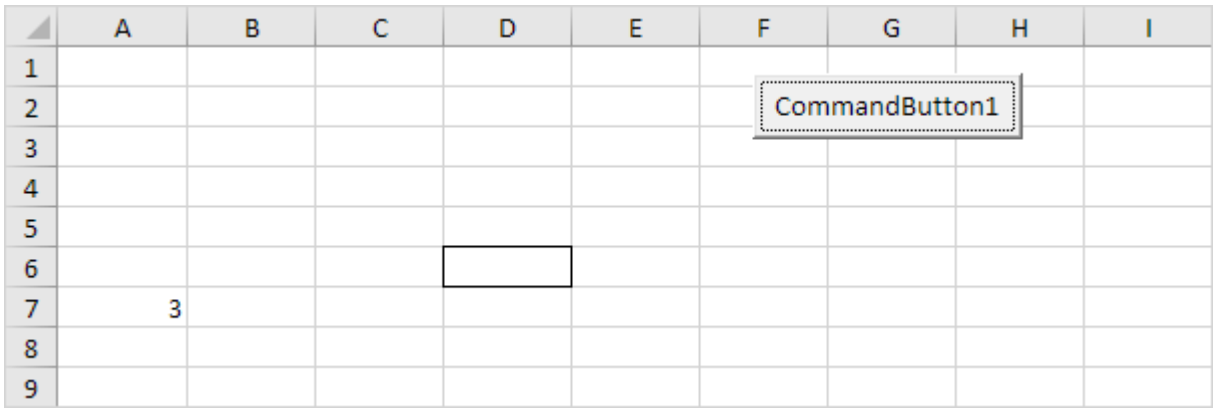
ActiveCell.EntireColumn.Cells(1).Value = 2




Note: borders in the image for illustration only.

9. Select cell D6. The following code line enters the value 3 into the first cell of the row below the row that contains the active cell.

```
ActiveCell.EntireRow.Offset(1, 0).Cells(1).Value = 3
```




Note: borders in the image for illustration only.

Join over 1 million Excel learners by following Excel Easy on Facebook, X, and LinkedIn.

5/14 Completed! Learn more about the range object >

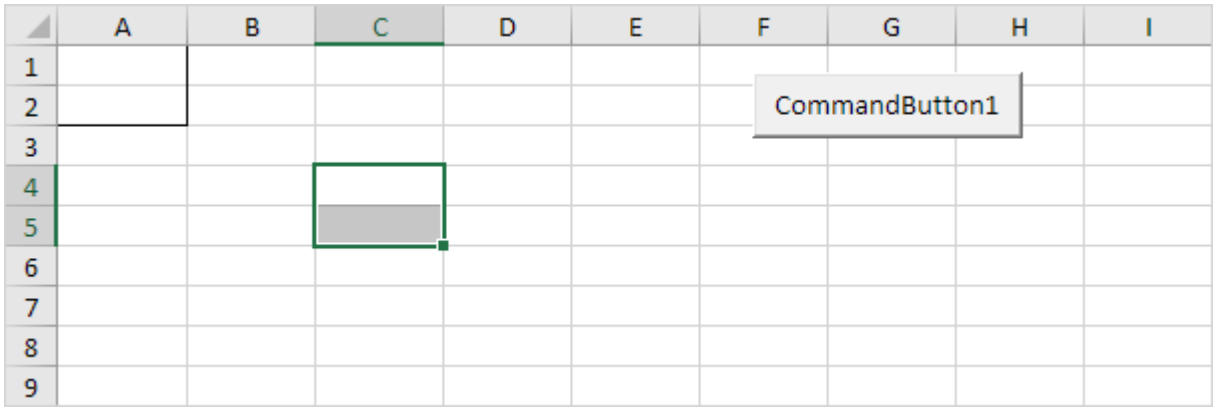
# Offset

The Offset property in Excel VBA takes the range which is a particular number of rows and columns away from a certain range.

Place a command button on your worksheet and add the following code lines:

1. The Offset property below returns a range which is 3 rows below and 2 columns to the right of a range.

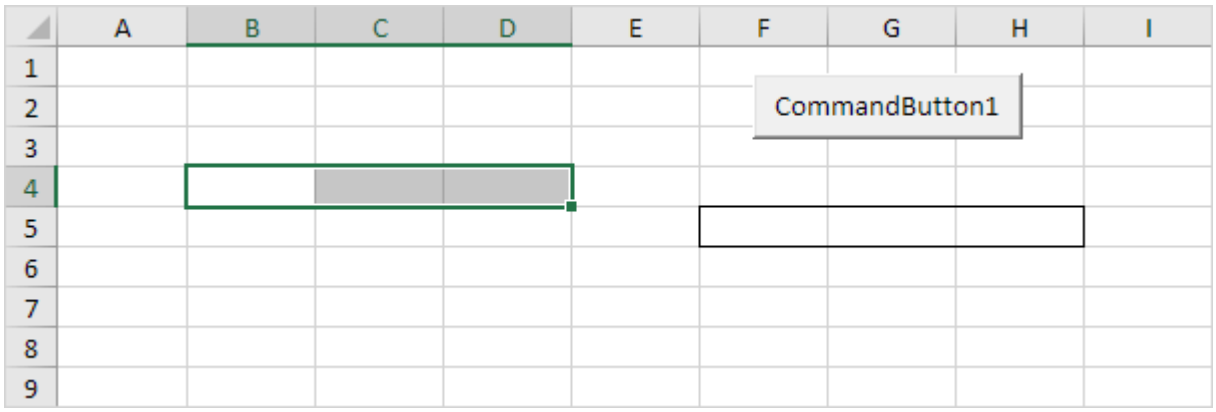
```
Range("A1:A2").Offset(3, 2).Select
```




Explanation: the Select method selects this range. The Offset property always takes the top left cell of a range as the starting point. Borders for illustration only.

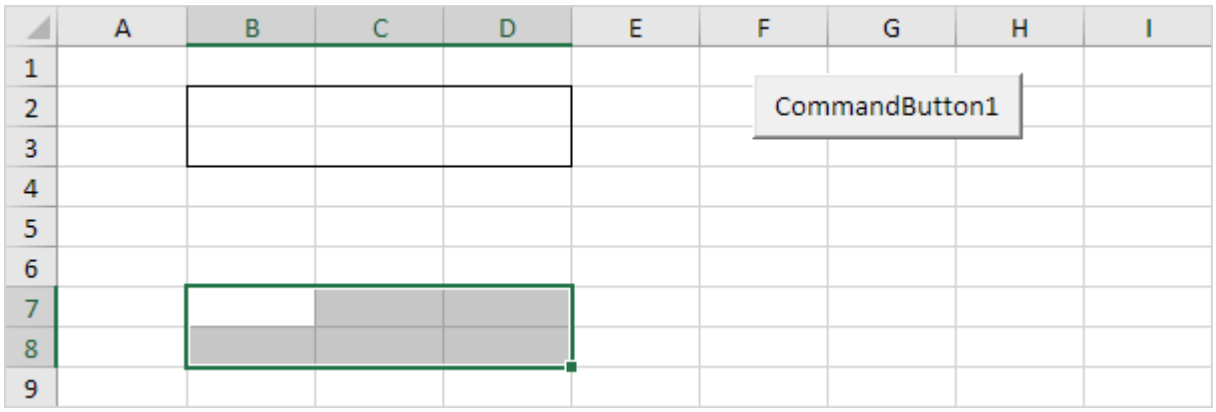
2. The Offset property below returns a range which is 1 row above and 4 columns to the left of a range.

```
Range("F5:H5").Offset(-1, -4).Select
```



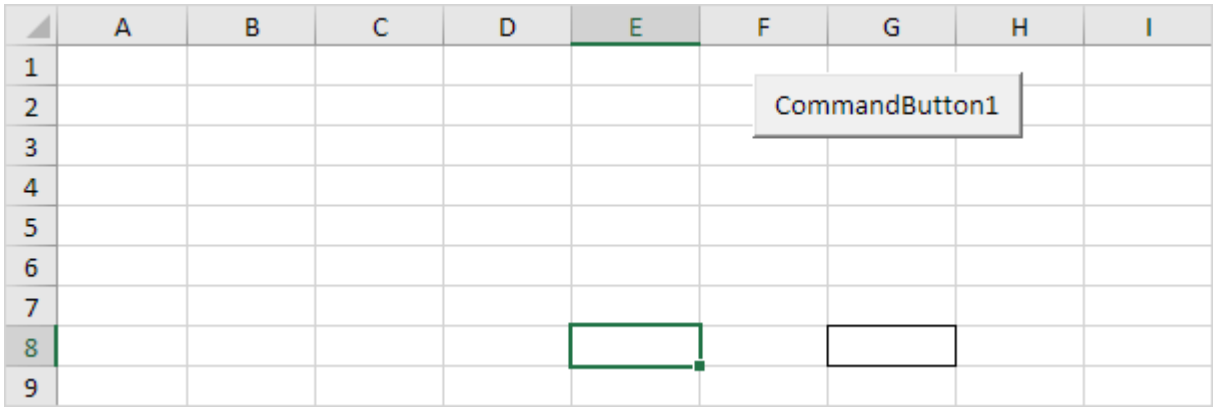

3. You can also offset by a number of rows only. For example, return a range which is 5 rows below a range.

```
Range("B2:D3").Offset(5).Select
```




4. Finally, you can also offset by a number of columns only. For example, return a cell which is 2 columns to the left of a cell.

```
Range("G8").Offset(0, -2).Select
```




For a practical example of the offset property, see our example program Create a Pattern.

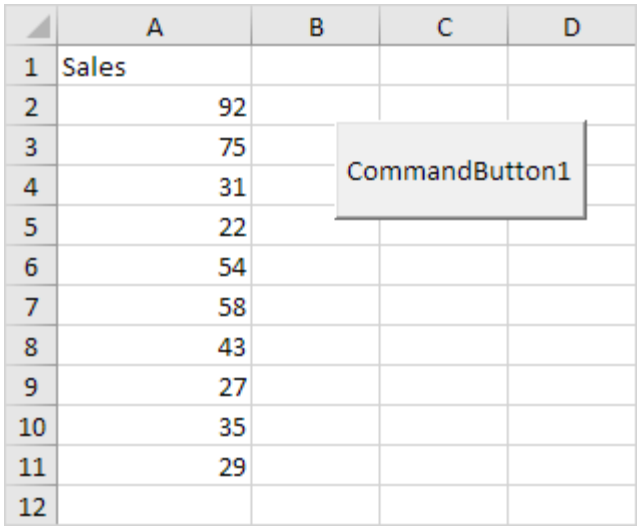
6/14 Completed! Learn more about the range object >

# From Active Cell to Last Entry

This example illustrates the End property of the Range object in Excel VBA. We will use this property to select the range from the Active Cell to the last entry in a column.

Situation:

Some sales figures in column A. Assume that you will be adding more sales figures over time.




Place a command button on your worksheet and add the following code lines:

1. To select the last entry in a column, simply add the following code line:

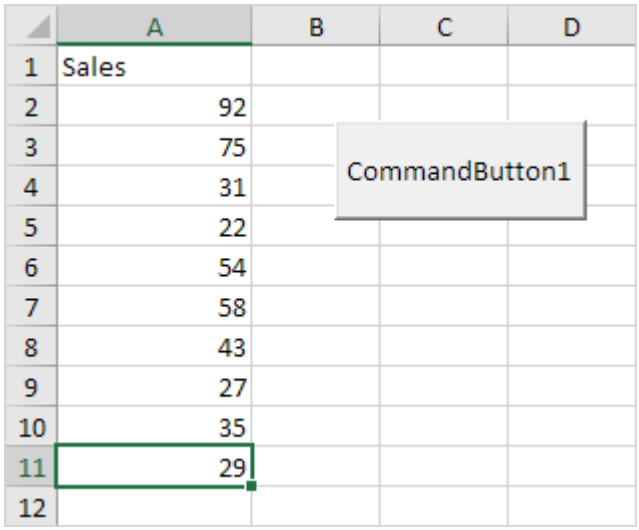
```
Range("A5").End(xlDown).Select
```

Note: instead of Range("A5"), you can also use Range("A1"), Range("A2"), etc. This code line is equivalent to pressing the

# END+DOWN ARROW : går til nederste felt i aktive region

# END+Up ARROW : går til øverste felt i aktive region

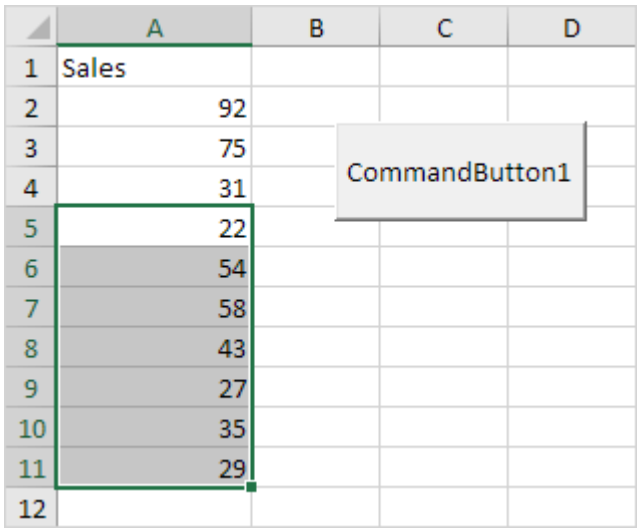
Result when you click the command button on the sheet:




2. To select the range from cell A5 to the last entry in the column, add the following code line:

Range(Range("A5"), Range("A5").End(xlDown)).Select

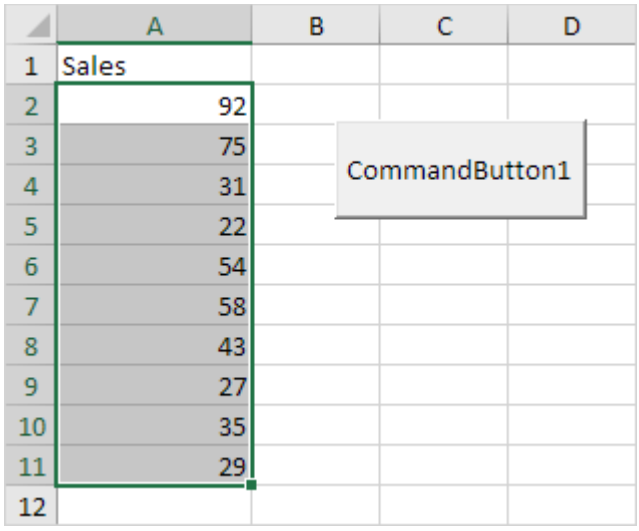
Result when you click the command button on the sheet:




3. To select the range from the Active Cell to the last entry in the column, simply replace Range("A5") with ActiveCell.

Range(ActiveCell, ActiveCell.End(xlDown)).Select

Result when you select cell A2 and click the command button on the sheet:




Note: you can use the constants xlUp, xlToRight and xlToLeft to move in the other directions. This way you can select a range from the Active Cell to the last entry in a row.

Join over 1 million Excel learners by following Excel Easy on Facebook, X, and LinkedIn.

7/14 Completed! Learn more about the range object

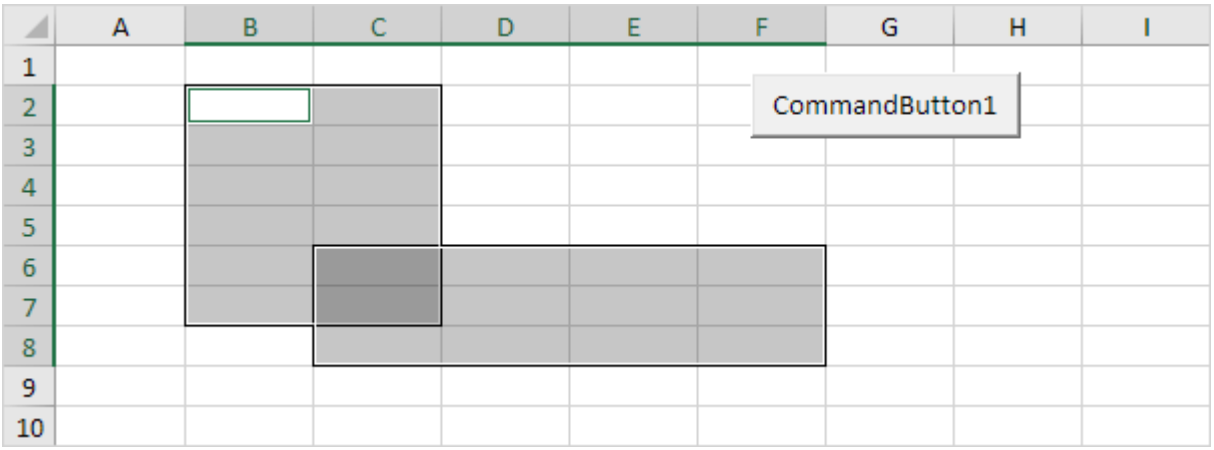
# Union and Intersect

The Union method in Excel VBA returns a Range object that represents the union of two or more ranges (borders below for illustration only).

Code line:

Union(Range("B2:C7"), Range("C6:F8")).Select

Result:




Note: the Union method doesn't return the mathematical union (cell C6 and cell C7 are included twice).

The Intersect method in Excel VBA returns a Range object that represents the intersection of two or more ranges (borders below for illustration only).

Code line:

```
Intersect(Range("B2:C7"), Range("C6:F8")).Select
```

# Font

The Font property of the Range object in Excel VBA gives access to a lot of other properties. That is because the Font property returns an object itself; the Font object. The Font object has many properties like the Color property and the Bold property.

# Color property

To change the color of an Excel range, use the Font property of the Range object, and then the Color property of the Font object.

1. Add the following code line:

```
Range("A1").Font.Color = -16776961
```

Explanation: Where do we get this strange number from? Well, we started the Macro Recorder and changed the color of a cell to red. You can do this for every color!

2. The following code line gives the exact same result.

```
Range("A1").Font.Color = vbRed
```

Explanation: vbRed is a sort of built-in constant in Excel VBA. Place your cursor on vbRed in the Visual Basic Editor and click F1 to see which other constants you can use.

3. The following code line gives the exact same result.

```
Range("A1").Font.Color = RGB(255, 0, 0)
```

Explanation: RGB stands for Red, Green and Blue. These are the three primary colors. Each component can take on a value from 0 to 255. With this function you can make every color. RGB(255,0,0) gives the pure Red color.

# Bold property

The following code line bolds a range:

```
Range("A1").Font.Bold = True
```

To unbold a range, you can use the False keyword. The Font object has many more properties. If you want to program these sort of things, just use the Macro Recorder to see how to do it! Usually code created by the Macro Recorder is too long. For example, the Macro Recorder creates the following code when we bold Range("A1").

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

'
    Range("A1").Select
    Selection.Font.Bold = True
End Sub
```

We have just seen that these two code lines can be written as one code line.

10/14 Completed! Learn more about the range object >

# Background Colors

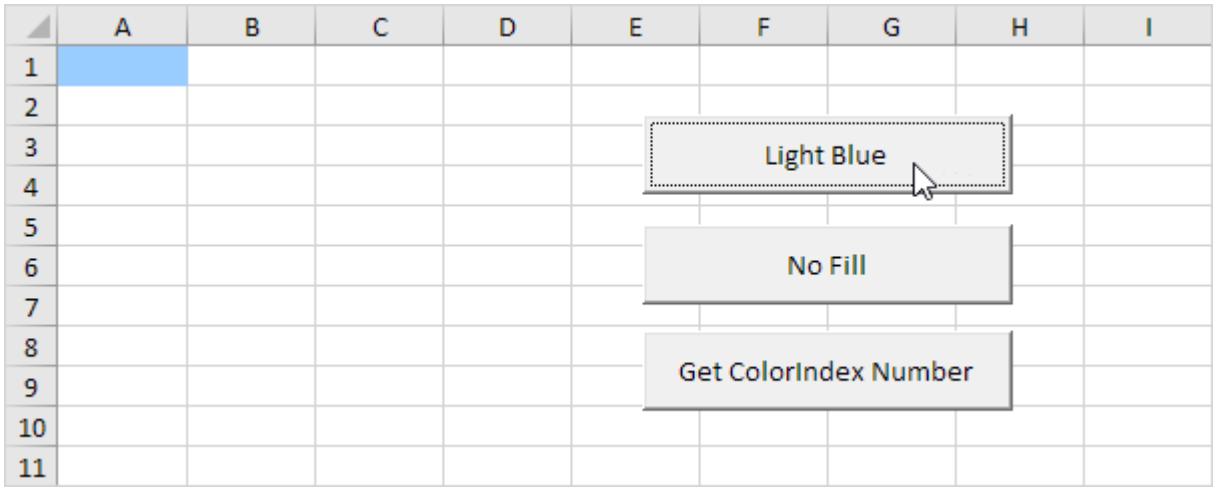
Changing background colors in Excel VBA is easy. Use the Interior property to return an Interior object. Then use the ColorIndex property of the Interior object to set the background color of a cell.

Place three command buttons on your worksheet and add the following code lines:

1. The code line below sets the background color of cell A1 to light blue.

```
Range("A1").Interior.ColorIndex = 37
```

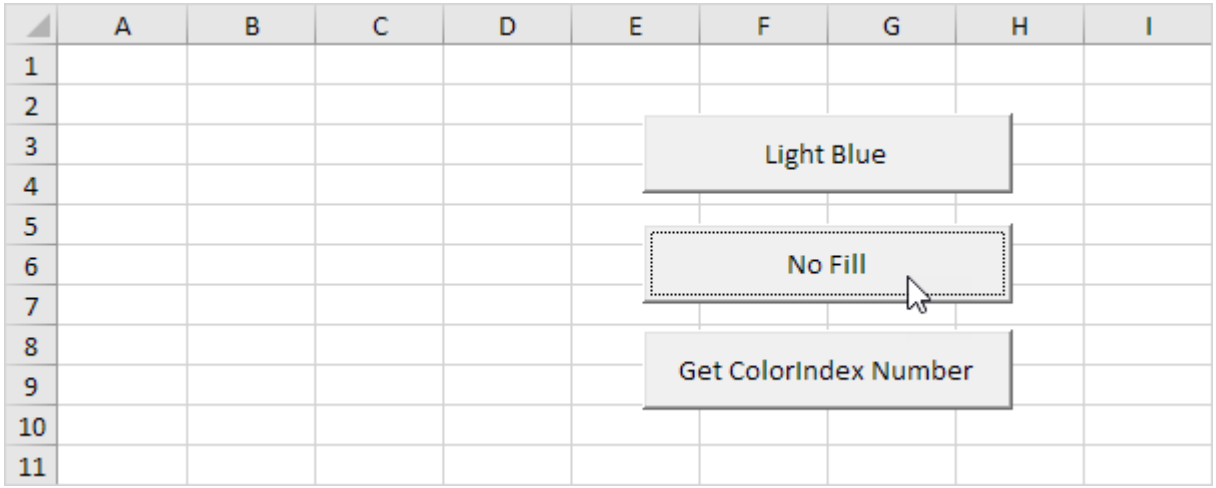
Result:




2. The following code line sets the background color of cell A1 to 'No Fill'.

```
Range("A1").Interior.ColorIndex = 0  'No Fill
```

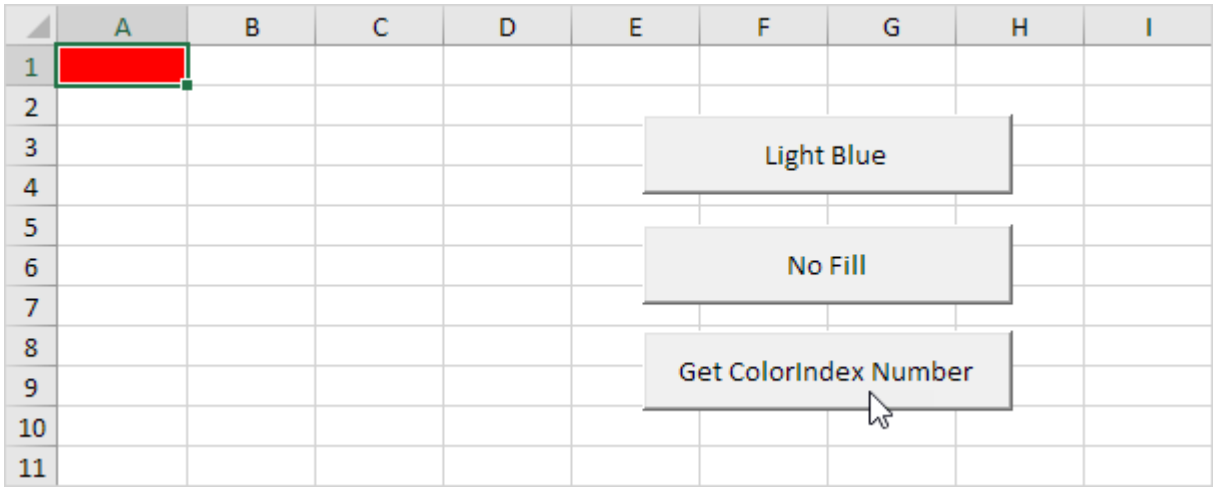
Result:



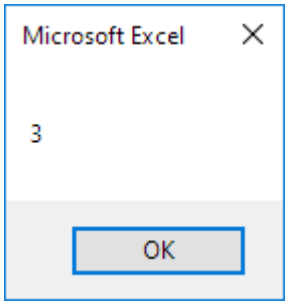

3. If you want to know the ColorIndex number of a color, simply ask Excel VBA.

```
MsgBox Selection.Interior.ColorIndex
```

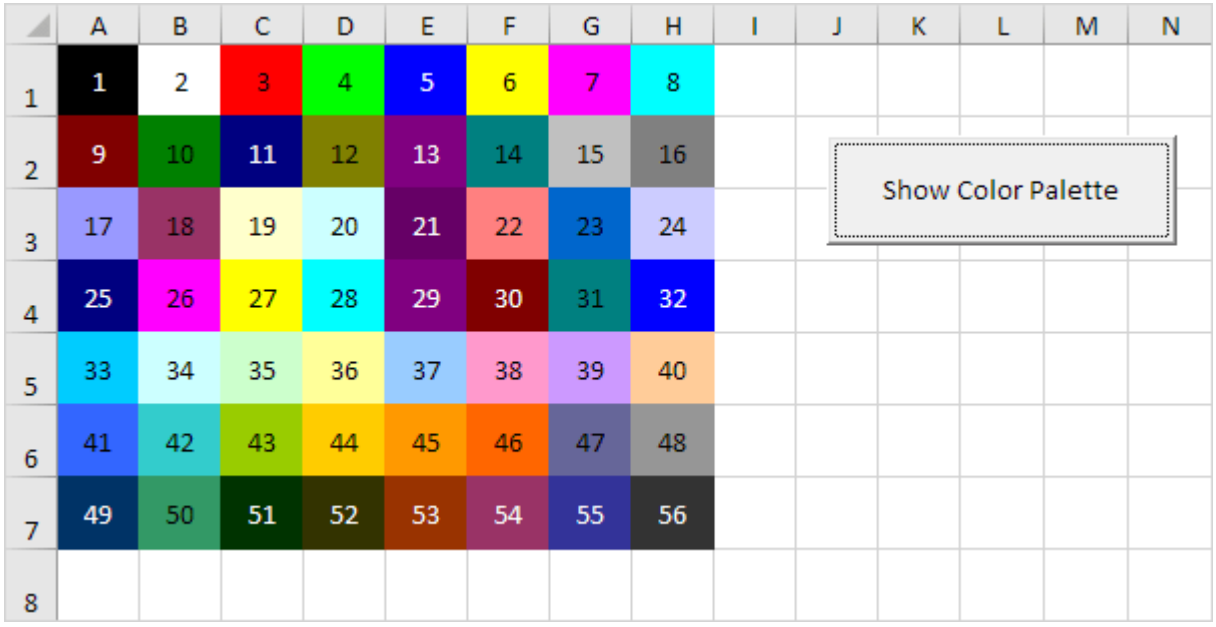
Select cell A1 and click the command button on the sheet:




Result:




The ColorIndex property gives access to a color palette of 56 colors.




Note: download the Excel file to see how we created this color palette.

5. If you can't find the specific color you are looking for, use the Color property and the RGB function.

```
Range("A1").Interior.Color = RGB(255, 0, 0)  'explicit til rød
```

Explanation: RGB stands for Red, Green and Blue. These are the three primary colors. Each component can take on a value from 0 to 255. With this function you can make every color. RGB(255,0,0) gives the pure Red color (ColorIndex = 3 produces the exact same result).



11/14 Completed! Learn more about the range object >

## Sort a Range

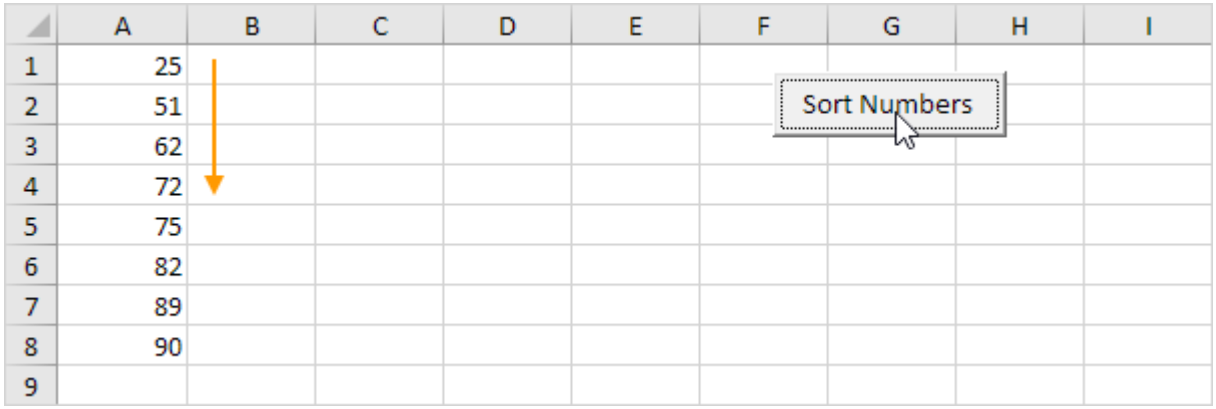
You can use the Sort method in Excel VBA to sort ranges in Excel. This article provides clear and practical examples to help you get started with sorting ranges using Excel VBA.

# Sort a Single Column

To sort a single column range, place a command button on your worksheet and add the following code line:

```
Range("A1:A8").Sort Key1:=Range("A1"), Order1:=xlAscending, Header:=xlNo
```

Result when you click the command button on the sheet:




Explanation: Key1:=Range("A1") specifies the primary key to sort by, which is the first cell in the range. Order1:=xlAscending sorts the data in ascending order. Header:=xlNo indicates that the range does not have a header row.

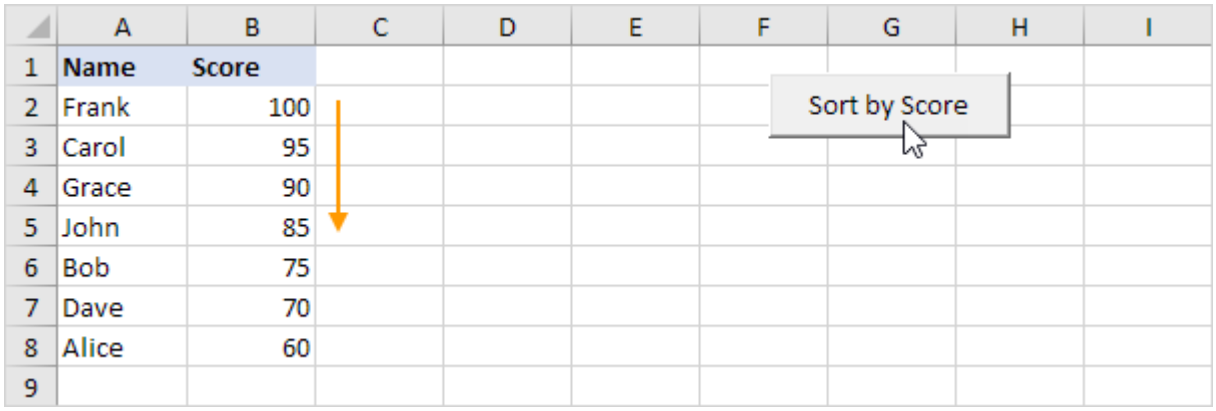
Sort a Range with Two Columns

Now, let's use the Sort method in Excel VBA to sort a range with two columns (see picture below). The Sort method works on the range A1:B8 this time.

To sort by the Score column in descending order, use the following code line:

```
Range("A1:B8").Sort Key1:=Range("B2"), Order1:=xlDescending, Header:=xlYes
```

Result when you click the command button on the sheet:




| | A | B |
|---|---|---|
| 1 | Name | Score |
| 2 | Frank | 100 |
| 3 | Carol | 95 |
| 4 | Grace | 90 |
| 5 | John | 85 |
| 6 | Bob | 75 |
| 7 | Dave | 70 |
| 8 | Alice | 60 |
| 9 | | |

Explanation: Key1:=Range("B2") specifies the primary key to sort by, which is the first data cell in the Score column. We use Range("B2") because our data has headers, and the actual sorting starts from the second row. Order1:=xlDescending sorts the data in descending order. Header:=xlYes indicates that the range has a header row.

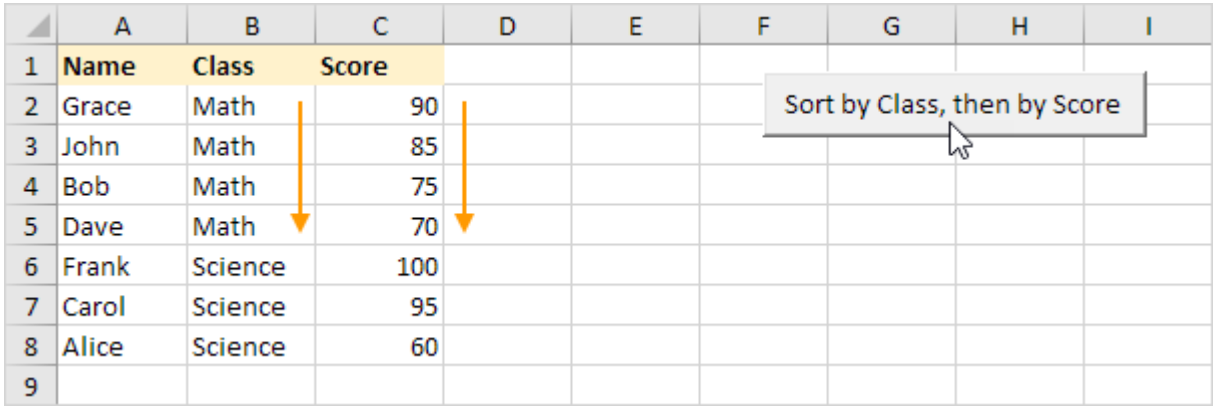
# Sort by Multiple Columns

Now, let's use the Sort method in Excel VBA to sort by multiple columns (see picture below). The Sort method works on the range A1:C8 this time.

To sort by Class in ascending order and then by Score in descending order, use the following code line:

Range("A1:C8").Sort Key1:=Range("B2"), Order1:=xlAscending, Key2:=Range("C2"), Order2:=xlDescending, Header:=xlYes

Result when you click the command button on the sheet:




| | A | B | C |
|---|---|---|---|
| 1 | Name | Class | Score |
| 2 | Grace | Math | 90 |
| 3 | John | Math | 85 |
| 4 | Bob | Math | 75 |
| 5 | Dave | Math | 70 |
| 6 | Frank | Science | 100 |
| 7 | Carol | Science | 95 |
| 8 | Alice | Science | 60 |
| 9 | | | |

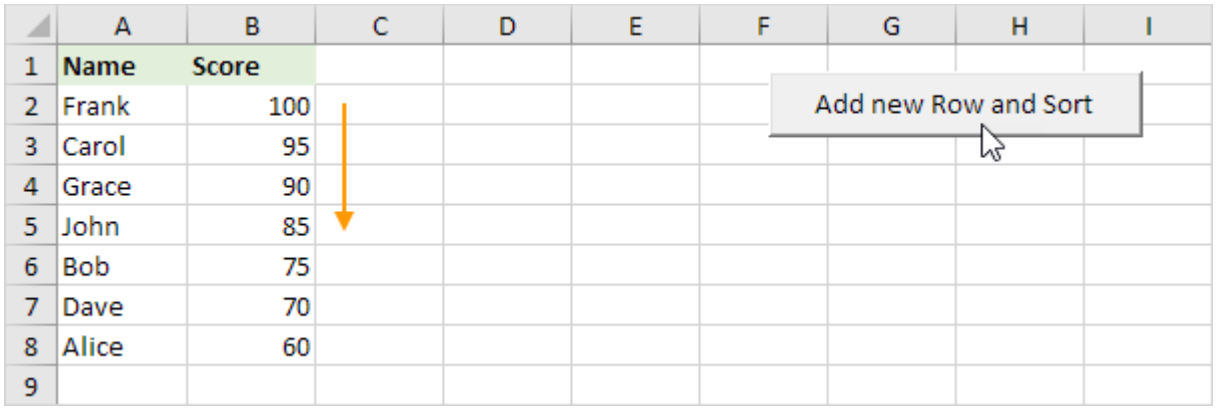
Explanation: this code uses two keys: the first key (Key1) sorts by Class in ascending order, and the second key (Key2) sorts by Score in descending order.

# Dynamic Sorting, hvor antal rækker kan ændres

For dynamic sorting, where the number of rows frequently changes, use the CurrentRegion property to include new data automatically.

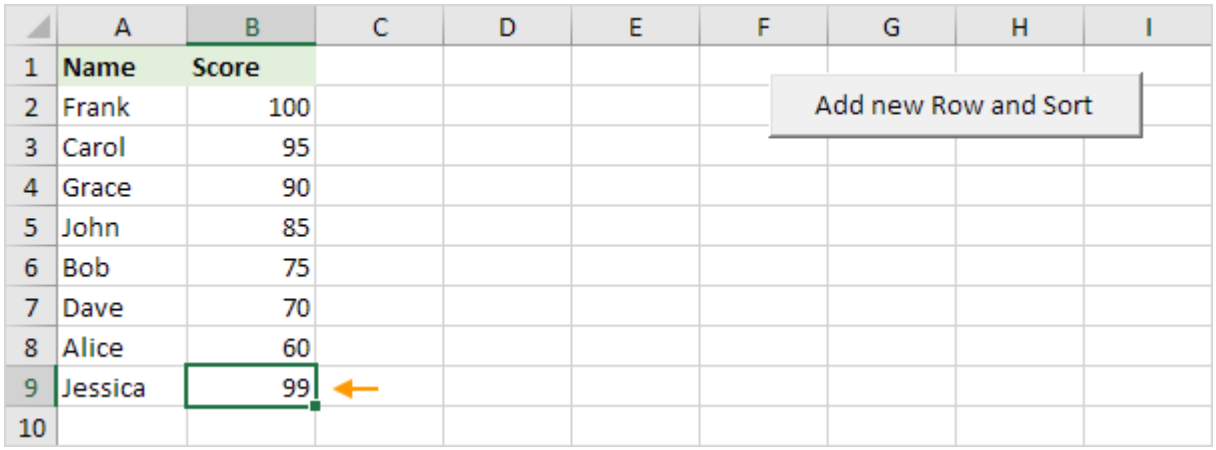
Range("A1").CurrentRegion.Sort Key1:=Range("B2"), Order1:=xlDescending, Header:=xlYes

1. Result when you click the command button on the sheet:



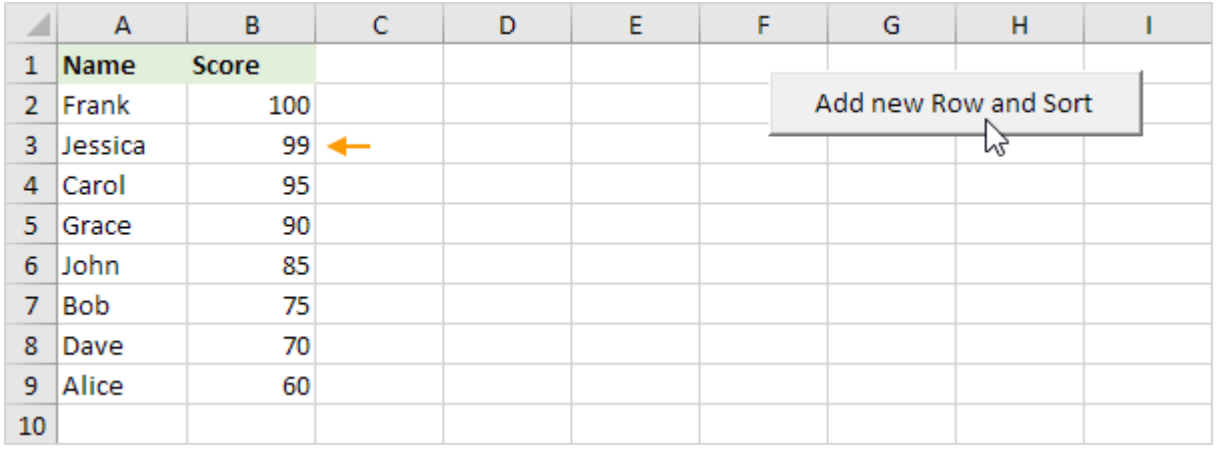

| | A | B |
|---|---|---|
| 1 | Name | Score |
| 2 | Frank | 100 |
| 3 | Carol | 95 |
| 4 | Grace | 90 |
| 5 | John | 85 |
| 6 | Bob | 75 |
| 7 | Dave | 70 |
| 8 | Alice | 60 |
| 9 | | |

2. Now, add a new row.




| | A | B |
|---|---|---|
| 1 | Name | Score |
| 2 | Frank | 100 |
| 3 | Carol | 95 |
| 4 | Grace | 90 |
| 5 | John | 85 |
| 6 | Bob | 75 |
| 7 | Dave | 70 |
| 8 | Alice | 60 |
| 9 | Jessica | 99 |
| 10 | | |

3. Sort again using the same VBA code line.




| | A | B |
|---|---|---|
| 1 | Name | Score |
| 2 | Frank | 100 |
| 3 | Jessica | 99 |
| 4 | Carol | 95 |
| 5 | Grace | 90 |
| 6 | John | 85 |
| 7 | Bob | 75 |
| 8 | Dave | 70 |
| 9 | Alice | 60 |
| 10 | | |

4. If your data contains empty rows, the CurrentRegion method may not work as expected. In such cases, use Cells.Sort to sort your data.

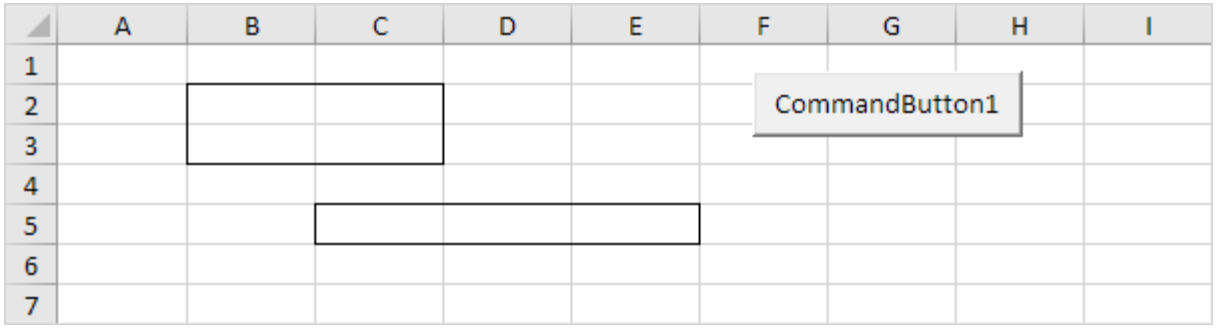
Cells.Sort Key1:=Columns("B"), Order1:=xlDescending, Header:=xlYes

Note: this code line sorts the entire sheet based on the specified column, even when there are empty rows. Ensure there is no data below the range to be sorted when using this code line!

12/14 Completed! Learn more about the range object >

## Areas Collection

This page illustrates the Areas collection in Excel VBA. In the example below, we have bordered Range("B2:C3,C5:E5"). This range has two areas. The comma separates the two areas.




Place a command button on your worksheet and add the following code lines:

1. First, we declare two Range objects. We call the Range objects rangeToUse and singleArea.

```
Dim rangeToUse As Range, singleArea As Range
```

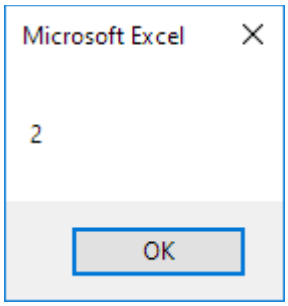
2. We initialize the Range object rangeToUse with Range("B2:C3,C5:E5")

```
Set rangeToUse = Range("B2:C3,C5:E5")
```

3. To count the number of areas of rangeToUse, add the following code line:

```
MsgBox rangeToUse.Areas.Count
```

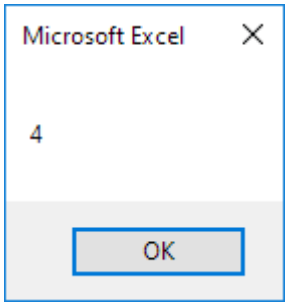
Result:




4. You can refer to the different areas of rangeToUse by using the index values. The following code line counts the numbers of cells of the first area.

```
MsgBox rangeToUse.Areas(1).Count
```

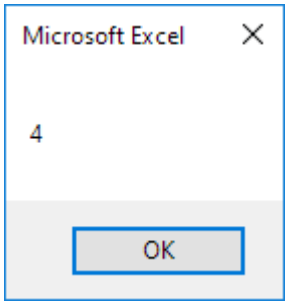
Result:

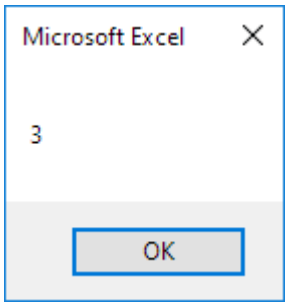



5. You can also loop through each area of rangeToUse and count the number of cells of each area. The macro below does the trick.

```
For Each singleArea In rangeToUse.Areas
    MsgBox singleArea.Count
Next singleArea
```

Result:




For a practical example of the areas collection, see our example program Compare Ranges.

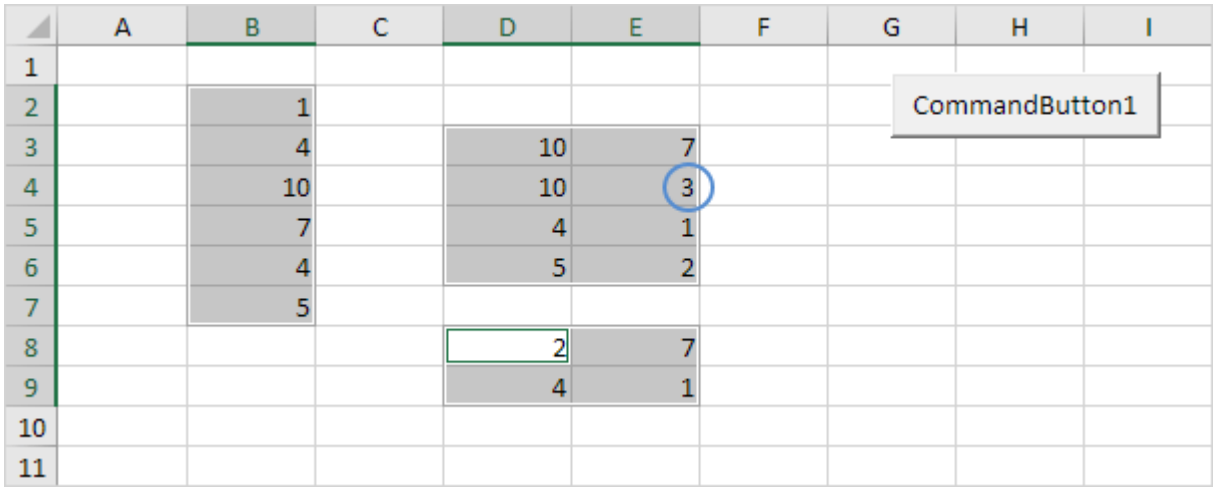
13/14 Completed! Learn more about the range object >
Go to Next Chapter: Variables

## Compare Ranges

Fil: compare-ranges

Below we will look at a program in Excel VBA that compares randomly selected ranges and highlights cells that are unique. If you are not familiar with areas yet, we highly recommend you to read this page first.

Situation:




Note: the only unique value in this example is the 3 since all other values occur in at least one more area.
To select Range("B2:B7,D3:E6,D8:E9"), hold down Ctrl and select each area.

Place a command button on your worksheet and add the following code lines:

1. First, we declare four Range objects and two variables of type Integer.

```
Dim rangeToUse As Range, singleArea As Range, cell1 As Range, cell2 As Range, i As Integer, j As Integer
```

2. We initialize the Range object rangeToUse with the selected range.

```
Set rangeToUse = Selection
```

3. Add the line which changes the background color of all cells to 'No Fill'. Also add the line which removes the borders of all cells.

```
Cells.Interior.ColorIndex = 0
Cells.Borders.LineStyle = xlNone
```

4. Inform the user when he or she only selects one area.

```
If Selection.Areas.Count <= 1 Then
    MsgBox "Please select more than one area."
Else


End If
```

The next code lines (at 5, 6 and 7) must be added between Else and End If.

5. Color the cells of the selected areas.

```
rangeToUse.Interior.ColorIndex = 38
```

6. Border each area.

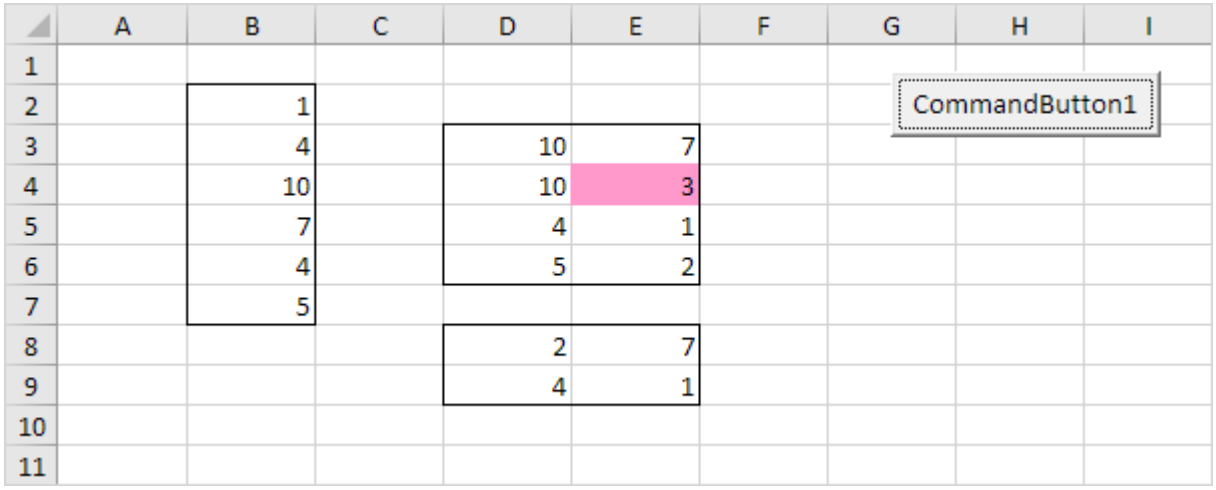
```
For Each singleArea In rangeToUse.Areas
  singleArea.BorderAround ColorIndex:=1, Weight:=xlThin
Next singleArea
```

7. The rest of this program looks as follows.

```
For i = 1 To rangeToUse.Areas.Count
   For j = i + 1 To rangeToUse.Areas.Count
      For Each cell1 In rangeToUse.Areas(i)
         For Each cell2 In rangeToUse.Areas(j)
            If cell1.Value = cell2.Value Then
               cell1.Interior.ColorIndex = 0
               cell2.Interior.ColorIndex = 0
            End If
         Next cell2
      Next cell1
   Next j
Next i
```

Explanation: this may look a bit overwhelming, but it's not that difficult. rangeToUse.Areas.Count equals 3, so the first two code lines reduce to For i = 1 to 3 and For j = i + 1 to 3. For i = 1, j = 2, Excel VBA compares all values of the first area with all values of the second area. For i = 1, j = 3, Excel VBA compares all values of the first area with all values of the third area. For i = 2, j = 3, Excel VBA compares all values of the second area with all values of the third area. If values are the same, it sets the background color of both cells to 'No Fill', because they are not unique.

Result when you click the command button on the sheet:




Join over 1 million Excel learners by following Excel Easy on Facebook, X, and LinkedIn.

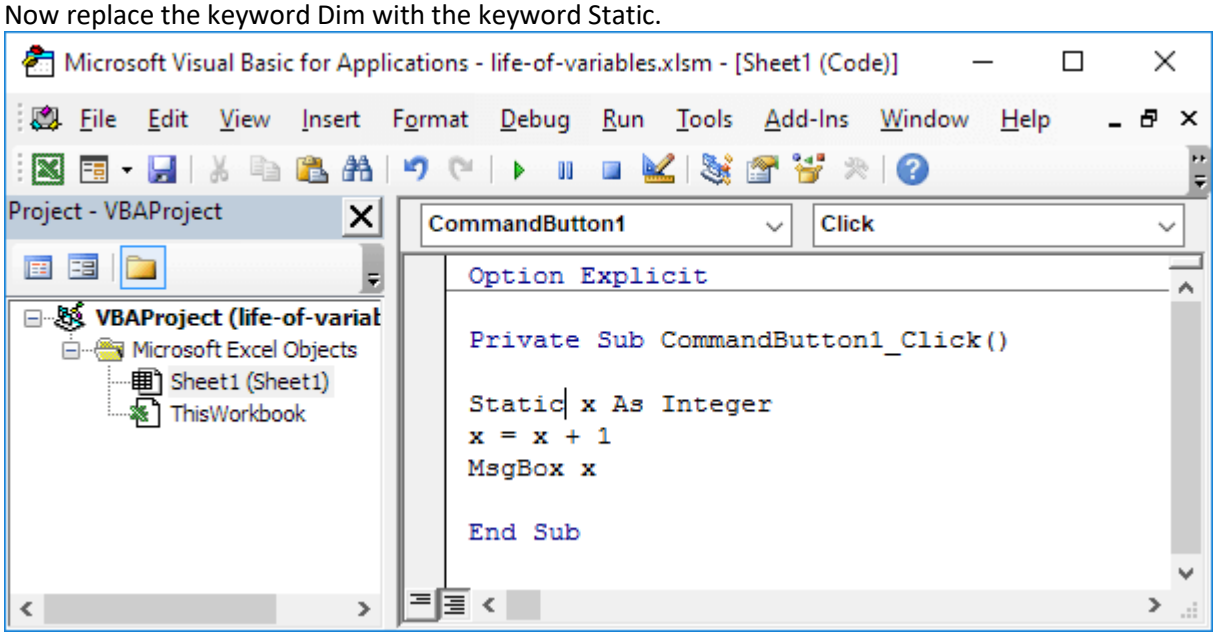
14/14 Completed! Learn more about the range object >
Go to Next Chapter: Variables

## **** Varible

## Option Explicit forces you to declare all your variables -> større sikkerhed

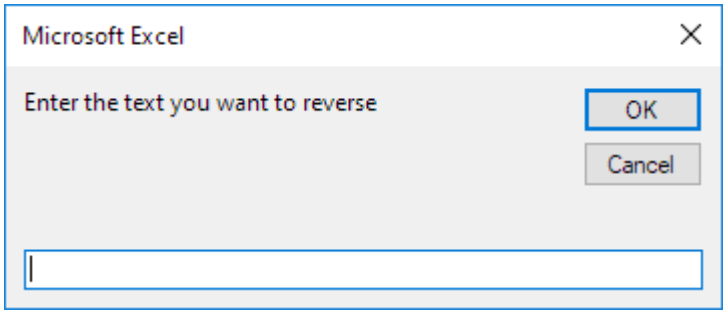
## Static variable beholder værdien efter procedurens afslutning

Explanation: Excel VBA destroys the variable when the procedure ends. Each time you click the command button on the sheet, Excel VBA creates the variable x again, adds the value 1 to it, and displays the result. Now replace the keyword Dim with the keyword Static.



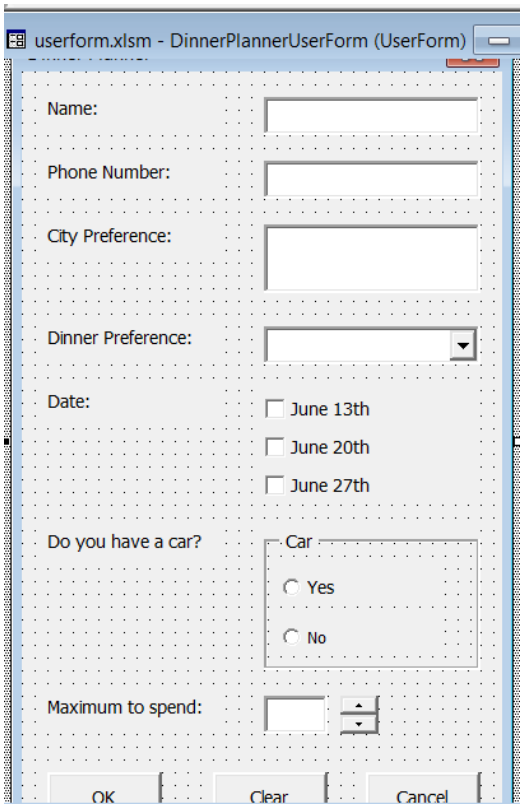

# Type Mismatch

```
Dim number As Double
number = InputBox("Enter a number", "Square Root")

If IsNumeric(number) Then
   MsgBox "The square root of " & number & " is " & Sqr(number)
Else
   MsgBox "Please enter a number"
End If

text = InputBox("Enter the text you want to reverse")
length = Len(text)
```




3. We start a For Next loop.
For i = 0 To length - 1
4. Now comes the simple trick. We take the last character from text and place it at the front of ReversedText. We can use the Mid function in Excel VBA to extract a character from a string. We use the & operator to concatenate (join) two strings.
reversedText = reversedText & Mid(text, (length - i), 1)
5. Don't forget to close the loop.
Next i

## Userform

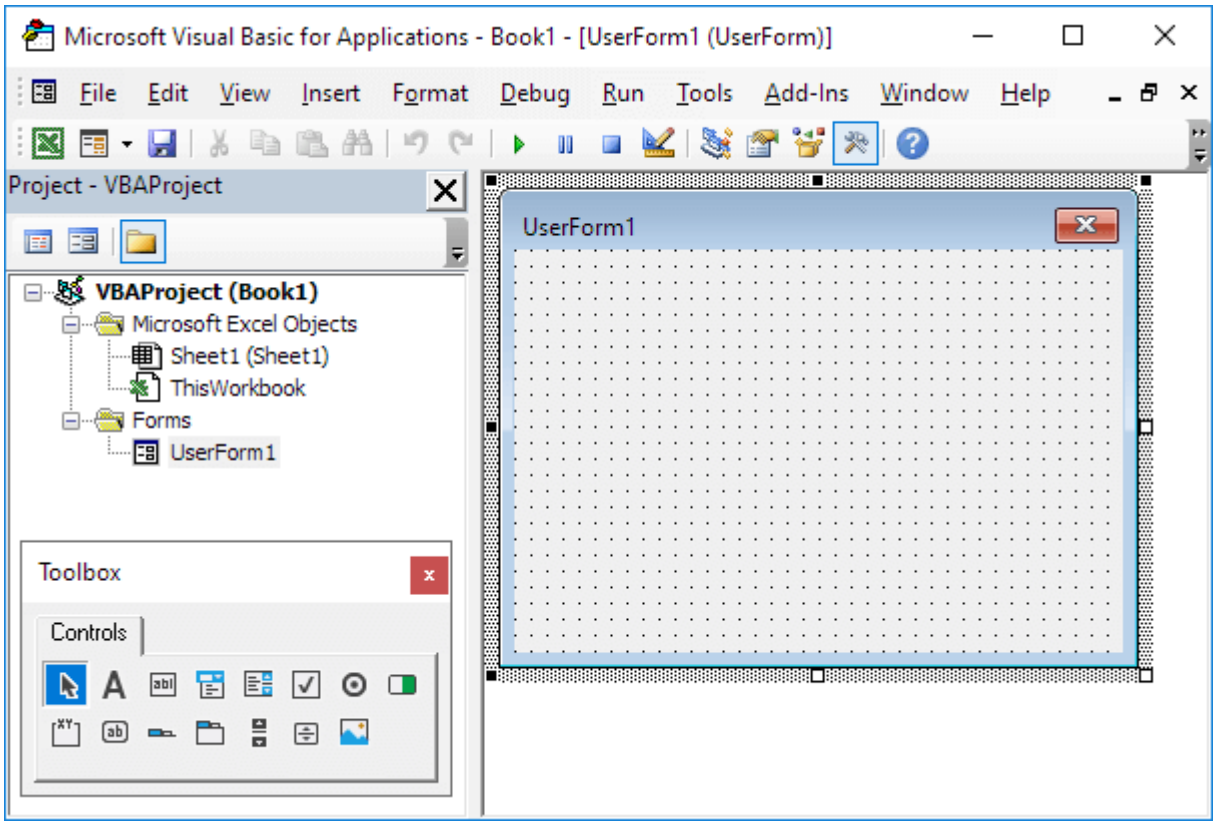



# Add the Controls

To add the controls to the Userform, execute the following steps.

1. Open the Visual Basic Editor. If the Project Explorer is not visible, click View, Project Explorer.

2. Click Insert, Userform. If the Toolbox does not appear automatically, click View, Toolbox. Your screen should be set up as below.




# Show the Userform

To show the Userform, place a command button on your worksheet and add the following code line:

```
Private Sub CommandButton1_Click()
    DinnerPlannerUserForm.Show
End Sub
```

We are now going to create the Sub UserForm_Initialize. When you use the Show method for the Userform, UserForm_Initialize.  will automatically be executed.

# koden

```
Option Explicit

Private Sub CancelButton_Click()
                Unload Me
End Sub
```

```
Private Sub ClearButton_Click()
        Call UserForm_Initialize
End Sub

Private Sub DinnerComboBox_Change()

End Sub

Private Sub MoneySpinButton_Change()
        MoneyTextBox.Text = MoneySpinButton.Value
End Sub


Private Sub OKButton_Click()
        Dim emptyRow As Long

        'Make Sheet1 active
        Sheet1.Activate

        'Determine emptyRow
        emptyRow = WorksheetFunction.CountA(Range("A:A")) + 1

        'Transfer information
        Cells(emptyRow, 1).Value = NameTextBox.Value
        Cells(emptyRow, 2).Value = PhoneTextBox.Value
        Cells(emptyRow, 3).Value = CityListBox.Value
        Cells(emptyRow, 4).Value = DinnerComboBox.Value

        If DateCheckBox1.Value = True Then Cells(emptyRow, 5).Value = DateCheckBox1.Caption

        If DateCheckBox2.Value = True Then Cells(emptyRow, 5).Value = Cells(emptyRow, 5).Value &
" " & DateCheckBox2.Caption

        If DateCheckBox3.Value = True Then Cells(emptyRow, 5).Value = Cells(emptyRow, 5).Value &
" " & DateCheckBox3.Caption

        If CarOptionButton1.Value = True Then
                Cells(emptyRow, 6).Value = "Yes"
        Else
                Cells(emptyRow, 6).Value = "No"
        End If

        Cells(emptyRow, 7).Value = MoneyTextBox.Value

End Sub


Private Sub UserForm_Initialize()

        'Empty NameTextBox
```

```
        NameTextBox.Value = ""

        'Empty PhoneTextBox
        PhoneTextBox.Value = ""

        'Empty CityListBox
        CityListBox.Clear

        'Fill CityListBox
        With CityListBox
                .AddItem "San Francisco"
                .AddItem "Oakland"
                .AddItem "Richmond"
        End With

        'Empty DinnerComboBox
        DinnerComboBox.Clear

        'Fill DinnerComboBox
        With DinnerComboBox
                .AddItem "Italian"
                .AddItem "Chinese"
                .AddItem "Frites and Meat"
        End With

        'Uncheck DataCheckBoxes
        DateCheckBox1.Value = False
        DateCheckBox2.Value = False
        DateCheckBox3.Value = False

        'Set no car as default
        CarOptionButton2.Value = True

        'Empty MoneyTextBox
        MoneyTextBox.Value = ""

        'Set Focus on NameTextBox
        NameTextBox.SetFocus

End Sub
```

Koden-slut

# activex-controls, fx Button, TextBox, Lister

https://www.excel-easy.com/vba/activex-controls.html

## *** 300 Examples

1 Create a Macro: With Excel VBA you can automate tasks in Excel by writing so called macros. In this chapter, learn how to create a simple macro.
2 MsgBox: The MsgBox is a dialog box in Excel VBA you can use to inform the users of your program.
3 Workbook and Worksheet Object: Learn more about the Workbook and Worksheet object in Excel VBA.
4 Range Object: The Range object, which is the representation of a cell (or cells) on your worksheet, is the most important object of Excel VBA.
5 Variables: This chapter teaches you how to declare, initialize and display a variable in Excel VBA.
6 If Then Statement: Use the If Then statement in Excel VBA to execute code lines if a specific condition is met.
7 Loop: Looping is one of the most powerful programming techniques. A loop in Excel VBA enables you to loop through a range of cells with just a few codes lines.
8 Macro Errors: This chapter teaches you how to deal with macro errors in Excel.
9 String Manipulation: In this chapter, you'll find the most important functions to manipulate strings in Excel VBA.
10 Date and Time: Learn how to work with dates and times in Excel VBA.
11 Events: Events are actions performed by users which trigger Excel VBA to execute code.
12 Array: An array is a group of variables. In Excel VBA, you can refer to a specific variable (element) of an array by using the array name and the index number.
13 Function and Sub: In Excel VBA, a function can return a value while a sub cannot.
14 Application Object: The mother of all objects is Excel itself. We call it the Application object. The application object gives access to a lot of Excel related options.
15 ActiveX Controls: Learn how to create ActiveX controls such as command buttons, text boxes, list boxes etc.
16 Userform: This chapter teaches you how to create an Excel VBA Userform.

You can find related examples and features on the right side of each chapter. Below you can find 80 popular examples.
1 Find Duplicates: This page teaches you how to find duplicate values (or triplicates) and how to find duplicate rows in Excel.

# *Find Duplicates

Duplicate Values | Triplicates | Duplicate Rows
This page teaches you how to **find duplicate values** (or triplicates) and how to find duplicate rows in **Excel**.

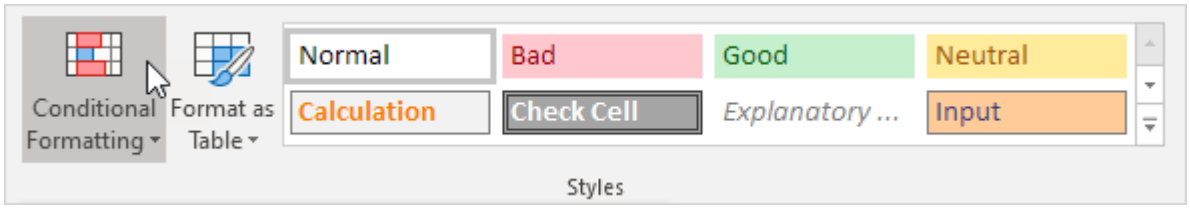
## Duplicate Values

To find and highlight duplicate values in Excel, execute the following steps.

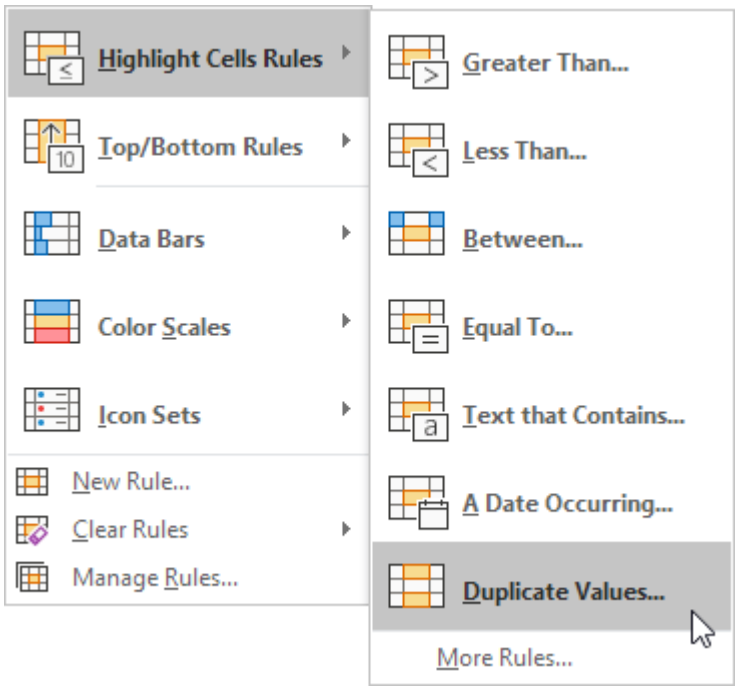
1. Select the range A1:C10.

| | A | B | C | D |
|---|---|---|---|---|
| 1 | Sierra | Tango | Charlie | |
| 2 | Kilo | Bravo | Yankee | |
| 3 | Golf | Mike | Delta | |
| 4 | Juliet | Alpha | Foxtrot | |
| 5 | Papa | X-ray | November | |
| 6 | Zulu | Sierra | Whiskey | |
| 7 | Romeo | Echo | Quebec | |
| 8 | India | Oscar | Delta | |
| 9 | Sierra | Lima | Uniform | |
| 10 | Hotel | Juliet | Victor | |
| 11 | | | | |

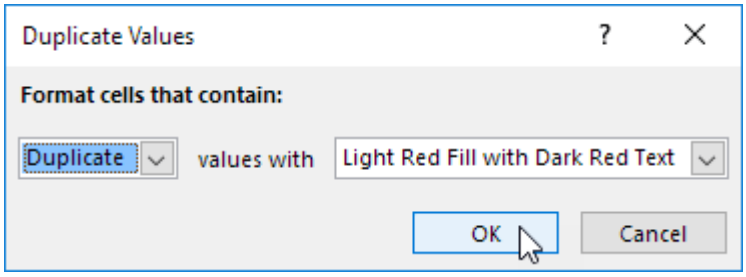
2. On the Home tab, in the Styles group, click Conditional Formatting.




3. Click Highlight Cells Rules, Duplicate Values.




4. Select a formatting style and click OK.




Result. Excel highlights the duplicate names.

| | A | B | C | D |
|---|---|---|---|---|
| 1 | Sierra | Tango | Charlie | |
| 2 | Kilo | Bravo | Yankee | |
| 3 | Golf | Mike | Delta | |
| 4 | Juliet | Alpha | Foxtrot | |
| 5 | Papa | X-ray | November | |
| 6 | Zulu | Sierra | Whiskey | |
| 7 | Romeo | Echo | Quebec | |
| 8 | India | Oscar | Delta | |
| 9 | Sierra | Lima | Uniform | |
| 10 | Hotel | Juliet | Victor | |
| 11 | | | | |

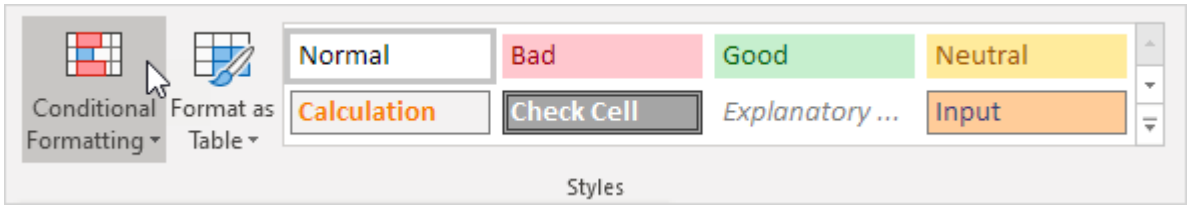
Note: select Unique from the first drop-down list to highlight the unique names.

# Triplicates

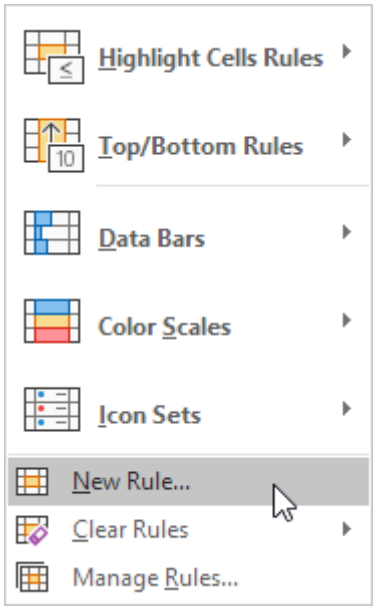
By default, Excel highlights duplicates (Juliet, Delta), triplicates (Sierra), etc. (see previous image). Execute the following steps to highlight triplicates only.

1. First, clear the previous conditional formatting rule.
2. Select the range A1:C10.

3. On the Home tab, in the Styles group, click Conditional Formatting.



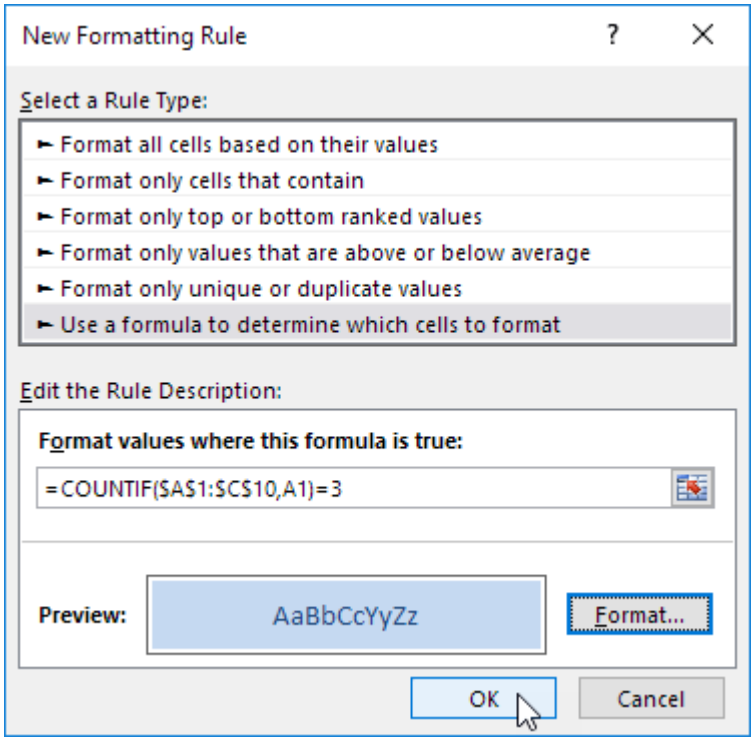

4. Click New Rule.




5. Select 'Use a formula to determine which cells to format'.

6. Enter the formula =COUNTIF($A$1:$C$10,A1)=3

7. Select a formatting style and click OK.




Result. Excel highlights the triplicate names.

| | A | B | C | D |
|---|---|---|---|---|
| 1 | Sierra | Tango | Charlie | |
| 2 | Kilo | Bravo | Yankee | |
| 3 | Golf | Mike | Delta | |
| 4 | Juliet | Alpha | Foxtrot | |
| 5 | Papa | X-ray | November | |
| 6 | Zulu | Sierra | Whiskey | |
| 7 | Romeo | Echo | Quebec | |
| 8 | India | Oscar | Delta | |
| 9 | Sierra | Lima | Uniform | |
| 10 | Hotel | Juliet | Victor | |
| 11 | | | | |

Explanation: =COUNTIF($A$1:$C$10,A1) counts the number of names in the range A1:C10 that are equal to the name in cell A1. If COUNTIF($A$1:$C$10,A1) = 3, Excel formats cell A1. Always write the formula for the upper-left cell in the selected range (A1:C10). Excel automatically copies the formula to the other cells. Thus, cell A2 contains the formula =COUNTIF($A$1:$C$10,A2)=3, cell A3 =COUNTIF($A$1:$C$10,A3)=3, etc. Notice how we created an absolute reference ($A$1:$C$10) to fix this reference.

Note: =COUNTIF($A$1:$C$10,A1)>3 highlights names that occur more than 3 times. You can use any formula you like.

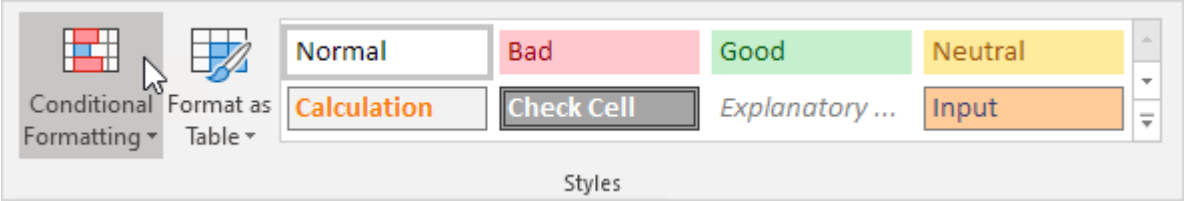
# Duplicate Rows

To find and highlight duplicate rows in Excel, use COUNTIFS (with the letter S at the end) instead of COUNTIF.

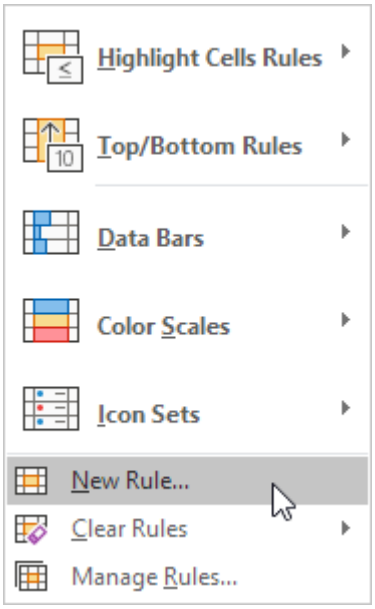
1. Select the range A1:C10.

| | A | B | C | D |
|---|---|---|---|---|
| 1 | Leopard | Africa | Zambia | |
| 2 | Lion | Africa | South Africa | |
| 3 | Elephant | Asia | Thailand | |
| 4 | Leopard | Asia | India | |
| 5 | Rhino | Africa | South Africa | |
| 6 | Buffalo | Asia | Cambodia | |
| 7 | Lion | Africa | South Africa | |
| 8 | Rhino | Asia | Nepal | |
| 9 | Buffalo | Africa | South Africa | |
| 10 | Elephant | Africa | Botswana | |
| 11 | | | | |

2. On the Home tab, in the Styles group, click Conditional Formatting.



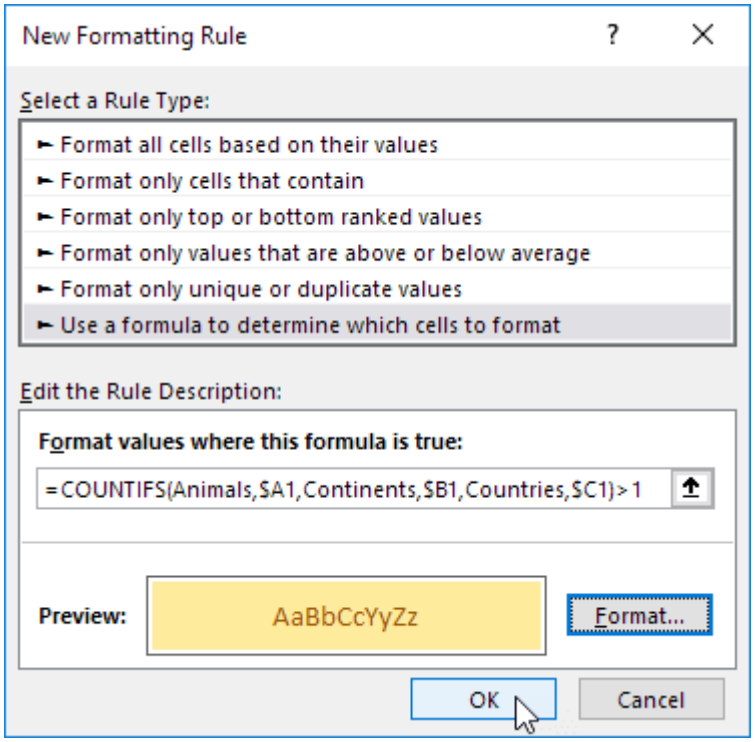

3. Click New Rule.




4. Select 'Use a formula to determine which cells to format'.

5. Enter the formula =COUNTIFS(Animals,$A1,Continents,$B1,Countries,$C1)>1

6. Select a formatting style and click OK.




Note: the named range Animals refers to the range A1:A10, the named range Continents refers to the range B1:B10 and the named range Countries refers to the range C1:C10. =COUNTIFS(Animals,$A1,Continents,$B1,Countries,$C1) counts the number of rows based on multiple criteria (Leopard, Africa, Zambia).

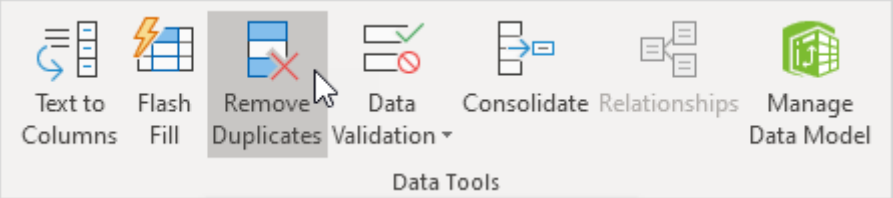
Result. Excel highlights the duplicate rows.

| | A | B | C | D |
|---|---|---|---|---|
| 1 | Leopard | Africa | Zambia | |
| 2 | Lion | Africa | South Africa | |
| 3 | Elephant | Asia | Thailand | |
| 4 | Leopard | Asia | India | |
| 5 | Rhino | Africa | South Africa | |
| 6 | Buffalo | Asia | Cambodia | |
| 7 | Lion | Africa | South Africa | |
| 8 | Rhino | Asia | Nepal | |
| 9 | Buffalo | Africa | South Africa | |
| 10 | Elephant | Africa | Botswana | |
| 11 | | | | |

Explanation: if COUNTIFS(Animals,$A1,Continents,$B1,Countries,$C1) > 1, in other words, if there are multiple (Leopard, Africa, Zambia) rows, Excel formats cell A1. Always write the formula for the upper-left cell in the selected range (A1:C10). Excel automatically copies the formula to the other cells. We fixed the reference to each column by placing a $ symbol in front of the column letter ($A1, $B1 and $C1). As a result, cell A1, B1 and C1 contain the same formula, cell A2, B2 and C2 contain the following formula:

=COUNTIFS(Animals,$A2,Continents,$B2,Countries,$C2)>1, etc.

7. Finally, you can use the Remove Duplicates tool in Excel to quickly remove duplicate rows. On the Data tab, in the Data Tools group, click Remove Duplicates.




In the example below, Excel removes all identical rows (blue) except for the first identical row found (yellow).

| | A | B | C | D |
|---|---|---|---|---|
| 1 | **Last Name** | **Sales** | **Country** | **Quarter** |
| 2 | Smith | $16,753.00 | UK | Qtr 3 |
| 3 | Johnson | $14,808.00 | USA | Qtr 4 |
| 4 | Williams | $10,644.00 | UK | Qtr 2 |
| 5 | Jones | $1,390.00 | USA | Qtr 3 |
| 6 | Brown | $4,865.00 | USA | Qtr 4 |
| 7 | Smith | $16,753.00 | UK | Qtr 3 |
| 8 | Williams | $12,438.00 | UK | Qtr 1 |
| 9 | Johnson | $9,339.00 | UK | Qtr 2 |
| 10 | Smith | $18,919.00 | USA | Qtr 3 |
| 11 | Jones | $9,213.00 | USA | Qtr 4 |
| 12 | Jones | $7,433.00 | UK | Qtr 1 |
| 13 | Smith | $16,753.00 | UK | Qtr 3 |
| 14 | Brown | $3,255.00 | USA | Qtr 2 |
| 15 | Williams | $14,867.00 | USA | Qtr 3 |
| 16 | Williams | $19,302.00 | UK | Qtr 4 |
| 17 | Smith | $9,698.00 | USA | Qtr 1 |
| 18 | | | | |

| | A | B | C | D |
|---|---|---|---|---|
| 1 | **Last Name** | **Sales** | **Country** | **Quarter** |
| 2 | Smith | $16,753.00 | UK | Qtr 3 |
| 3 | Johnson | $14,808.00 | USA | Qtr 4 |
| 4 | Williams | $10,644.00 | UK | Qtr 2 |
| 5 | Jones | $1,390.00 | USA | Qtr 3 |
| 6 | Brown | $4,865.00 | USA | Qtr 4 |
| 7 | Williams | $12,438.00 | UK | Qtr 1 |
| 8 | Johnson | $9,339.00 | UK | Qtr 2 |
| 9 | Smith | $18,919.00 | USA | Qtr 3 |
| 10 | Jones | $9,213.00 | USA | Qtr 4 |
| 11 | Jones | $7,433.00 | UK | Qtr 1 |
| 12 | Brown | $3,255.00 | USA | Qtr 2 |
| 13 | Williams | $14,867.00 | USA | Qtr 3 |
| 14 | Williams | $19,302.00 | UK | Qtr 4 |
| 15 | Smith | $9,698.00 | USA | Qtr 1 |
| 16 | | | | |
| 17 | | | | |
| 18 | | | | |

Note: visit our page about removing duplicates to learn more about this great Excel tool.

2 Drop-down List: Drop-down lists in Excel are helpful if you want to be sure that users select an item from a list, instead of typing their own values.

3 Vlookup: The VLOOKUP function is one of the most popular functions in Excel. This page contains many easy to follow VLOOKUP examples.

# VLOOKUP

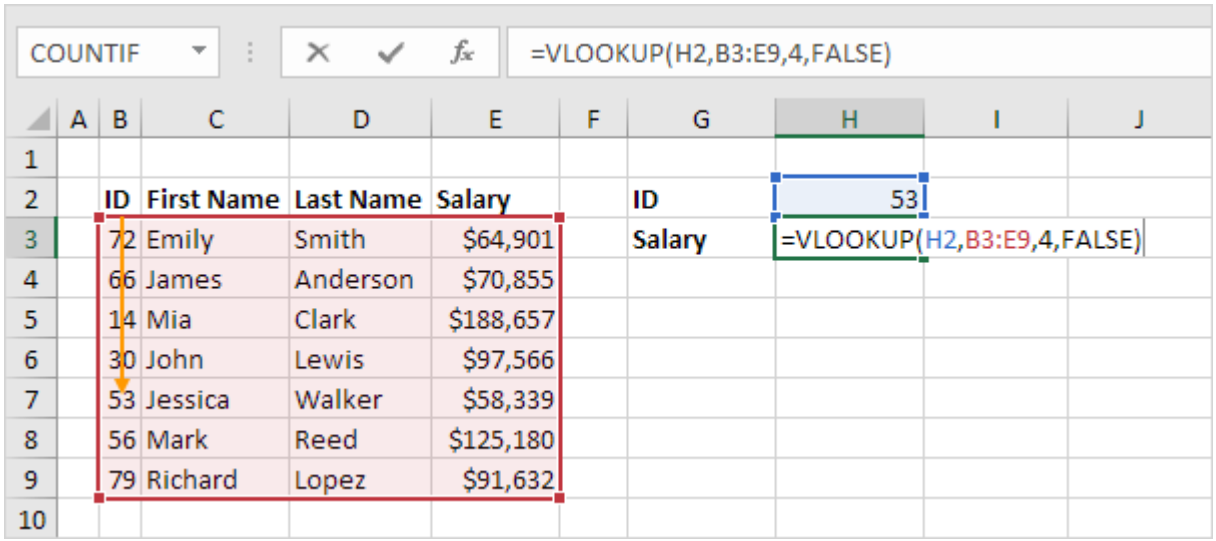
Exact Match | Approximate Match | Vlookup Looks Right | First Match | Partial Match | Vlookup is Case-insensitive | Multiple Criteria | #N/A error | Multiple Lookup Tables | Index and Match | Table Magic | Xlookup

The **VLOOKUP function** is one of the most popular functions in **Excel**. This page contains many easy to follow VLOOKUP examples.

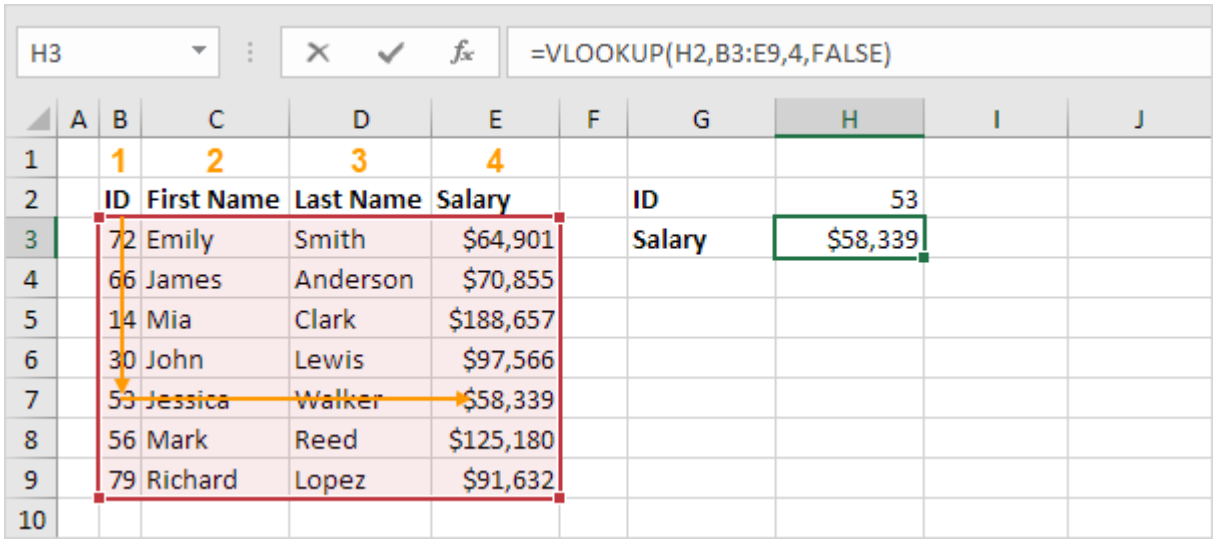
# Exact Match giver case sensitive match

Most of the time you are looking for an exact match when you use the VLOOKUP function in Excel. Let's take a look at the arguments of the VLOOKUP function.

1. The simple VLOOKUP function below looks up the value 53 (first argument) in the leftmost column of the red table (second argument).




2. The value 4 (third argument) tells the VLOOKUP function to return the value in the same row from the fourth column of the red table.



Note: the Boolean FALSE (fourth argument) tells the VLOOKUP function to return an exact match. If the VLOOKUP function cannot find the value 53 in the first column, it will return a #N/A error.

4 Histogram: This example teaches you how to make a histogram in Excel.
5 Regression: This page teaches you how to run a linear regression analysis in Excel and how to interpret the Summary Output.
6 Percent Change: The percent change formula is used very often in Excel. For example, let's use this formula to calculate the Monthly Change and Total Change.
7 Pareto Chart: A Pareto chart combines a column chart and a line graph. The Pareto principle states that, for many events, roughly 80% of the effects come from 20% of the causes.
8 Loan Amortization Schedule: This example teaches you how to create a loan amortization schedule in Excel.
9 Random Numbers: Excel has two very useful functions when it comes to generating random numbers. RAND and RANDBETWEEN.
10 Remove Duplicates: This example teaches you how to remove duplicates in Excel.
11 If: The IF function is one of the most used functions in Excel. This page contains many easy to follow IF examples.
12 Lock Cells: You can lock cells in Excel if you want to protect cells from being edited.

# *Lock Cells

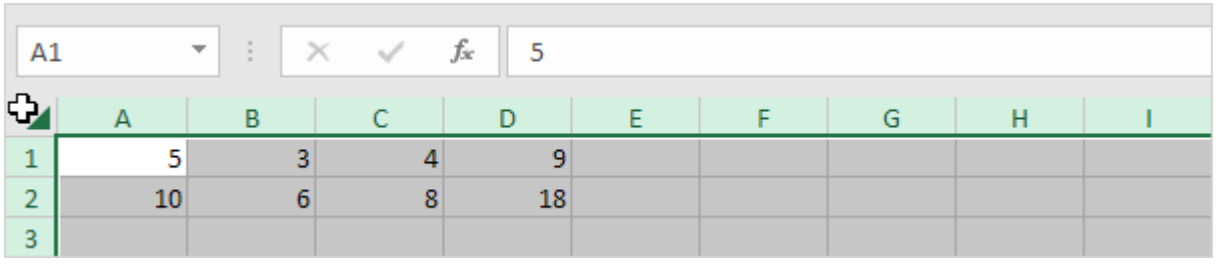
Lock All Cells | Lock Specific Cells | Lock Formula Cells

You can **lock cells** in **Excel** if you want to protect cells from being edited.

## Lock All Cells

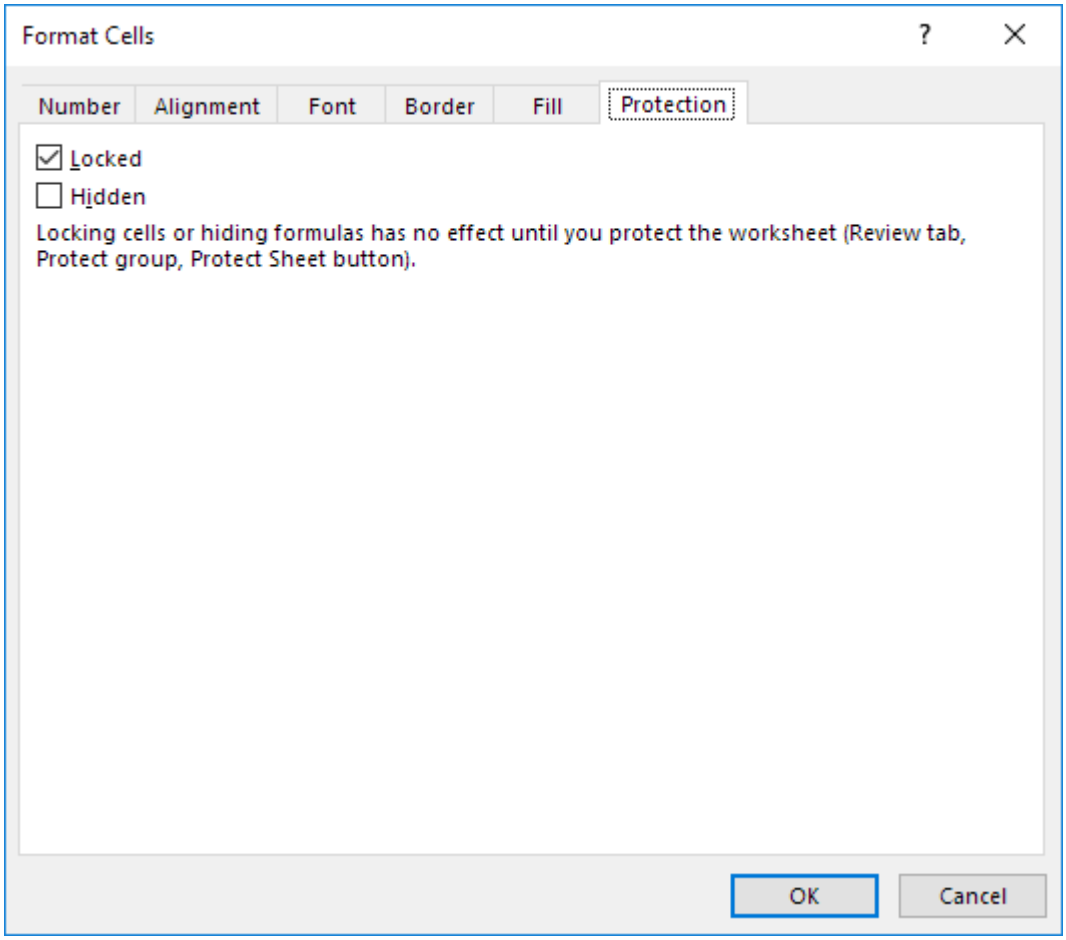
By default, all cells in Excel are **locked**. However, locking cells has no effect until you protect the worksheet.

1. Select all cells.




2. Right click, and then click Format Cells (or press CTRL + 1).

3. On the Protection tab, you can verify that all cells are locked by default.




4. Click OK or Cancel.

5. Protect the sheet.

# Celle-referencer

Eksempler

Eksempel 1

I eksemplerne bruges funktionen INDEKS til at finde værdien i den celle, hvor en række og en kolonne krydser hinanden.

Kopiér eksempeldataene i følgende tabel, og sæt dem ind i celle A1 i et nyt Excel-regneark. For at få formlerne til at vise resultater skal du markere dem, trykke på **F2** og derefter trykke på **Enter**.

| Data=1,1 | Data | |
|---|---|---|
| Æbler=A2 | Citroner | |
| Bananer=A3 | Pærer=B3 | |
| **Formel** | **Beskrivelse** | **Res** |
| =INDEKS(A2:B3;2;2) | Værdi ved skæringen af anden række og anden kolonne i området A2:B3. | Pær |
| =INDEKS(A2:B3;2;1) | Værdi ved skæringen af anden række og første kolonne i området A2:B3. | Ban |

Eksempel 2

I dette eksempel bruges funktionen INDEKS i en matrixformel til at finde værdierne i to celler, der er angivet en 2x2-matrix.

**Bemærk!:** Hvis du har en aktuel version af Microsoft 365, kan du indtaste formlen i cellen øverst til venstre i outputområdet og derefter trykke på **Enter** for at bekræfte formlen som en dynamisk matrixformel. Ellers skal formlen angives som en ældre matrixformel ved først at markere to tomme celler, indtaste formlen i cellen øverst til venstre i outputområdet og derefter trykke på **CTRL+SKIFT+ENTER** for at bekræfte den. Excel indsætter klammeparenteser for dig i begyndelsen og slutningen af formlen. Se Retningslinjer for og eksempler på matrixformler for at få mere at vide om matrixformler.

| Formel | Beskrivelse | R |
|---|---|---|
| =INDEKS({1,2;3,4},0,2) | Den værdi, der findes i den første række og anden kolonne i matrixen. Matrixen indeholder 1 og 2 i den første række og 3 og 4 i den anden række. virker ikke | 2 |
| | Den værdi, der findes i den anden række, anden kolonne i matrixen (samme mat virker ikke rix som herover). | 4 |

Toppen af siden

Referenceformular

Beskrivelse

Returnerer referencen fra cellen i skæringspunktet mellem en bestemt række og kolonne. Hvis referencen består af ikke-tilstødende markeringer, kan du vælge den markering, der skal søges i.

Syntaks

# INDEKS(reference;rækkenr;[kolonnenr];[områdenr]) definerer område

Indeksfunktionens referenceformular har følgende argumenter:

- **reference** Påkrævet. En reference til et eller flere celleområder.
    - Hvis du angiver et område, der ikke støder op til referencen, skal du sætte referencen i parenteser.
    - Hvis hvert område i referencen kun indeholder én række eller kolonne, er henholdsvis row_num eller column_num argument valgfrit. For en enkelt rækkereference skal du f.eks. bruge INDEKS(reference,,kolonne).
- **row_num** Påkrævet. Nummeret på rækken i den reference, der skal returneres en reference fra.
- **column_num** Valgfrit. Nummeret på kolonnen i den reference, der skal returneres en reference fra.
- **area_num** Valgfrit. Vælger et område i referencen, som skæringspunktet mellem row_num og column_num skal returneres fra. Det første område, der markeres eller angives, tildeles nummer 1, andet er nummer 2 osv. Hvis area_num udelades, bruger INDEKS område 1. De områder, der er angivet her, skal alle være placeret på ét ark. Hvis du angiver områder, der ikke er på samme ark som hinanden, vil det medføre en #VÆRDI! som fejl. Hvis du skal bruge områder, der er placeret på forskellige ark, anbefales det, at du bruger matrixformen i indeksfunktionen, og bruger en anden funktion til at beregne det område, der udgør matrixen. Du kan f.eks bruge funktionen VÆLG til at beregne, hvilket område der skal bruges.

Hvis Reference f.eks. beskriver cellerne (A1:B4,D1:E4,G1:H4), er area_num 1 området A1:B4, area_num 2 er området D1:E4 og area_num 3 er området G1:H4.

Bemærkninger

- Når referencen og area_num har markeret et bestemt område, markerer row_num og column_num en bestemt celle: row_num 1 er den første række i området, column_num 1 er den første kolonne osv. Den reference, der returneres af INDEX, er skæringspunktet mellem row_num og column_num.
- Hvis du angiver row_num eller column_num til 0 (nul), returnerer INDEKS referencen for henholdsvis hele kolonnen eller rækken.
- row_num, column_num og area_num skal pege på en celle i referencen; ellers returnerer INDEKS et #REF! fejl. Hvis row_num og column_num udelades, returnerer INDEKS det område i reference, der er angivet af area_num.
- Resultatet af funktionen INDEKS er en reference og fortolkes sådan af andre formler. Afhængigt af formlen kan den returnerede værdi af INDEKS bruges som en reference eller en værdi. F.eks. svarer formlens CELLE("bredde",INDEKS(A1:B2,1,2)) til CELLE("bredde",B1). Funktionen CELLE bruger den returnerede værdi af INDEKS som en cellereference. En formel som 2*INDEKS(A1:B2;1;2) oversætter derimod den returnerede værdi af INDEKS til tallet i celle B1.

Eksempler

Kopiér eksempeldataene i følgende tabel, og sæt dem ind i celle A1 i et nyt Excel-regneark. For at få formlerne til at vise resultater skal du markere dem, trykke på F2 og derefter trykke på Enter.

| Frugt=A1 | Pris | Antal |
|---|---|---|
| Æbler=A2 | DKK 0,69 | 40 |
| Bananer | DKK 0,34 | 38 |
| Citroner | DKK 0,55 | 15 |
| Appelsiner | DKK 0,25 | 25 |
| Pærer | DKK 0,59 | 40 |
| Mandler | DKK 2,80 | 10 |
| Cashewnødder=A8 | DKK 3,55 | 16 |
| Jordnødder | DKK 1,25 | 20 |
| Valnødder | DKK 1,75=B10 | 12 |
| **Formel** | **Beskrivelse** | **Resulta** |
| =INDEKS(A2:C6; 2; 3) | Skæringen mellem anden række og tredje kolonne i området A2:C6, der er indholdet af celle C3. | 38 |
| =INDEKS((A1:C6; A8:C11), 2; 2; 2) | Skæringen mellem anden række og anden kolonne i det andet område af A8:C11, der er indholdet af celle B9. | 1,25 |
| =SUM(INDEKS(A1:C11; 0; 3; 1)) | Summen af tredje kolonne i det første område i området A1:C11, der er summen af C1:C11. rownr=3 må betyde at alle rækker vælges | 216 |
| =SUM(B2:INDEKS(A2:C6; 5; 2)) | Summen af området, der starter fra B2 og slutter ved skæringen af den femte række og den anden kolonne i området A2:A6, som er summen af B2:B6. | 2,42 |

Toppen af siden

Se også

Funktionen LOPSLAG

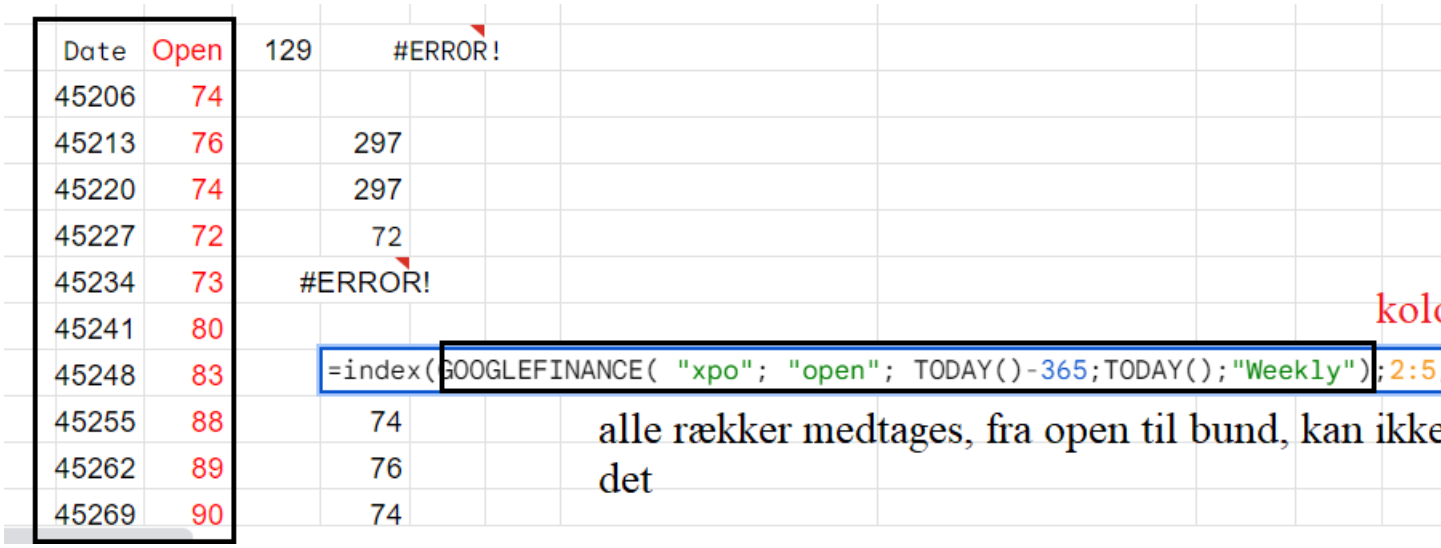
Funktionen SAMMENLIGN

Funktionen INDIREKTE

Retningslinjer for og eksempler på matrixformler

Opslags- og referencefunktioner (reference)

# Google-sheet




**=MAX(index(GOOGLEFINANCE( "xpo"; "open"; TODAY()-5*365;TODAY();"Weekly") ;;2)) finder maxværiden I kolonne 2 for området defineret som**

**=MAX(INDEX(GOOGLEFINANCE( "xpo"; "open"; TODAY()-365;TODAY();"Weekly"); 0; 2; 1))**

# Funktionen FORSKYDNING/offset giver mulighed for definition af område fra definret referencecelle og højde,bredde

https://www.excel-easy.com/examples/offset.html#google_vignette

Funktionen forskydning bruges til at henvise til et specifikt område eller celle ved at bruge en anden celle som et startpunkt. Den kan have op til 5 argumenter.

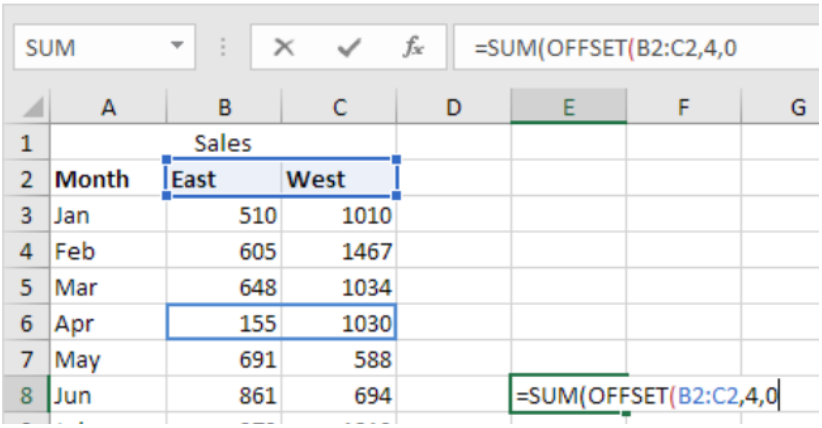
- Reference: er den celle, hvorfra du vil basere forskydningen.
- Rækker: er antallet af rækker, som du ønsker, at referencecellen skal flytte forbi.
- Kolonner: er antallet af kolonner, som du ønsker, at referencecellen skal flytte forbi.
- Højde (valgfrit): er antallet af rækker, du vil have området skal være.
- Bredde (valgfrit): er antallet af kolonner, du vil have området skal være.

Som et simpelt eksempel, se på formlen herunder:

=FORSKYDNING(E2, -1,0,1,1)

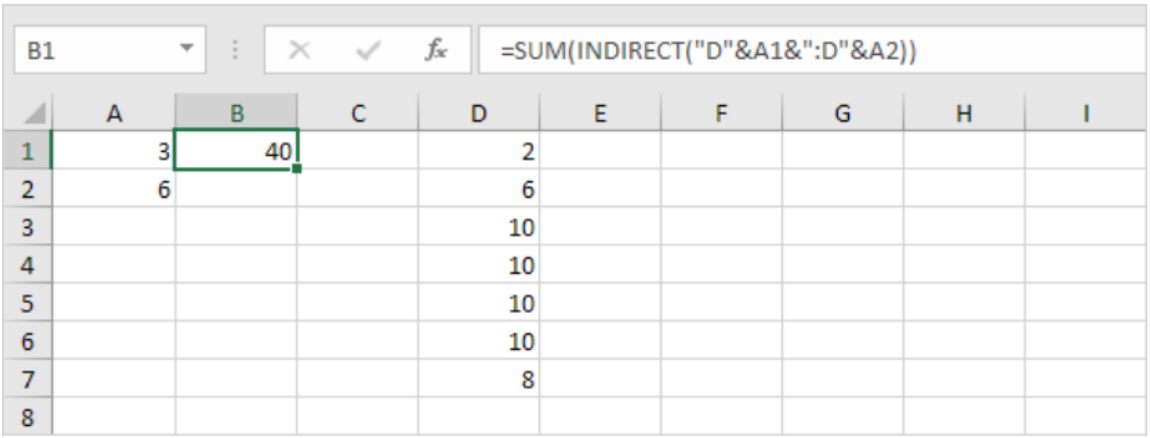
Formlen ovenfor ville starte ved celle E2, bevæge sig op med -1, bevæge sig højre ved 0 og skabe et område, der er 1 celle højt og 1 celle bredt. Resultatet heraf ville være Cell E1.

**Højde=bredde=0:**




# Indirect omdanner string til cell adresser

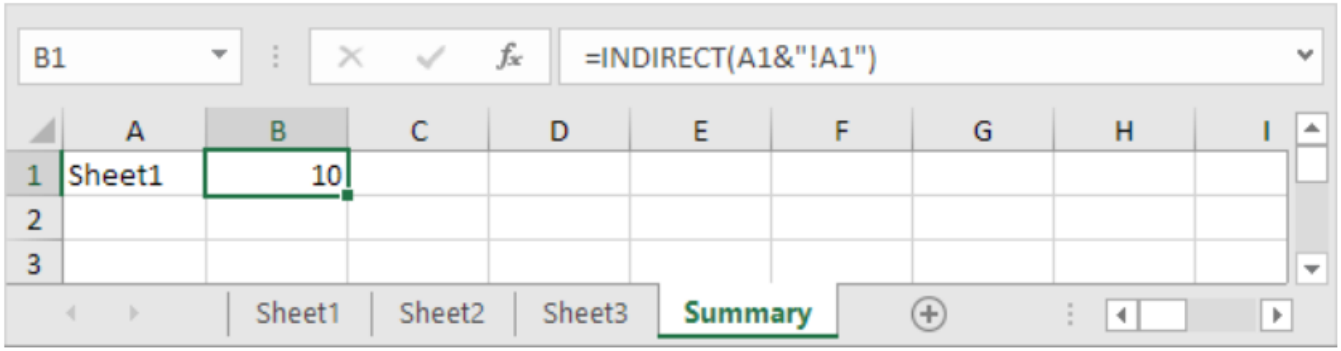
Use the & operator to join the string "D" with the value in cell A1(1), INDIRECT("D"&A1) -> INDIRECT("D1") ->D1 cellen




Explanation: the formula above reduces to =SUM(INDIRECT("D3:D6")). The INDIRECT function

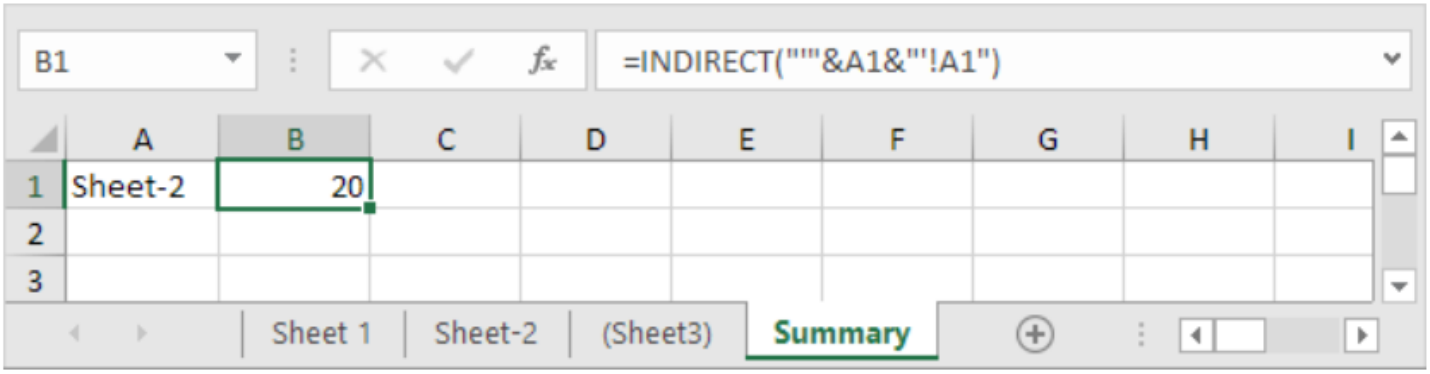
=AVERAGE(INDIRECT("Scores")) reduces to =AVERAGE(Scores), hvor Score er named range

2. On the Summary sheet, enter the INDIRECT function shown below. Use the & operator to join the sheet name in cell A1 with "!A1".

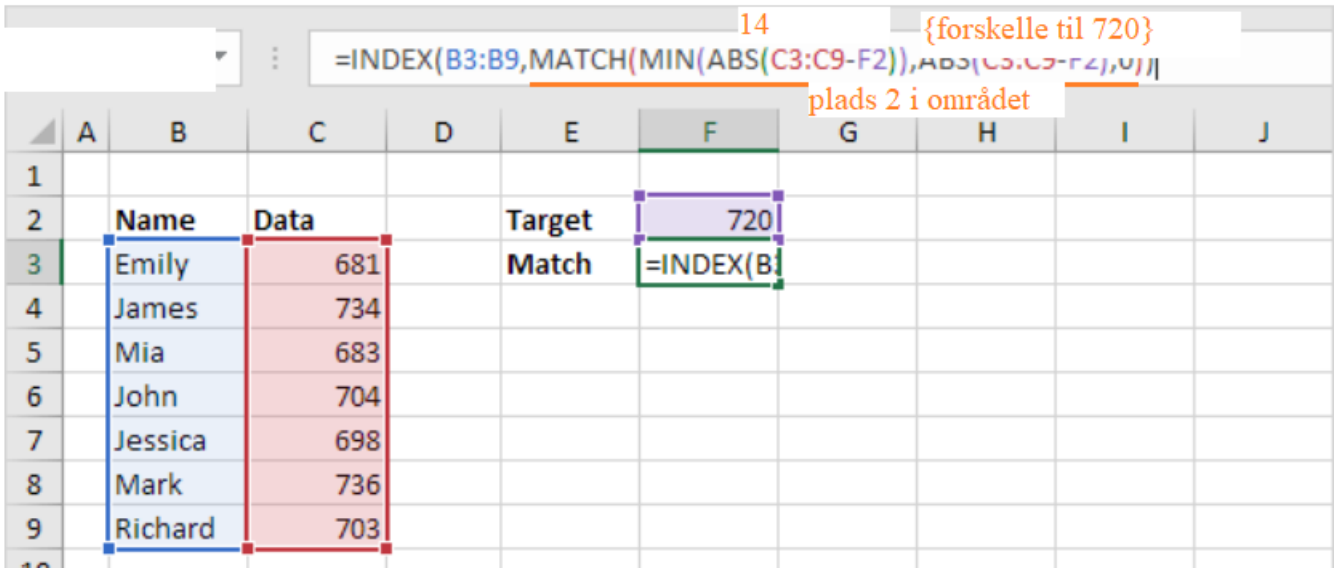



Explanation: the formula above reduces to =INDIRECT("Sheet1!A1"). The INDIRECT function converts the text string "Sheet1!A1" into a valid worksheet reference. In other words,

3. If your sheet names contain spaces or other special characters, enclose the sheet name in single quotation marks. Modify the INDIRECT function as shown below.



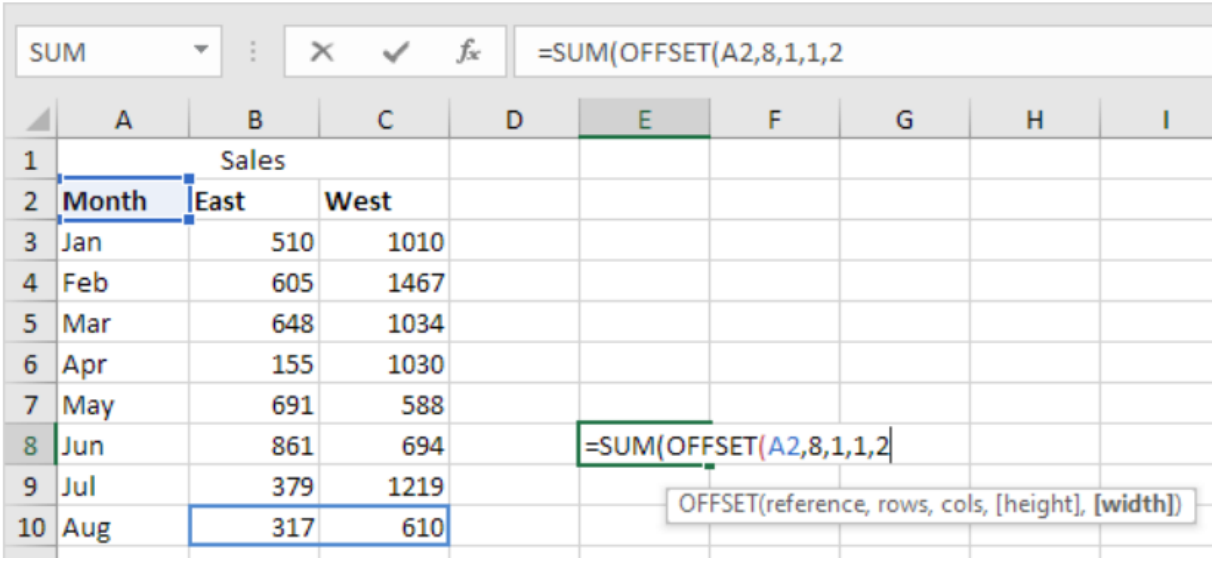

# Find cellen med værdi nærmest søgeværdi




# Return a Range of cells med offset

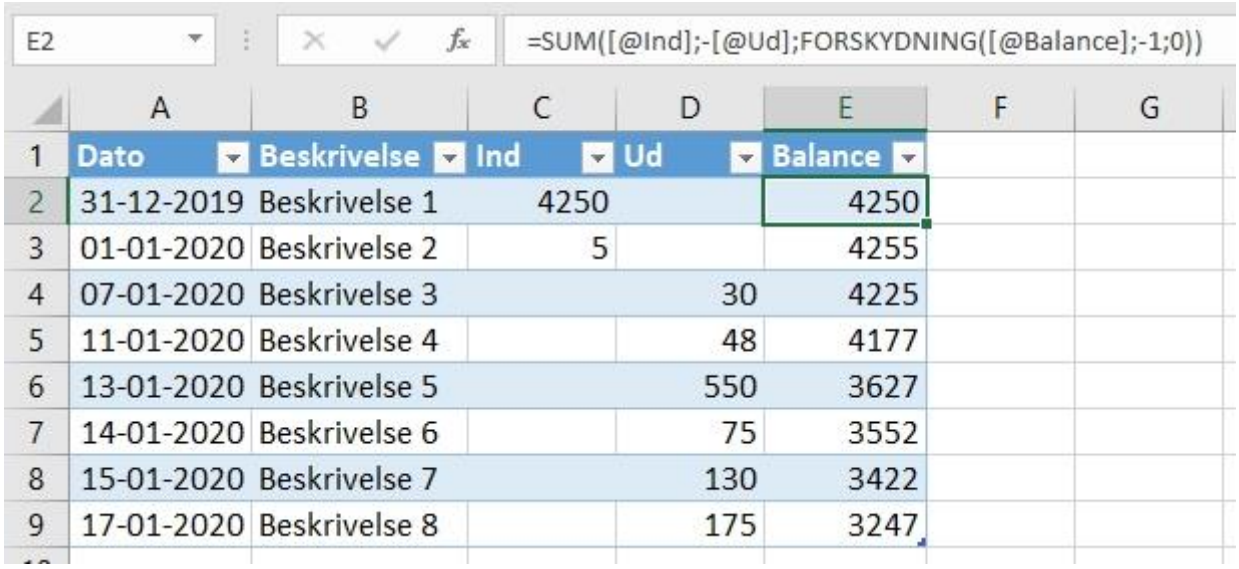
Alright. Let's now use the OFFSET function in Excel to return a range (two or more cells).

1. The OFFSET function below returns the 1 x 2 range that is 8 rows below and 1 column to the right of cell A2. The SUM function calculates the sum of this range.




giver 317+610=927

Vi kan inkludere denne metode i vores løbende total formel.




| | A | B | C | D | E | F | G |
|---|---|---|---|---|---|---|---|
| 1 | Dato | Beskrivelse | Ind | Ud | Balance | | |
| 2 | 31-12-2019 | Beskrivelse 1 | 4250 | | 4250 | | |
| 3 | 01-01-2020 | Beskrivelse 2 | 5 | | 4255 | | |
| 4 | 07-01-2020 | Beskrivelse 3 | | 30 | 4225 | | |
| 5 | 11-01-2020 | Beskrivelse 4 | | 48 | 4177 | | |
| 6 | 13-01-2020 | Beskrivelse 5 | | 550 | 3627 | | |
| 7 | 14-01-2020 | Beskrivelse 6 | | 75 | 3552 | | |
| 8 | 15-01-2020 | Beskrivelse 7 | | 130 | 3422 | | |
| 9 | 17-01-2020 | Beskrivelse 8 | | 175 | 3247 | | |

'=SUM(FORSKYDNING(B1; 0; 0; 4; 2)) sum af området 4x2, 4 rækker,2 søjler fra B1

=MIN(OFFSET(G150; 0; 1; 4; 1)): finder minimum i området fra H150, 4x1