EXCEL MACRO MASTERY
(HTTPS://EXCELMACROMASTERY.COM/)

THE MISSING VBA HANDBOOK

The Complete Guide to Ranges and Cells in Excel VBA

JANUARY 2, 2015 (HTTPS://EXCELMACROMASTERY.COM/EXCEL-VBA-RANGE-CELLS/) BY PAUL KELLY (HTTPS://EXCELMACROMASTERY.COM/AUTHOR/ADMIN/) · 91 COMMENTS (HTTPS://EXCELMACROMASTERY.COM/EXCEL-VBA-RANGE-CELLS/#COMMENTS)

"It is a capital mistake to theorize before one has data"- Sir Arthur Conan Doyle

This post covers everything you need to know about using **Cells** and **Ranges** in VBA. You can read it **from start to finish** as it is layed out in a logical order. If you prefer you can use the table of contents below to go to a section of your choice.

If you want information about using **Range** then check out the Range property section.

For more information on using the **Cells** go to the Cells property section.

Other topics included are Offset property, reading values between cells, reading values to arrays and formatting cells.

Contents [hide]

adbox/145db6b73f72a2%3A106f25298346dc/5676073085829120/)

A Quick Guide to Ranges and Cells

Function	Takes	Returns	Example	Gives
Range	cell address	multiple cells	.Range("A1:A4")	$A$1:$A$4
Cells	row, column	one cell	.Cells(1,5)	$E$1
Offset	row, column	multiple cells	Range("A1:A2") .Offset(1,2)	$C$2:$C$3
Rows	row(s)	one or more rows	.Rows(4) .Rows("2:4")	$4:$4 $2:$4
Columns	column(s)	one or more columns	.Columns(4) .Columns("B:D")	$D:$D $B:$D

ks-

Introduction

This is the third post dealing with the three main elements of VBA. These three elements are the Workbooks (http://excelmacromastery.com/complete-guide-excel-vba-workbook/), Worksheets (http://excelmacromastery.com/the-complete-guide-to-worksheets-in-excel-vba/) and Ranges/Cells. Cells are by far the most important part of Excel. Almost everything you do in Excel starts and ends with Cells.

Generally speaking, you do three main things with Cells

omment-

1. Read
2. Write
3. Change the format

omment-

Excel has a number of methods for accessing cells such as **Range**, **Cells** and **Offset**. “Why do I need them”, “When should you use them?”,”Which is best ?” are questions I am often asked.

In this post I will fully investigate each one of these methods of access and provide you with answers to those questions.

Let’s start with the simplest method of accessing cells – using the Range property of the worksheet.

The Range Property

The worksheet has a Range property which you can use to access cells in VBA. The Range property takes the same argument that most Excel Worksheet functions take e.g. “A1”, “A3:C6” etc.

The following example shows you how to place a value in a cell using the Range property.

```
Public Sub WriteToCell()

    ' Write number to cell A1 in sheet1 of this workbook
    ThisWorkbook.Worksheets("Sheet1").Range("A1") = 67

    ' Write text to cell A2 in sheet1 of this workbook
    ThisWorkbook.Worksheets("Sheet1").Range("A2") = "John Smith"

    ' Write date to cell A3 in sheet1 of this workbook
    ThisWorkbook.Worksheets("Sheet1").Range("A3") = #11/21/2017#

End Sub
```

As you can see Range is a member of the worksheet which in turn is a member of the Workbook. This follows the same hierarchy as in Excel so should be easy to understand. To do something with Range you must first specify the workbook and worksheet it belongs to.

For the rest of this post I will use the code name of the sheet. This makes the code clearer as I will not need to specify the workbook each time. You can use a sheet directly with the code name as long as it is in the current workbook.

You can see the Code Name (http://excelmacromastery.com/the-complete-guide-to-worksheets-in-excel-vba/#Using_the_code_name_of_a_Worksheet)of the sheet in the VBAProject window. It is the name outside the parenthesis.

The following code shows the above example using the Code Name (http://excelmacromastery.com/the-complete-guide-to-worksheets-in-excel-vba/#Using_the_code_name_of_a_Worksheet) of the worksheet.

```
Public Sub UsingCodeName()

    ' Write number to cell A1 in sheet1 of this workbook
    cnSheet1.Range("A1") = 67

    ' Write text to cell A2 in sheet1 of this workbook
    cnSheet1.Range("A2") = "John Smith"

    ' Write date to cell A3 in sheet1 of this workbook
    cnSheet1.Range("A3") = #11/21/2017#

End Sub
```

I will use the worksheet code name in the rest of the examples. It makes the code much easier to read.

You can also write to multiple cells using the Range property

```
Public Sub WriteToMulti()

    ' Write number to a range of cells
    cnSheet1.Range("A1:A10") = 67

    ' Write text to multiple ranges of cells
    cnSheet1.Range("B2:B5,B7:B9") = "John Smith"

End Sub
```

CLICK HERE TO DOWNLOAD THE FREE PDF VERSION OF THIS POST (https://excelmacromastery.leadpages.co/leadbox/145bdb773f72a2%3A106f252983

The Cells Property of the Worksheet

The worksheet object has another property called **Cells** which is very similar to range. There are two differences

1. Cells returns a range of one cell only
2. Cells takes row and column as arguments

The example below shows you how to write values to cells using both the Range and Cells property

```
Public Sub UsingCells()

    ' Write to A1
    cnSheet1.Range("A1") = 10
    cnSheet1.Cells(1, 1) = 10

    ' Write to A10
    cnSheet1.Range("A10") = 10
    cnSheet1.Cells(10, 1) = 10

    ' Write to E1
    cnSheet1.Range("E1") = 10
    cnSheet1.Cells(1, 5) = 10

End Sub
```

You may be wondering when you should use Cells and when you should use Range. Using Range is useful for accessing the same cells each time the Macro runs.

For example, if you were using a Macro to calculate a total and write it to cell A10 every time then Range would be suitable for this task.

Using the Cells property is useful if you are accessing a cell based on a number that may vary. It is easier explain this with an example.

The following code finds the first blank cell in the first spreadsheet row and writes text to it.

```
Public Sub WriteToFirstBlankCell()

    ' Get last column from left that is not blank
    Dim lLastCol As Integer
    lLastCol = cnSheet1.Range("A1").End(xlToRight).Column

    ' If reaches the last column, there is not cell with data
    lLastCol = IIf(lLastCol = cnSheet1.Columns.Count, 1, lLastCol)

    ' Write text to first blank cell in Row 1
    cnSheet1.Cells(1, lLastCol + 1) = "John Smith"

End Sub
```

In this example we have the number of the column and the row.

To use Range here would require us to convert these values to the letter/number cell reference e.g. “C1”. Using the Cells property allows us to provide a row and a column number to access a cell.

Sometimes you may want to return more than one cell using row and column numbers. The next section shows you how to do this.

Using Cells and Range together

As you have seen you can only access one cell using the Cells property. If you want to return a range of cells then you can use Cells with Ranges as follows

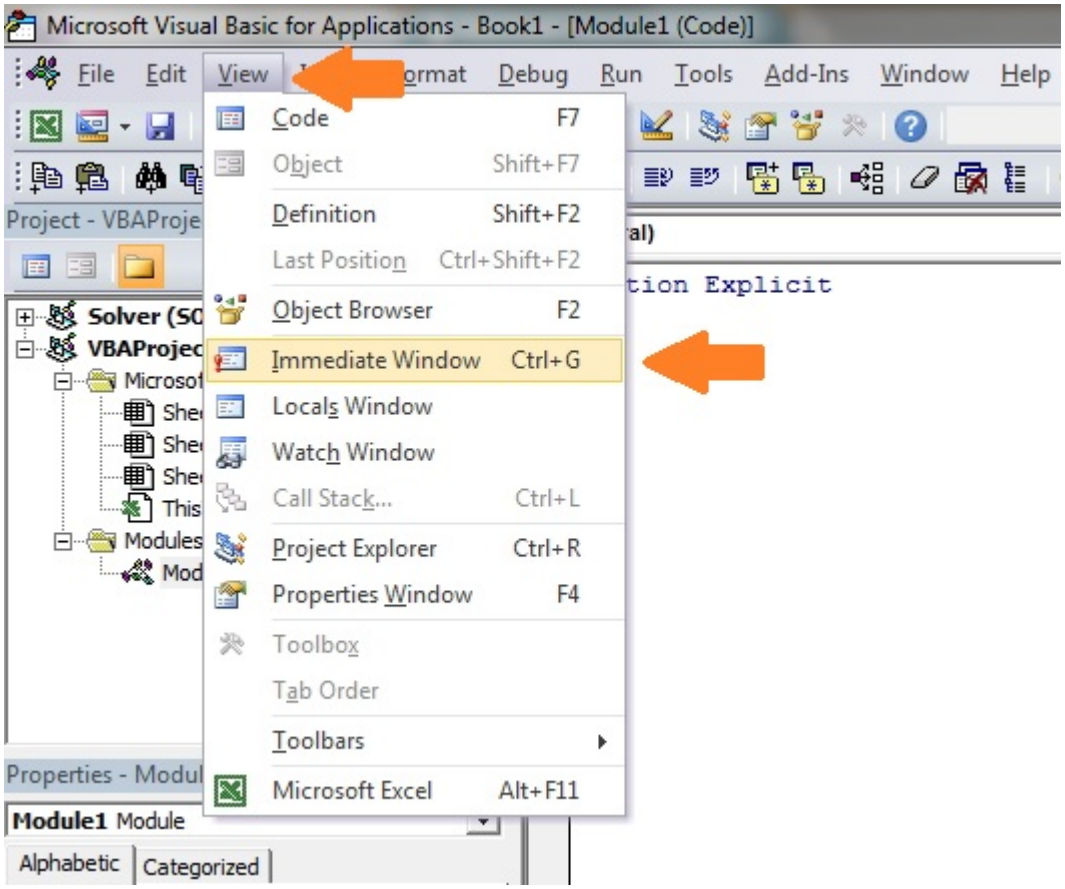
```
Public Sub UsingCellsWithRange()

    With cnSheet1
        ' Write 5 to Range A1:A10 using Cells property
        .Range(.Cells(1, 1), .Cells(10, 1)) = 5

        ' Format Range B1:Z1 to be bold
        .Range(.Cells(1, 2), .Cells(1, 26)).Font.Bold = True

    End With

End Sub
```

As you can see, you provide the start and end cell of the Range. Sometimes it can be tricky to see which range you are dealing with when the value are all numbers. Range has a property called **Address** which displays the letter/ number cell reference of any range. This can come in very handy when you are debugging or writing code for the first time.

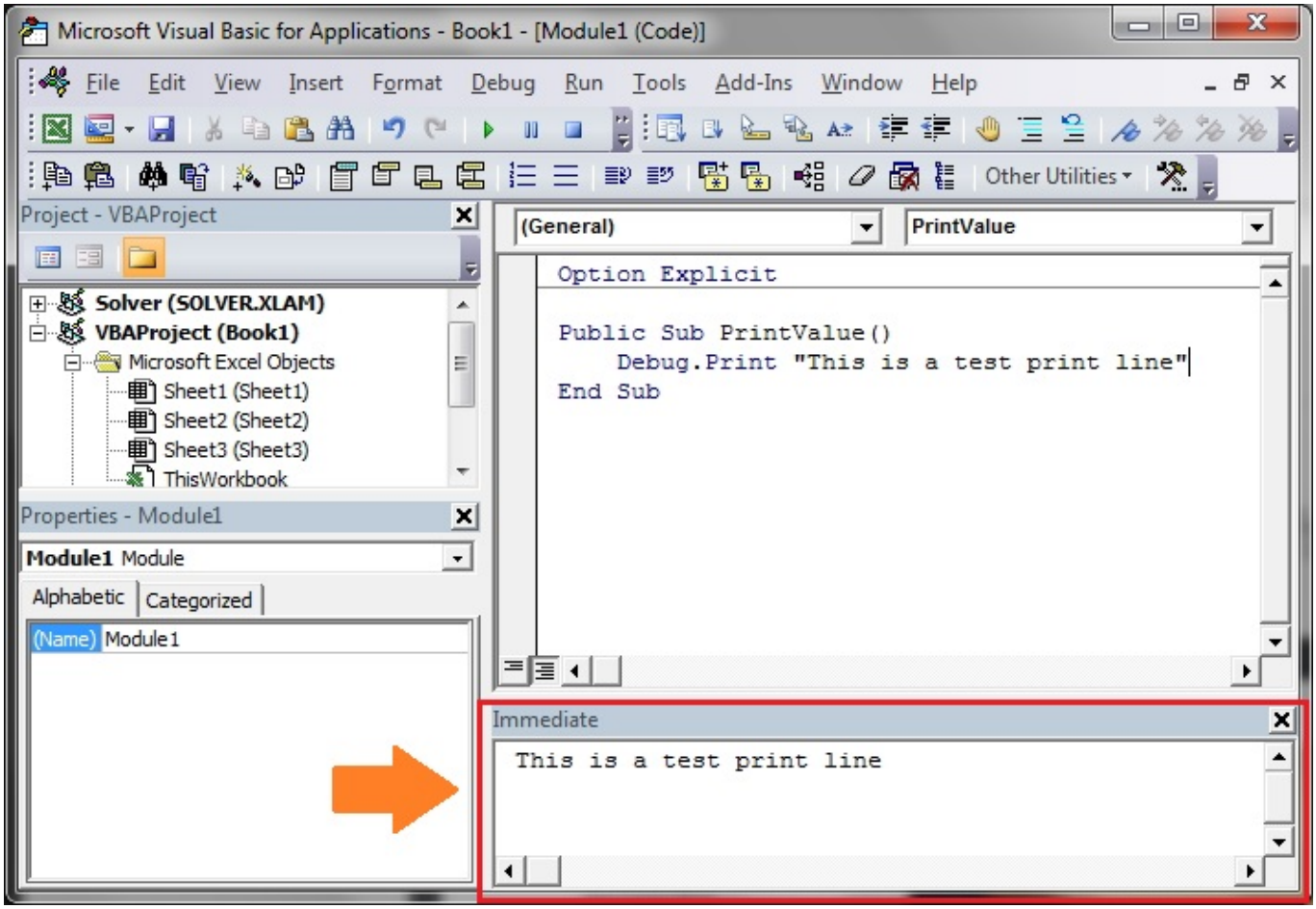
In the following example we print out the address of the ranges we are using.

```
Public Sub ShowRangeAddress()

    ' Note: Using underscore allows you to split up lines of code
    With cnSheet1

        ' Write 5 to Range A1:A10 using Cells property
        .Range(.Cells(1, 1), .Cells(10, 1)) = 5
        Debug.Print "First address is : " _
            + .Range(.Cells(1, 1), .Cells(10, 1)).Address

        ' Format Range B1:Z1 to be bold
        .Range(.Cells(1, 2), .Cells(1, 26)).Font.Bold = True
        Debug.Print "Second address is : " _
            + .Range(.Cells(1, 2), .Cells(1, 26)).Address

    End With

End Sub
```

In the example I used **Debug.Print** to print to the Immediate Window. To view this window select View->Immediate Window(or Ctrl G)



(http://excelmacromastery.com/wp-content/uploads/2014/12/ImmediateWindow2.jpg)



(http://excelmacromastery.com/wp-content/uploads/2014/12/ImmediateSampeText.jpg)

The Offset Property of Range

Range has a property called Offset. The term Offset refers to a count from the original position. It is used a lot in certain areas of programming. With the Offset property you can get a Range of cells the same size and a certain distance from the current range. The reason this is useful is that sometimes you may want to select a Range based on a certain condition. For example in the screenshot below there is a column for each day of the week. Given the day number(i.e. Monday=1, Tuesday=2 etc.) we need to write the value to the correct column.

G	H	I	J	K	L	M	N
	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday	Sunday
	€111.01		€456.99		€432.25		€710.17

We will first attempt to do this without using Offset.

```
' This sub tests with different values
Public Sub TestSelect()

    ' Monday
    SetValueSelect 1, 111.21
    ' Wednesday
    SetValueSelect 3, 456.99
    ' Friday
    SetValueSelect 5, 432.25
    ' Sunday
    SetValueSelect 7, 710.17

End Sub

' Writes the value to a column based on the day
Public Sub SetValueSelect(lDay As Long, lValue As Currency)

    Select Case lDay
        Case 1: cnSheet1.Range("H3") = lValue
        Case 2: cnSheet1.Range("I3") = lValue
        Case 3: cnSheet1.Range("J3") = lValue
        Case 4: cnSheet1.Range("K3") = lValue
        Case 5: cnSheet1.Range("L3") = lValue
        Case 6: cnSheet1.Range("M3") = lValue
        Case 7: cnSheet1.Range("N3") = lValue
    End Select

End Sub
```

As you can see in the example, we need to add a line for each possible option. This is not an ideal situation. Using the Offset Property provides a much cleaner solution

```
' This sub tests with different values
Public Sub TestOffset()

    DayOffSet 1, 111.01
    DayOffSet 3, 456.99
    DayOffSet 5, 432.25
    DayOffSet 7, 710.17

End Sub

Public Sub DayOffSet(lDay As Long, lValue As Currency)

    ' We use the day value with offset specify the correct column
    cnSheet1.Range("G3").Offset(, lDay) = lValue

End Sub
```

As you can see this solution is much better. If the number of days in increased then we do not need to add any more code. For Offset to be useful there needs to be some kind of relationship between the positions of the cells. If the Day columns in the above example were random then we could not use Offset. We would have to use the first solution.

One thing to keep in mind is that Offset retains the size of the range. So .Range(“A1:A3”).Offset(1,1) returns the range B2:B4. Below are some more examples of using Offset

```
Public Sub UsingOffset()

    ' Write to B2 - no offset
    cnSheet1.Range("B2").Offset() = "Cell B2"

    ' Write to C2 - 1 column to the right
    cnSheet1.Range("B2").Offset(, 1) = "Cell C2"

    ' Write to B3 - 1 row down
    cnSheet1.Range("B2").Offset(1) = "Cell B3"

    ' Write to C3 - 1 column right and 1 row down
    cnSheet1.Range("B2").Offset(1, 1) = "Cell C3"

    ' Write to A1 - 1 column left and 1 row up
    cnSheet1.Range("B2").Offset(-1, -1) = "Cell A1"

    ' Write to range E3:G13 - 1 column right and 1 row down
    cnSheet1.Range("D2:F12").Offset(1, 1) = "Cells E3:G13"

End Sub
```

Using Rows and Columns as Ranges

If you want to do something with an entire Row or Column you can use the Rows or Columns property of the Worksheet. They both take one parameter which is the row or column number you wish to access

```
Public Sub UseRowAndColumns()

    ' Set the font size of column B to 9
    cnSheet1.Columns(2).Font.Size = 9

    ' Set the width of columns D to F
    cnSheet1.Columns("D:F").ColumnWidth = 4

    ' Set the font size of row 5 to 18
    cnSheet1.Rows(5).Font.Size = 18

End Sub
```

Using Range in place of Worksheet

You can also use Cells, Rows and Columns as part of a Range rather than part of a Worksheet. You may have a specific need to do this but otherwise I would avoid the practice. It makes the code more complex. Simple code is your friend. It reduces the possibility of errors.

The code below will set the second column of the range to bold. As the range has only two rows the entire column is considered B1:B2

```
Public Sub UseColumnsInRange()

    ' This will set B1 and B2 to be bold
    cnSheet1.Range("A1:C2").Columns(2).Font.Bold = True

End Sub
```

Reading Values from one Cell to another

In most of the examples so far we have written values to a cell. We do this by placing the range on the left of the equals sign and the value to place in the cell on the right. To write data from one cell to another we do the same. The destination range goes on the left and the source range goes on the right.

The following example shows you how to do this

```
Public Sub ReadValues()

    ' Place value from B1 in A1
    cnSheet1.Range("A1") = cnSheet1.Range("B1")

    ' Place value from B3 in sheet2 to cell A1
    cnSheet1.Range("A1").Value = cnSheet2.Range("B3")

    ' Place value from B1 in cells A1 to A5
    cnSheet1.Range("A1:A5") = cnSheet1.Range("B1")

    ' You need to use the "Value" property to read multiple cells
    cnSheet1.Range("A1:A5") = cnSheet1.Range("B1:B5").Value

End Sub
```

As you can see from this example it is not possible to read from multiple cells. If you want to do this you can use the Copy function of Range with the Destination parameter

```
Public Sub CopyValues()

    ' Store the copy range in a variable
    Dim rgCopy As Range
    Set rgCopy = cnSheet1.Range("B1:B5")

    ' Use this to copy from more than one cell
    rgCopy.Copy Destination:=cnSheet1.Range("A1:A5")

    ' You can paste to multiple destinations
    rgCopy.Copy Destination:=cnSheet1.Range("A1:A5,C2:C6")

End Sub
```

The Copy function copies everything including the format of the cells. It is the same result as manually copying and pasting a selection. You can see more about it in the Copying and Pasting Cells section.

Reading Values to variables

The last section showed you how to read from one cell to another. You can also read from a cell to a variable. A variable is used to store values while a Macro is running. You normally do this when you want to manipulate the data before writing it somewhere. The following is a simple example using a variable. As you can see the value of the item to the right of the equals is written to the item to the left of the equals.

```
Public Sub UseVar()

    ' Create
    Dim val As Integer

    ' Read number from cell
    val = cnSheet1.Range("A1")

    ' Add 1 to value
    val = val + 1

    ' Write new value to cell
    cnSheet1.Range("A2") = val

End Sub
```

To read text to a variable you use a variable of type **String**.

```
Public Sub UseVarText()

    ' Declare a variable of type string
    Dim sText As String

    ' Read value from cell
    sText = cnSheet1.Range("A1")

    ' Write value to cell
    cnSheet1.Range("A2") = sText

End Sub
```

You can write a variable to a range of cells. You just specify the range on the left and the value will be written to all cells in the range.

```
Public Sub VarToMulti()

    ' Read value from cell
    cnSheet1.Range("A1:B10") = 66

End Sub
```

You cannot read from multiple cells to a variable. However you can read to an array which is a collection of variables. We will look at doing this in the next section.

How to Copy and Paste Cells

If you want to copy and paste a range of cells then you do not need to select them. This is a common error made by new VBA users.

You can simply copy a range of cells like this

```
Range("A1:B4").Copy Destination:=Range("C5")
```

Using this method copies everything – values, formats, formulas and so on. If you want to copy individual items you can use the **PasteSpecial** property of range.

It works like this

```
Range("A1:B4").Copy
Range("F3").PasteSpecial Paste:=xlPasteValues
Range("F3").PasteSpecial Paste:=xlPasteFormats
Range("F3").PasteSpecial Paste:=xlPasteFormulas
```

The following table shows a full list of all the paste types

Paste Type
xlPasteAll
xlPasteAllExceptBorders
xlPasteAllMergingConditionalFormats
xlPasteAllUsingSourceTheme
xlPasteColumnWidths
xlPasteComments
xlPasteFormats
xlPasteFormulas
xlPasteFormulasAndNumberFormats
xlPasteValidation
xlPasteValues
xlPasteValuesAndNumberFormats

Reading a Range of Cells to an Array

You can also copy values by assigning the value of one range to another.

```
Range("A3:Z3").Value = Range("A1:Z1").Value
```

The **value** of range in this example is considered to be a variant array. What this means is that you can easily read from a range of cells to an array. You can also write from an array to a range of cells. If you are not familiar with arrays you can check them out in this post: The Complete Guide to Arrays in Excel VBA (http://excelmacromastery.com/the-complete-guide-to-using-arrays-in-excel-vba/).

The following code shows an example of using an array with a range.

```
Public Sub ReadToArray()

    ' Create dynamic array
    Dim StudentMarks() As Variant

    ' Read 26 values into array from the first row
    StudentMarks = Range("A1:Z1").Value

    ' Do something with array here

    ' Write the 26 values to the third row
    Range("A3:Z3").Value = StudentMarks

End Sub
```

Keep in mind that the array created by the read is a 2 dimensional array. This is because a spreadsheet stores values in two dimensions i.e. rows and columns

Going through all the cells in a Range

Sometimes you may want to go through each cell one at a time to check value.

You can do this using a For Each (http://excelmacromastery.com/the-ultimate-guide-to-loops-in-excel-vba/#The_For_Each_Loop) loop shown in the following code

```
Public Sub TraversingCells()

    ' Go through each cells in the range
    Dim rg As Range
    For Each rg In cnSheet1.Range("A1:A10,A20")
        ' Print address of cells that are negative
        If rg.Value < 0 Then
            Debug.Print rg.Address + " is negative."
        End If
    Next

End Sub
```

You can also go through consecutive Cells using the Cells property and a standard For (http://excelmacromastery.com/the-ultimate-guide-to-loops-in-excel-vba/#The_For_Loop)loop.

The standard loop is more flexible about the order you use but it is slower than a For Each loop.

```
Public Sub TraverseCells()

    ' Go through cells from A1 to A10
    Dim i As Long
    For i = 1 To 10
        ' Print address of cells that are negative
        If Range("A" & i).Value < 0 Then
            Debug.Print Range("A" & i).Address + " is negative."
        End If
    Next

    ' Go through cells in reverse i.e. from A10 to A1
    For i = 10 To 1 Step -1
        ' Print address of cells that are negative
        If Range("A" & i) < 0 Then
            Debug.Print Range("A" & i).Address + " is negative."
        End If
    Next

End Sub
```

Formatting Cells

Sometimes you will need to format the cells the in spreadsheet. This is actually very straightforward. The following example shows you various formatting you can add to any range of cells

```
Public Sub FormattingCells()

    With cnSheet1

        ' Format the font
        .Range("A1").Font.Bold = True
        .Range("A1").Font.Underline = True
        .Range("A1").Font.Color = rgbNavy

        ' Set the number format to 2 decimal places
        .Range("B2").NumberFormat = "0.00"
        ' Set the number format to a date
        .Range("C2").NumberFormat = "dd/mm/yyyy"
        ' Set the number format to general
        .Range("C3").NumberFormat = "General"
        ' Set the number format to text
        .Range("C4").NumberFormat = "Text"

        ' Set the fill color of the cell
        .Range("B3").Interior.Color = rgbSandyBrown

        ' Format the borders
        .Range("B4").Borders.LineStyle = xlDash
        .Range("B4").Borders.Color = rgbBlueViolet

    End With

End Sub
```

Main Points

The following is a summary of the main points

1. **Range** returns a range of cells
2. **Cells** returns one cells only
3. You can **read** from one cell to another

4. You can **read** from a range of cells to another range of cells.
5. You can **read** values from cells to variables and vice versa.
6. You can **read** values from ranges to arrays and vice versa
7. You can use a **For Each** or **For** loop to run through every cell in a range.
8. The properties **Rows** and **Columns** allow you to access a range of cells of these types

What's Next?

The three most important elements of VBA are Workbooks (http://excelmacromastery.com/complete-guide-excel-vba-workbook/), Worksheets (http://excelmacromastery.com/the-complete-guide-to-worksheets-in-excel-vba/)and Cells and Ranges. If you feel you have a good understanding of these the you may want to check out The Ultimate Guide to Loops in Excel VBA (http://excelmacromastery.com/the-ultimate-guide-to-loops-in-excel-vba/). You can also view a list of all the VBA posts here (http://excelmacromastery.com/a-quick-guide-to-the-vba-posts/).

Free PDF of this Post

CLICK HERE TO DOWNLOAD THE FREE PDF VERSION OF THIS POST (https://excelmacromastery.leadpages.co/leadbox/146ccea73f72a2%3A106f252983

Note: I periodically archive comments to maintain the page speed.

Cells (https://excelmacromastery.com/tag/cells/) Format Cells (https://excelmacromastery.com/tag/format-cells/) Offset

(https://excelmacromastery.com/tag/offset/) Range
(https://excelmacromastery.com/tag/range/) VBA
(https://excelmacromastery.com/tag/vba/) Worksheets
(https://excelmacromastery.com/tag/worksheets/)

Previous
The Complete Guide To The VBA Worksheet (https://excelmacromastery.com/excel-vba-worksheet/)

Next
The Complete Guide to Using Arrays in Excel VBA
(https://excelmacromastery.com/excel-vba-array/)

91 COMMENTS

Adam (http://n/a)

May 23, 2017 at 8:06 pm (https://excelmacromastery.com/excel-vba-range-cells/comment-page-2/#comment-6910)

Hi Paul,
I am new at excel macros and I've stumbled on looping issue. I have a macro that does what I want but it requires certain guessing
on a user part and I don't like the guessing.
I have formulas in two columns D7 (=E6-$D$4) & E6 (=D6-(ABS(D6)^0.5)) starting row 6.
100, $D$4=01 and H2=20 are manual entries and H2 is guessed number of iterations.

D E

6 100 90
7 89.9 80.41843895
8 80.31843895 71.35638344
9 71.25638344 62.81503382
10 62.71503382 54.79575141
11 54.69575141 47.30009381
12 47.20009381 40.32986136

13 40.22986136 33.88715994
14 33.78715994 27.97448758
15 27.87448758 22.59485809
16 22.49485809 17.75198363
17 17.65198363 13.45055725
18 13.35055725 9.696715815
19 9.596715815 6.498859167
20 6.398859167 3.869262526
21 3.769262526 1.82780366
22 1.72780366 0.413344205
23 0.313344205 -0.24642736
24 -0.34642736 -0.935008162
25 -1.035008162 -2.05236167
26 -2.15236167 -3.619454602
27 -3.719454602 -5.648043361

I want to loop through the formulas until value in one column changes to negative and and then macro stops instead as running
predefined number of times to eliminate situation when guessed number of iterations will be too small and negative value currently in E23 won't be reached.
However when I eliminate H20 and I'm trying to use loop with condition to loop until first negative value in column E is reached, macro crushes.

My current code is as below and I'd really appreciate your comments, I can't find a suitable example online.

Best regards,

Adam

```
Sub AddRowFormula()
Dim n As Range, rng As Range, rng1 As Range
‘ # of iterations
Set n = Range(“H2”)
‘ cell where formulas start
Set rng = Range(“D7”)
rng.Select
line2:
Range(rng.Offset(1, 0), rng.Offset(3, 0)).EntireRow.Insert
Range(rng, rng.End(xlToRight)).Copy
```

```
Range(rng, rng.Offset(n, 0)).PasteSpecial
Set rng = rng.Offset(n + 1, 0)
If rng = “” Then
GoTo line1
Else
GoTo line2
End If
line1:
Application.CutCopyMode = False
Range(“D7”).Select
MsgBox “Macro completed”
End Sub
```

Reply (https://excelmacromastery.com/excel-vba-range-cells/?replytocom=6910#respond)

Paul Kelly

May 29, 2017 at 9:42 am (https://excelmacromastery.com/excel-vba-range-cells/comment-page-2/#comment-7293)

Hi Adam,

I am not 100% clear on what you are trying to achieve with your macro.

“I want to loop through the formulas until value in one column changes to negative and and then macro stops”

You haven’t said why you are copying and pasting and what you are trying to achieve.

Paul

Reply (https://excelmacromastery.com/excel-vba-range-cells/?replytocom=7293#respond)

Adam

May 29, 2017 at 6:01 pm (https://excelmacromastery.com/excel-vba-range-cells/comment-page-2/#comment-7298)

Hi Paul,
Thank you for replying and my apology for not being clear about my issue.
The goal is to run the calculations until a break point between positive and negative values in column E is reached, in this case:
E22 = 0.413344205
E23 = -0.24642736
I want the calculations to stop when the first negative number will be obtained – in this case E23.
Right now the macro I am using is working based on fixed number of calculations stated in H2 and equals 20 and the number of iterations is chosen arbitrarily, simply speaking just a guess. I want to eliminate the guess work and keep copying the formulas down until first negative value in column E will show up. I'm not concerned with the number of iterations, if it will take three iterations to get first negative or it will take few hundreds or even few thousands. As long as I'll get the break point I'll be happy:-)
I hope this makes sense but if not I'll try to make it clearer.
Best Regards,
Adam

Reply (https://excelmacromastery.com/excel-vba-range-cells/?replytocom=7298#respond)

Paul Kelly

May 30, 2017 at 12:51 pm (https://excelmacromastery.com/excel-vba-range-cells/comment-page-2/#comment-7310)

This should do the trick

```
Sub GetBreak()

    Dim bFound As Boolean
    bFound = False

    Dim row As Long
    row = 7

    Do
        Sheet1.Range("D" & row & ":E"
                    Sheet1.Range("D"

        ' Check for negative value
        If Sheet1.Range("D" & row + 1
            bFound = True
        End If

        ' Move to next row
        row = row + 1

    Loop Until bFound = True

End Sub
```

Reply (https://excelmacromastery.com/excel-vba-range-cells/?replytocom=7310#respond)

Adam

June 5, 2017 at 12:27 pm (https://excelmacromastery.com/excel-vba-range-cells/comment-page-2/#comment-7502)

Thank you very much Paul.
I works perfectly the way I wanted:-)
It is also a great example

because it only works in specific range without inserting extra rows which could mess with other parts of the sheet.

Best Regards,

Adam

Nick

May 31, 2017 at 1:34 pm (https://excelmacromastery.com/excel-vba-range-cells/comment-page-2/#comment-7355)

HI Paul,

Great tutorials, thanks!!

When I try and run:

“Public Sub WriteToFirstBlankCell()

‘ Get last column from left that is not blank
Dim lLastCol As Integer
lLastCol = cnSheet1.Range(“A1”).End(xlToRight).Column

‘ Write text to first blank cell in Row 1
cnSheet1.Cells(1, lLastCol + 1) = “John Smith”

End Sub”

I get two problems, one my excel doesn’t like the number 1 as the start of a variable.
the other is I get a ” run-time error 1004: Application-defined or object-defined error”
when I try to execute the code

Reply (https://excelmacromastery.com/excel-vba-range-cells/?replytocom=7355#respond)

Nick

May 31, 2017 at 3:15 pm (https://excelmacromastery.com/excel-vba-range-cells/comment-page-2/#comment-7359)

Update: changing “Range(“A1”) to Cells(1,1) did the trick!

```
Sub WriteToFirstBlankCell()

Dim OneLastCol As Integer
'OneLastCol = cnSheet1.Range("A1").End(xlToRight).Column
OneLastCol = cnSheet1.Cells(1, 1).End(xlToRight).Column

cnSheet1.Cells(1, OneLastCol + 1) = "John Smith"
' cnSheet1.Cells(1, OneastCol + 1) = "John Smith"

End Sub
```

Kind Regards,
Nick

Reply (https://excelmacromastery.com/excel-vba-range-cells/?replytocom=7359#respond)

Paul Kelly

May 31, 2017 at 3:42 pm (https://excelmacromastery.com/excel-vba-range-cells/comment-page-2/#comment-7361)

Hi Nick,

It is the letter l in front of LastCol rather than the number one:-)

Range("A1") will work fine. *Cells(1,1)* is essentially the same thing.

If you only have text in cell A1 in the first row then this error occurs.

The following line should prevent this error(add it before the last line with "John Smith"

```
lLastCol = IIf(lLastCol = cnSheet1.Columns.Count, 1, lLastCol)
```

Regards
Paul

Reply (https://excelmacromastery.com/excel-vba-range-cells/?replytocom=7361#respond)

Nick

June 1, 2017 at 3:22 pm (https://excelmacromastery.com/excel-vba-range-cells/comment-page-2/#comment-7388)

Great, Thanks Paul 🙂

It seems the Cells(1,1) reference works after a fashion with a blank worksheet – it just sticks "John Smith" right at the end of the sheet

Kind Regards,

Nick

Reply (https://excelmacromastery.com/excel-vba-range-cells/?replytocom=7388#respond)

Nick

June 1, 2017 at 2:58 pm (https://excelmacromastery.com/excel-vba-range-cells/comment-page-2/#comment-7386)

Hi Paul,

In the example "Reading Values from one Cell to another" you have the example:
"' Will not work – You cannot read from multiple cells
cnSheet1.Range("A1:A5") = cnSheet1.Range("B1:B5")"

If you add .value to the end of the code you can then duplicate the range B1:B5
I'm not sure why this seems to make the difference – do you mind explaining?

Kind Regards,

Nick

Reply (https://excelmacromastery.com/excel-vba-range-cells/?replytocom=7386#respond)

Paul Kelly

June 1, 2017 at 3:14 pm (https://excelmacromastery.com/excel-vba-range-cells/comment-page-2/#comment-7387)

Hi Nick,

I changed that example slightly to use Value. Before otherwise it is a bit misleading.

Value for a single cell is a single value.
Value for multiple cells is a 2D array.

You cannot use the default with multiple – only with single. I'm not sure why this is the case.

Regards
Paul

Reply (https://excelmacromastery.com/excel-vba-range-cells/?replytocom=7387#respond)

Nick

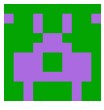
June 1, 2017 at 3:28 pm (https://excelmacromastery.com/excel-vba-range-cells/comment-page-2/#comment-7389)

Hi Paul,
Thanks – great instructions, I've learned a whole load about VBA from your site already!

Regards,

Nick

Reply (https://excelmacromastery.com/excel-vba-range-cells/?replytocom=7389#respond)



Josalyn Raymundo

June 8, 2017 at 11:47 am (https://excelmacromastery.com/excel-vba-range-cells/comment-page-2/#comment-7678)

Hi Paul,

I need your help. Here's my problem:
A1 Contains a drop-down list. B1 will populate (via drop down list also) based on selected item from A1.

Say if select "Tissue" from A1, then I can only see different brand of tissues from B1
If I select "Fruit" from A1, then I can only see fruit selections from B1
I will not be allowed to fill-out B1 unless I select something from A1.

My dilemma now is that If both cells are already filled-out, I can still go back to A1 and change it. It will accept the change made and will not prompt an error even if B1 was not changed. In effect the data in A1 and B1 are now misaligned. Can you help me with a code where it will prevent any changes in A1 if the data in B1 do not corresponds with A1?

Thanks and appreciate any help you can provide.

Reply (https://excelmacromastery.com/excel-vba-range-cells/?replytocom=7678#respond)

Paul Kelly

June 10, 2017 at 10:03 am (https://excelmacromastery.com/excel-vba-range-cells/comment-page-2/#comment-7738)

Hi Josalyn

Can you post a snippet of your code?

Reply (https://excelmacromastery.com/excel-vba-range-cells/?replytocom=7738#respond)

Angus Arrol

June 30, 2017 at 8:28 pm (https://excelmacromastery.com/excel-vba-range-cells/comment-page-2/#comment-8620)

Hello Paul,

Compliments on such a clearly-written tutorial! Almost as impressive is the extensive comments attached to it! With apologies, I haven't read through all of them to see if anyone else has the same question.

My question is whether Range().Offset starts offsets at zero or one. In the section "The Offset Property of Range", the DayOffSet() example function seems to take lDay=1 as meaning no offset, lDay=2 as an offset of one, and so on. However the following UsingOffset() example

seems to take 1 as an offset of one. Am I missing something?

In any case, thanks for such a well-written tutorial!
Cheers, Angus

Reply (https://excelmacromastery.com/excel-vba-range-cells/?replytocom=8620#respond)

Paul Kelly

July 1, 2017 at 6:24 am (https://excelmacromastery.com/excel-vba-range-cells/comment-page-2/#comment-8642)

Offset starts at 1.
For column 0 is not offset, -1 is one column to the left, 1 is one column to the right.

Reply (https://excelmacromastery.com/excel-vba-range-cells/?replytocom=8642#respond)

Angus Arrol

July 2, 2017 at 8:50 am (https://excelmacromastery.com/excel-vba-range-cells/comment-page-2/#comment-8711)

Hello Paul,

I am comparing the SetValueSelect() and DayOffSet() subroutines. When lDay = 1, SetValueSelect() executes

Case 1: cnSheet1.Range(“G3”) = lValue

which puts lValue in cell G3. But when DayOffSet() executes

cnSheet1.Range(“G3”).Offset(, lDay) = lValue

doesn’t that put lValue in cell H3?

Cheers, Angus

Reply (https://excelmacromastery.com/excel-vba-range-cells/?replytocom=8711#respond)

Paul Kelly

July 4, 2017 at 2:55 am (https://excelmacromastery.com/excel-vba-range-cells/comment-page-2/#comment-8788)

Hi Angus,

I see what you mean. The *SetValueSelect* sub should indeed start at H3.

I have updated the example.

Thanks for pointing that out.

Paul

Reply (https://excelmacromastery.com/excel-vba-range-cells/?replytocom=8788#respond)

Anshuman

July 19, 2017 at 7:00 am (https://excelmacromastery.com/excel-vba-range-cells/comment-page-2/#comment-9395)

Hello Paul,

I am having some trouble in understand the cell traverse pattern of FOR loop.
The code is {i,j,k as integer: g1b double}
For k = 1 To 7
For j = 1 To 7
For j = 1 To 6
ThisWorkbook.Sheets(“Consumption”).Cells(11 * i + k – 4, j +2).Value

Next i
ThisWorkbook.Sheets(“Emissions Calculation”).Cells(8 + k, j + 13).Value = g1b
Next j
Next k
end sub

I understand that “i” will go 11 multiples in the sheet, but i did not understand [+ k – 4] part.
Glad if you could help me

Reply (https://excelmacromastery.com/excel-vba-range-cells/?replytocom=9395#respond)

Paul Kelly

July 21, 2017 at 4:25 am (https://excelmacromastery.com/excel-vba-range-cells/comment-page-2/#comment-9503)

Hi Anshuman

The third For loop should be *For i* instead of *For j*.

The loops work like this
K=1,j=1,i=1
k=1,j=1,i=2
k=1,j=1,i=3

i = 6 then
k=1,j=2,i=1
k=1,j=2,i=2

and so on until
k=7,j=7,i=6

Reply (https://excelmacromastery.com/excel-vba-range-cells/?replytocom=9503#respond)

randy

July 24, 2017 at 7:55 pm (https://excelmacromastery.com/excel-vba-range-cells/comment-page-2/#comment-9657)

I have a combo box that contains a list of worksheet names. the user will select the appropriate sheet in the combo box and then when another button is selected I want it to display a listing of all information in column “A” of the previously selected sheet

Reply (https://excelmacromastery.com/excel-vba-range-cells/?replytocom=9657#respond)

Paul Kelly

July 26, 2017 at 7:15 am (https://excelmacromastery.com/excel-vba-range-cells/comment-page-2/#comment-9696)

Hi Randy,

You can do it like this

```
' Get the worksheet from the name in the combobox
Dim sh As Worksheet
Set sh = ThisWorkbook.Worksheets(Trim(Sheet1.ComboBox1.Value))

' Get the last row
Dim lastrow As Long
lastrow = sh.Cells(sh.Rows.Count, 1).End(xlUp).Row

' read through the cells
Dim i As Long
For i = 1 To lastrow
    ' Print the value of all the cells
    Debug.Print sh.Range("A" & i)
Next i
```

Reply (https://excelmacromastery.com/excel-vba-range-cells/?replytocom=9696#respond)

Justin

August 3, 2017 at 6:28 am (https://excelmacromastery.com/excel-vba-range-cells/comment-page-2/#comment-13038)

Hi Paul,

I am trying to do a time tracker and I have come up with the simplest vba along with an inserted start/stop button. I was able to do starttimestamp when I press the button on A1 then the active cell will now go to B1 then hit the button for endtime stamp again. It works fine however what I wanted to do next was to put the active cell after B1 this time on A2 for start time then B2 for end, and then A3 then B3 and so on. Could you please edit and maybe give a little hint on how to do this. Thank you so much Paul!

Sub EnterCurrentTime()
‘

‘ EnterCurrentTime Macro
‘ This macro enters the current time into the active cell

```
ActiveCell.Value = Now()
ActiveCell.Offset(0, 1).Select
End Sub
```

Reply (https://excelmacromastery.com/excel-vba-range-cells/?replytocom=13038#respond)

Stefano

August 28, 2017 at 3:34 pm (https://excelmacromastery.com/excel-vba-range-cells/comment-page-2/#comment-15266)

Hi Paul ad thank you very much for the great info.
How do I select a range of cells (say a table, for example with a fixed number of columns and a variable number of rows), making sure that my selection captures every time all the non empty rows?
In other words, how do I specify the parameters of a range by using variables and not constants (such as startingcell:endingcell instead of “B1:C3”)?
Kindly let me know.
Best Regards

Reply (https://excelmacromastery.com/excel-vba-range-cells/?replytocom=15266#respond)

Hilton

September 6, 2017 at 7:30 am (https://excelmacromastery.com/excel-vba-range-cells/comment-page-2/#comment-16080)

Hi Paul,

I earnestly need your help.
I have a source file holding ongong projects names in column B with their respective status in adjacent rows. Hence each project has its status adjacent to it in the same row but under columns C,D,E,F etc. Also their is possibility of new projects to be added to the source in the same structure above.
I need a separate automated workbook called ” QUICK report “to automatically extract these Project with their statuses into specific cells and always check for newly added project/s to likewise extract into specific Cells. The extraction should stop at any blank row.

Reply (https://excelmacromastery.com/excel-vba-range-cells/?replytocom=16080#respond)

Paul Kelly

September 7, 2017 at 4:11 am (https://excelmacromastery.com/excel-vba-range-cells/comment-page-2/#comment-16110)

Hi Hilton,

What code has your written so far?

Reply (https://excelmacromastery.com/excel-vba-range-cells/?replytocom=16110#respond)

← Older Comments (https://excelmacromastery.com/excel-vba-range-cells/comment-page-1/#comments)

LEAVE A REPLY

Your email address will not be published. Required fields are marked *

Name *

Email *

Website

Post Comment