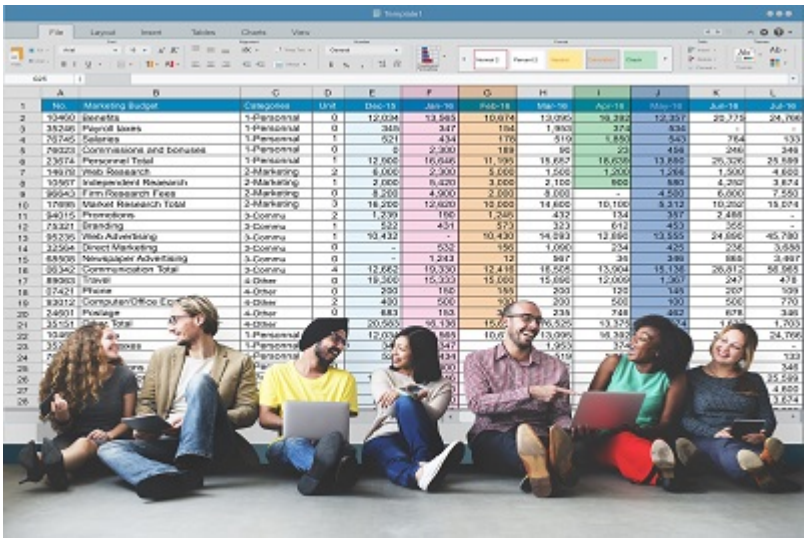
Excel Macro Mastery (https://excelmacromastery.com/)

THE MISSING VBA HANDBOOK

The Complete Guide To The VBA Worksheet

BY PAUL KELLY (HTTPS://EXCELMACROMASTERY.COM/AUTHOR/ADMIN/) · 45 COMMENTS (HTTPS://EXCELMACROMASTERY.COM/EXCEL-VBA-WORKSHEET/#COMMENTS)



(/#facebook) (/#twitter) (/#pinterest) (/#linkedin) (/#reddit)

(https://www.addtoany.com/share#url=https%3A%
vba-worksheet%2F&title=The%20Complete%20Gu

“The visionary starts with a clean sheet of paper, and re-imagines the world” – Malcolm Gladwell

This post provides a complete guide to using the Excel **VBA Worksheet** in Excel VBA. If you want to know how to do something quickly then check out the quick guide to the VBA

Worksheet below.

If you are new to VBA then this post is a great place to start. I like to break things down into simple terms and explain them in plain English without the jargon.

You can **read through the post from start to finish** as it is written in a logical order. If you prefer, you can use the table of contents below and go directly to the topic of your choice.

Contents [hide]

A Quick Guide to the VBA Worksheet

The following table gives a quick run down to the different worksheet methods.

Note: I use *Worksheets* in the table below without specifying the workbook i.e. *Worksheets* rather than *ThisWorkbook.Worksheets*, *wk.Worksheets* etc. This is to make the examples clear and easy to read. You should always specify the workbook when using *Worksheets*. Otherwise

the active workbook will be used by default.

Task	How to
Access worksheet by name	Worksheets("Sheet1")
Access worksheet by position from left	Worksheets(2) Worksheets(4)
Access the left most worksheet	Worksheets(1)
Access the right most worksheet	Worksheets(Worksheets.Count)
Access using worksheet code name(current workbook only)	see Code Name section below
Access using worksheet code name(other workbook)	see Code Name section below
Access the active worksheet	ActiveSheet
Declare worksheet variable	**Dim** sh **As** Worksheet
Assign worksheet variable	**Set** sh = Worksheets("Sheet1")
Add worksheet	Worksheets.Add
Add worksheet and assign to variable	**Set** sh =Worksheets.Add
Add worksheet to first position(left)	Worksheets.Add Before:=Worksheets(1)
Add worksheet to last position(right)	Worksheets.Add after:=Worksheets(Worksheets.Count)
Add multiple worksheets	Worksheets.Add Count:=3
Activate Worksheet	sh.Activate
Copy Worksheet	sh.Copy
Copy after a worksheet	sh1.Copy After:=Sh2

Task	How to
Copy before a worksheet	sh1.Copy Before:=Sh2
Delete Worksheet	sh.Delete
Delete Worksheet without warning	Application.DisplayAlerts = **False** sh.Delete Application.DisplayAlerts = **True**
Change worksheet name	sh.Name = "Data"
Show/hide worksheet	sh.Visible = xlSheetHidden sh.Visible = xlSheetVisible
Loop through all worksheets(For)	**Dim** i **As Long** **For** i = 1 **To** Worksheets.Count **Debug.Print** Worksheets(i).Name **Next** i
Loop through all worksheets(For Each)	**Dim** sh **As** Worksheet **For Each** sh **In** Worksheets **Debug.Print** sh.Name **Next**

Introduction

The three most important elements of VBA are the Workbook (https://excelmacromastery.com/excel-vba-workbook/), the Worksheet and Cells (https://excelmacromastery.com/excel-vba-range-cells). Of all the code your write, 90% will involve one or all of them.

The most **common use of the worksheet** in VBA is for accessing its cells. You may use it to protect, hide, add, move or copy a worksheet. However, you will mainly use it to perform some action on one or more cells on the worksheet.

Using Worksheets is more straightforward than using workbooks. With workbooks you may need to open them, find which folder they are in, check if they are in use and so on. With a worksheet, it either exists in the workbook or it doesn't.

Accessing the Worksheet

In VBA, each workbook has a collection of worksheets. There is an entry in this collection for each worksheet in the workbook. This collection is simply called **Worksheets** and is used in a very similar way to the **Workbooks** collection. To get access to a worksheet all you have to do is supply the name.

The code below writes "Hello World" in Cell A1 of Sheet1, Sheet2 and Sheet3 of the current workbook.

```
' https://excelmacromastery.com/
Public Sub WriteToCell1()

    ' Write To cell A1 In Sheet1,Sheet2 And Sheet3
    ThisWorkbook.Worksheets("Sheet1").Range("A1") = "Hello World"
    ThisWorkbook.Worksheets("Sheet2").Range("A1") = "Hello World"
    ThisWorkbook.Worksheets("Sheet3").Range("A1") = "Hello World"

End Sub
```

The **Worksheets** collection is always belong to a workbook. If we don't specify the workbook then the active workbook is used by default.

```
' https://excelmacromastery.com/
Public Sub WriteToCell1()

    ' Worksheets refers to the worksheets in the active workbook
    Worksheets("Sheet1").Range("A1") = "Hello World"
    Worksheets("Sheet2").Range("A1") = "Hello World"
    Worksheets("Sheet3").Range("A1") = "Hello World"

End Sub
```

Hide Worksheet

The following examples show how to hide and unhide a worksheet

```
ThisWorkbook.Worksheets("Sheet1").Visible = xlSheetHidden

ThisWorkbook.Worksheets("Sheet1").Visible = xlSheetVisible
```

If you want to prevent a user accessing the worksheet, you can make it “very hidden”. This means it can only be made visible by the code.

```
' Hide from user access
ThisWorkbook.Worksheets("Sheet1").Visible = xlVeryHidden

' This is the only way to make a xlVeryHidden sheet visible
ThisWorkbook.Worksheets("Sheet1").Visible = xlSheetVisible
```

Protect Worksheet

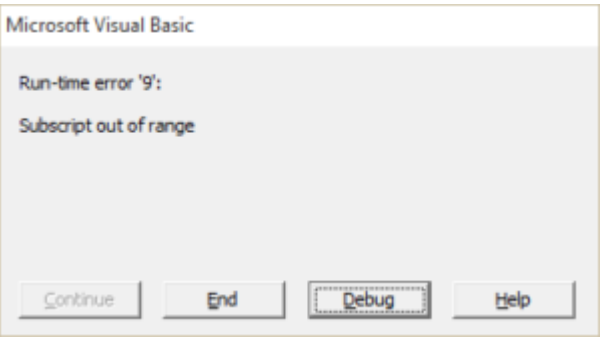
Another example of using the worksheet is when you want to protect it

```
ThisWorkbook.Worksheets("Sheet1").Protect Password:="MyPass"

ThisWorkbook.Worksheets("Sheet1").Unprotect Password:="MyPass"
```

Subscript Out of Range

When you use *Worksheets* you may get the error:

Run-time Error 9 Subscript out of Range



This means you tried to access a worksheet that doesn't exist. This may happen for the following reasons

1. The worksheet name given to *Worksheets* is spelled incorrectly.
2. The name of the worksheet has changed.
3. The worksheet was deleted.
4. The index was to large e.g. You used *Worksheets(5)* but there are only four worksheets
5. The wrong workbook is being used e.g. *Workbooks(**"book1.xlsx"**).Worksheets("Sheet1")* instead of *Workbooks(**"book3.xlsx"**).Worksheets("Sheet1")*.

If you still have issues then use one of the loops from Loop Through The Worksheets section to print the names of all worksheets the collection.

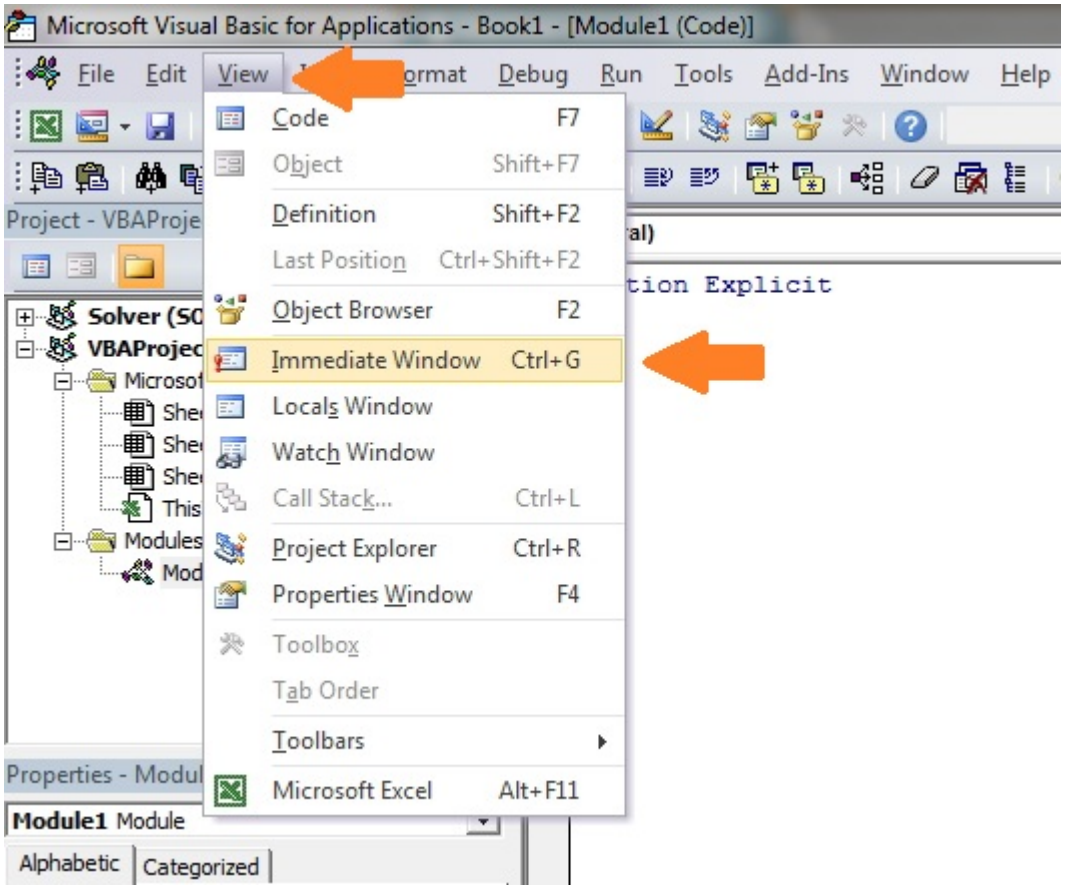
Using the Index to Access the Worksheet

So far we have been using the sheet name to access the sheet. The index refers to the sheet tab position in the workbook. As the position can easily be changed by the user it is not a good idea to use this.

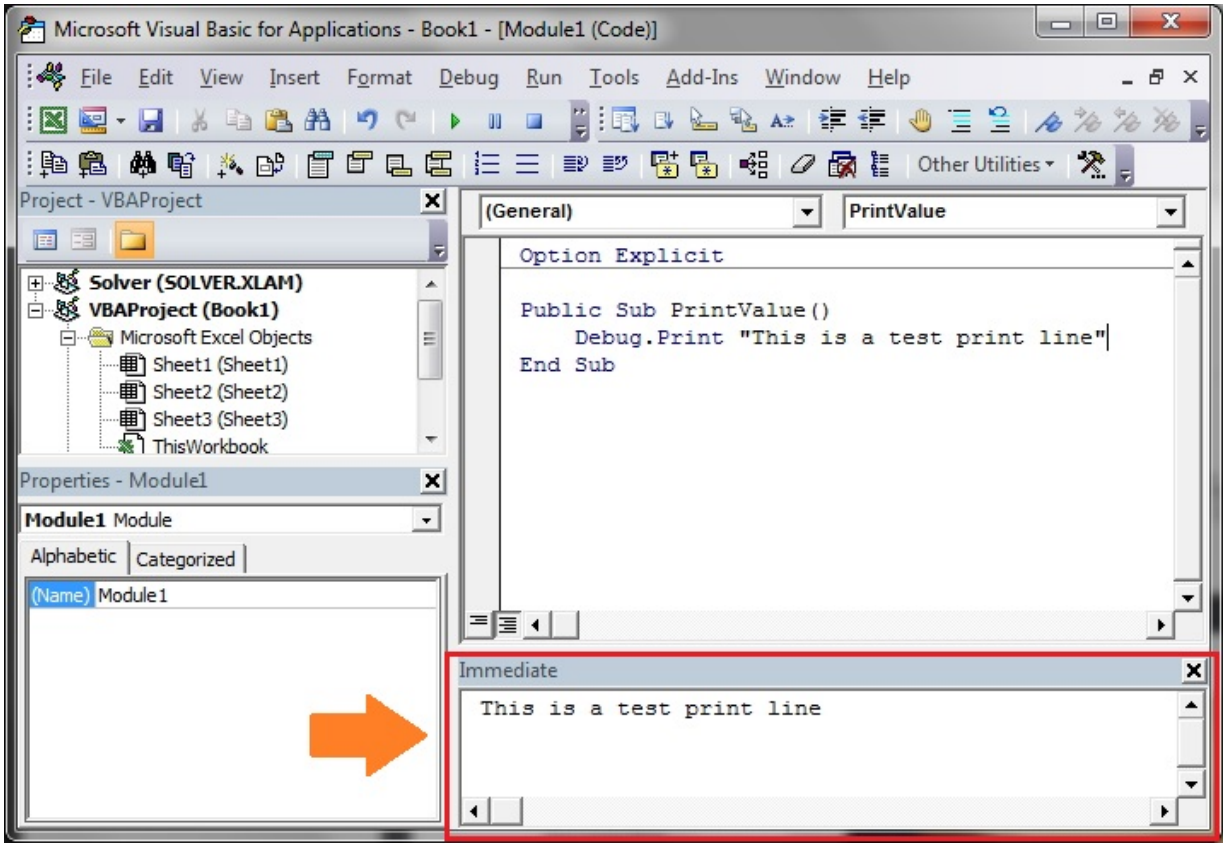
The following code shows examples of using the index

```
' https://excelmacromastery.com/
' Using this code is a bad idea as
' sheet positions changes all the time
Public Sub UseSheetIdx()

    With ThisWorkbook
        ' Left most sheet
        Debug.Print .Worksheets(1).Name
        ' The third sheet from the left
        Debug.Print .Worksheets(3).Name
        ' Right most sheet
        Debug.Print .Worksheets(.Worksheets.Count).Name
    End With

End Sub
```

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



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



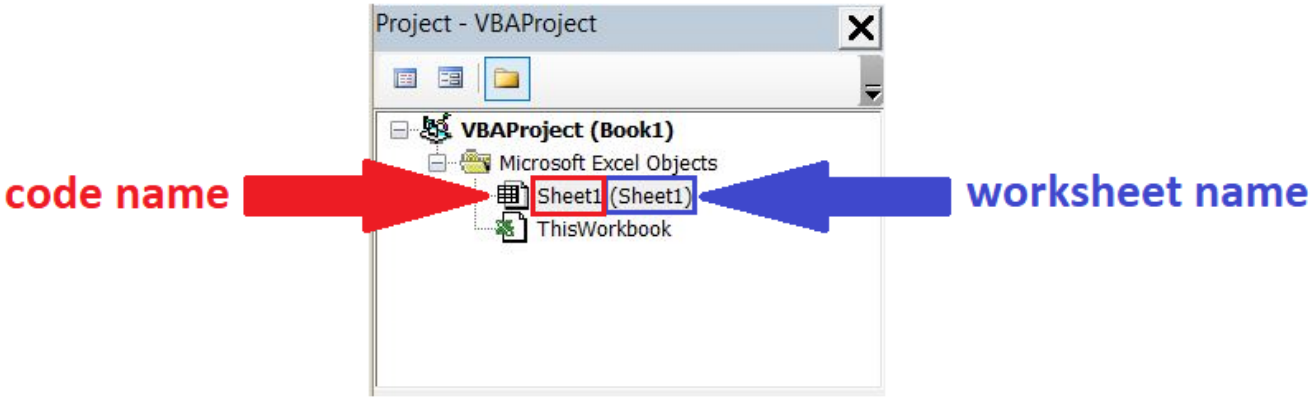
(https://excelmacromastery.com/wp-content/uploads/2014/12/ImmediateSampeText.jpg)

Using the Code Name of a Worksheet

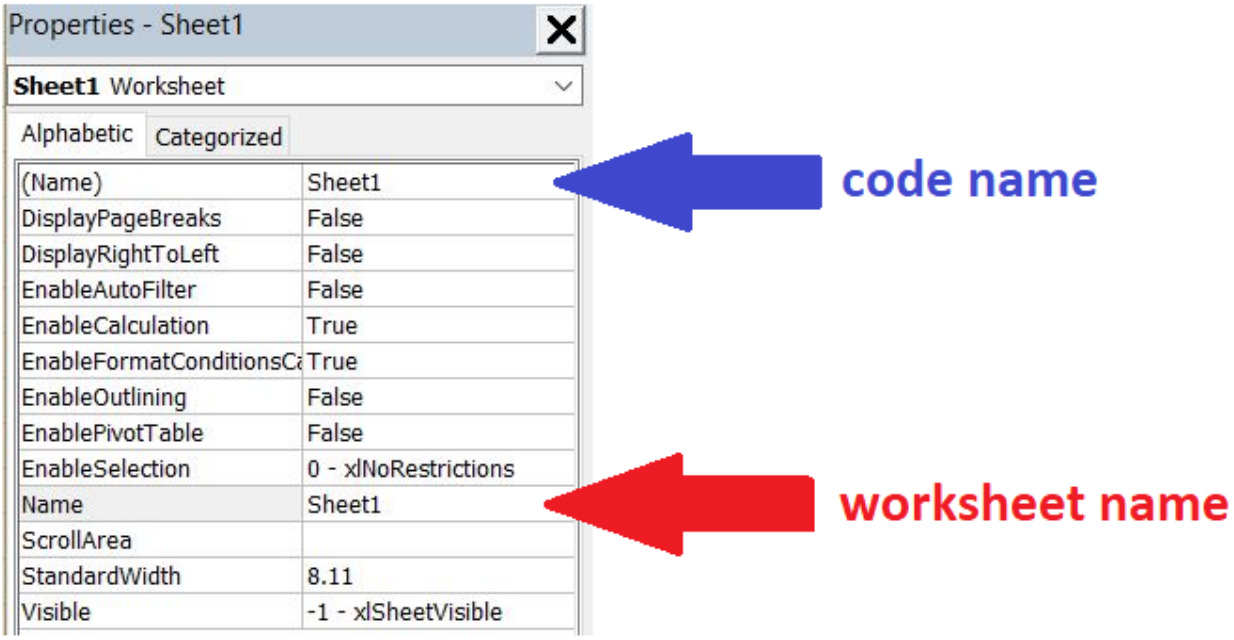
The best method of accessing the worksheet is using the code name. Each worksheet has a sheet name and a code name. The sheet name is the name that appears in the worksheet tab in Excel.

Changing the sheet name does not change the code name meaning that referencing a sheet by the code name is a good idea.

If you look in the VBE property window you will see both names. In the image you can see that the code name is the name outside the parenthesis and the sheet name is in the parenthesis.



You can change both the sheet name and the code name in the property window of the sheet(see image below).



If your code refers to the code name then the user can change the name of the sheet and it will not affect your code. In the example below we reference the worksheet directly using the code name.

```
' https://excelmacromastery.com/
Public Sub UseCodeName2()

    ' Using the code name of the worksheet
    Debug.Print CodeName.Name
    CodeName.Range("A1") = 45
    CodeName.Visible = True

End Sub
```

This makes the code easy to read and safe from the user changing the sheet name.

Code Name in other Workbooks

There is one drawback to using the code name. It can only refer to worksheets in the workbook that contains the code i.e. ThisWorkbook.

However, we can use a simple function to find the code name of a worksheet in a different workbook.

```
' https://excelmacromastery.com/
Public Sub UseSheet()

    Dim sh As Worksheet
    ' Get the worksheet using the codename
    Set sh = SheetFromCodeName("CodeName", ThisWorkbook)
    ' Use the worksheet
    Debug.Print sh.Name

End Sub

' This function gets the worksheet object from the Code Name
Public Function SheetFromCodeName(Name As String, bk As Workbook) As Worksheet

    Dim sh As Worksheet
    For Each sh In bk.Worksheets
        If sh.CodeName = Name Then
           Set SheetFromCodeName = sh
           Exit For
        End If
    Next sh

End Function
```

Using the above code means that if the user changes the name of the worksheet then your code will not be affected.

There is another way of getting the sheet name of an external workbook using the code name. You can use the **VBProject** element of that Workbook.

You can see how to do this in the example below. I have included this for completeness only and I would recommend using the method in the previous example rather than this one.

```
' https://excelmacromastery.com/
Public Function SheetFromCodeName2(codeName As String _
                            , bk As Workbook) As Worksheet

    ' Get the sheet name from the CodeName using the VBProject
    Dim sheetName As String
    sheetName = bk.VBProject.VBComponents(codeName).Properties("Name")

    ' Use the sheet name to get the worksheet object
    Set SheetFromCodeName2 = bk.Worksheets(sheetName)

End Function
```

Code Name Summary

The following is a quick summary of using the Code Name

1. The code name of the worksheet can be used directly in the code e.g. *Sheet1.Range*
2. The code name will still work if the worksheet name is changed.
3. The code name can only be used for worksheets in the same workbook as the code.
4. Anywhere you see *ThisWorkbook.Worksheets("sheetname")* you can replace it with the code name of the worksheet.
5. You can use the *SheetFromCodeName* function from above to get the code name of worksheets in other workbooks.

The Active Sheet

The **ActiveSheet** object refers to the worksheet that is currently active. You should only use ActiveSheet if you have a specific need to refer to the worksheet that is active.

Otherwise you should specify the worksheet you are using.

If you use a worksheet method like **Range** and don't mention the worksheet, it will use the active worksheet by default.

```
' Write to Cell A1 in the active sheet
ActiveSheet.Range("A1") = 99

' Active sheet is the default if no sheet used
Range("A1") = 99
```

Declaring a Worksheet Object

Declaring a worksheet object is useful for making your code neater and easier to read.

The next example shows code for updating ranges of cells. The first Sub does not declare a worksheet object. The second sub declares a worksheet object and the code is therefore much clearer.

```
' https://excelmacromastery.com/
Public Sub SetRangeVals()

    Debug.Print ThisWorkbook.Worksheets("Sheet1").Name
    ThisWorkbook.Worksheets("Sheet1").Range("A1") = 6
    ThisWorkbook.Worksheets("Sheet1").Range("B2:B9").Font.Italic = True
    ThisWorkbook.Worksheets("Sheet1").Range("B2:B9").Interior.Color = rgbRed

End Sub
```

```
' https://excelmacromastery.com/
Public Sub SetRangeValsObj()

    Dim sht As Worksheet
    Set sht = ThisWorkbook.Worksheets("Sheet1")

    sht.Range("A1") = 6
    sht.Range("B2:B9").Font.Italic = True
    sht.Range("B2:B9").Interior.Color = rgbRed

End Sub
```

You could also use the With keyword with the worksheet object as the next example shows.

```
' https://excelmacromastery.com/
Public Sub SetRangeValsObjWith()

    Dim sht As Worksheet
    Set sht = ThisWorkbook.Worksheets("Sheet1")

    With sht
        .Range("A1") = 6
        .Range("B2:B9").Font.Italic = True
        .Range("B2:B9").Interior.Color = rgbRed
    End With

End Sub
```

Accessing the Worksheet in a Nutshell

With all the different ways to access a worksheet, you may be feeling overwhelmed or confused. So in this section, I am going to break it down into simple terms

1. If you want to use whichever worksheet is currently active then use ActiveSheet.

```
ActiveSheet.Range("A1") = 55
```

2. If the worksheet is in the same workbook as the code then use the Code Name.

```
Sheet1.Range("A1") = 55
```

3. If the worksheet is in a different workbook then first get workbook and then get the worksheet.

```
' Get workbook
Dim wk As Workbook
Set wk = Workbooks.Open("C:\Docs\Accounts.xlsx", ReadOnly:=True)

' Then get worksheet
Dim sh As Worksheet
Set sh = wk.Worksheets("Sheet1")
```

If you want to protect against the user changing the sheet name then use the **SheetFromCodeName** function from the Code Name section.

```
' Get workbook
Dim wk As Workbook
Set wk = Workbooks.Open("C:\Docs\Accounts.xlsx", ReadOnly:=True)

' Then get worksheet
Dim sh As Worksheet
Set sh = SheetFromCodeName("sheetcodename",wk)
```

Add Worksheet

The examples in this section show you how to add a new worksheet to a workbook. If you do not supply any arguments to the **Add** function then the new worksheet will be placed before the active worksheet.

When you add a Worksheet, it is created with a default name like “Sheet4”. If you want to change the name then you can easily do this using the **Name** property.

The following example adds a new worksheet and changes the name to “Accounts”. If a worksheet with the name “Accounts” already exists then you will get an error.

```
' https://excelmacromastery.com/
Public Sub AddSheet()

    Dim sht As Worksheet

    ' Adds new sheet before active sheet
    Set sht = ThisWorkbook.Worksheets.Add
    ' Set the name of sheet
    sht.Name = "Accounts"

    ' Adds 3 new sheets before active sheet
    ThisWorkbook.Worksheets.Add Count:=3

End Sub
```

In the previous example, you are adding worksheets in relation to the active worksheet. You can also specify the exact position to place the worksheet.

To do this you need to specify which worksheet the new one should be inserted before or after.

The following code shows you how to do this.

```
' https://excelmacromastery.com/
Public Sub AddSheetFirstLast()

    Dim shtNew As Worksheet
    Dim shtFirst As Worksheet, shtLast As Worksheet

    With ThisWorkbook

        Set shtFirst = .Worksheets(1)
        Set shtLast = .Worksheets(.Worksheets.Count)

        ' Adds new sheet to first position in the workbook
        Set shtNew = Worksheets.Add(Before:=shtFirst)
        shtNew.Name = "FirstSheet"

        ' Adds new sheet to last position in the workbook
        Set shtNew = Worksheets.Add(After:=shtLast)
        shtNew.Name = "LastSheet"

    End With

End Sub
```

Delete Worksheet

To delete a worksheet you simply call the **Delete** member.

```
Dim sh As Worksheet
Set sh = ThisWorkbook.Worksheets("Sheet12")
sh.Delete
```


Excel will display a warning message when you delete a worksheet. If you want to hide this

message you can use the code below

```
Application.DisplayAlerts = False
sh.Delete
Application.DisplayAlerts = True
```

There are two issues to watch out for when it comes to deleting worksheets.

If you try to access the worksheet after deleting it you will get the “Subscript out of Range” error we saw in the Accessing the Worksheet section.

```
Dim sh As Worksheet
Set sh = ThisWorkbook.Worksheets("Sheet2")
sh.Delete

' This line will give 'Subscript out of Range' as "Sheet2" does not exist
Set sh = ThisWorkbook.Worksheets("Sheet2")
```

The second issue is when you assign a worksheet variable. If you try to use this variable after the worksheet is deleted then you will get an Automation error like this

Run-Time error -21147221080 (800401a8') Automation Error

If you are using the Code Name of the worksheet rather than a variable, then this will cause Excel to crash rather than give the automation error.

The following example shows how an automation errors occurs

```
sh.Delete

' This line will give Automation error
Debug.Assert sh.Name
```

If you assign the Worksheet variable to a valid worksheet it will work fine

```
sh.Delete

' Assign sh to another worksheet
Set sh = Worksheets("sheet3")

' This line will work fine
Debug.Assert sh.Name
```

Loop Through the Worksheets

The Worksheets member of Workbooks is a collection of worksheets belonging to a workbook. You can go through each sheet in the worksheets collection using a For Each (https://excelmacromastery.com/vba-for-loop/#The_VBA_For_Each_Loop) Loop or a For (https://excelmacromastery.com/vba-for-loop/#The_VBA_For_Loop) Loop.

The following example uses a **For Each** loop.

```
' https://excelmacromastery.com/
Public Sub LoopForEach()

    ' Writes "Hello World" into cell A1 for each worksheet
    Dim sht As Worksheet
    For Each sht In ThisWorkbook.Worksheets
         sht.Range("A1") = "Hello World"
    Next sht

End Sub
```

The next example uses the standard **For** loop

```
' https://excelmacromastery.com/
Public Sub LoopFor()

    ' Writes "Hello World" into cell A1 for each worksheet
    Dim i As Long
    For i = 1 To ThisWorkbook.Worksheets.Count
         ThisWorkbook.Worksheets(i).Range("A1") = "Hello World"
    Next sht

End Sub
```

You have seen how to access all open workbooks and how to access all worksheets in ThisWorkbook. Lets take it one step further. Lets access all worksheets in all open workbooks.

Note: If you use code like this to write to worksheets then back everything up first as you could end up writing the incorrect data to all the sheets.

```
' https://excelmacromastery.com/
Public Sub AllSheetNames()

    ' Prints the workbook and sheet names for
    ' all sheets in open workbooks
    Dim wrk As Workbook
    Dim sht As Worksheet
    For Each wrk In Workbooks
        For Each sht In wrk.Worksheets
            Debug.Print wrk.Name + ":" + sht.Name
        Next sht
    Next wrk

End Sub
```

Using the Sheets Collection

The workbook has another collection similar to Worksheets called **Sheets**. This causes confusion at times among users. To explain this first you need to know about a sheet type that is a chart.

It is possible in Excel to have a sheet that is a chart. To do this

1. Create a chart on any sheet.
2. Right click on the chart and select Move.
3. Select the first option which is “New Sheet” and click Ok.

Now you have a workbook with sheets of type worksheet and one of type chart.

- The **Worksheets** collection refers to all worksheets in a workbook. It does not include sheets of type chart.
- The **Sheets** collection refers to all sheets belonging to a workbook including sheets of type chart.

There are two code examples below. The first goes through all the Sheets in a workbook and prints the name of the sheet and type of sheet it is. The second example does the same with the Worksheets collection.

To try out these examples you should add a Chart sheet to your workbook first so you will see the difference.

```
' https://excelmacromastery.com/
Public Sub CollSheets()

    Dim sht As Variant
    ' Display the name and type of each sheet
    For Each sht In ThisWorkbook.Sheets
        Debug.Print sht.Name & " is type " & TypeName(sht)
    Next sht

End Sub

Public Sub CollWorkSheets()

    Dim sht As Variant
    ' Display the name and type of each sheet
    For Each sht In ThisWorkbook.Worksheets
        Debug.Print sht.Name & " is type " & TypeName(sht)
    Next sht

End Sub
```

If do not have chart sheets then using the **Sheets** collection is the same as using the **Worksheets** collection.

Conclusion

This concludes the post on the VBA Worksheet. I hope you found it useful.

The three most important elements of Excel VBA are Workbooks (https://excelmacromastery.com/excel-vba-workbook), Worksheets and Ranges and Cells (https://excelmacromastery.com/excel-vba-range-cells/). These elements will be used in almost everything you do. Understanding them will make you life much easier and make learning VBA much simpler.

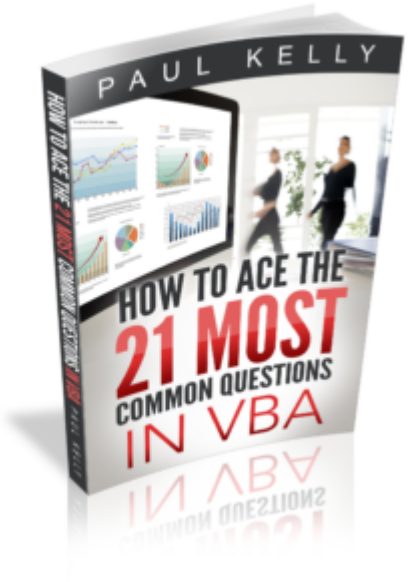
What's Next?

Free VBA Tutorial If you are new to VBA or you want to sharpen your existing VBA skills then why not try out the The Ultimate VBA Tutorial (https://excelmacromastery.com/vba-tutorial-1/).

Related Training: Get full access to the Excel VBA training webinars and all the tutorials (https://excelmacromastery.com/excel-vba-webinars/).

(**NOTE:** Planning to build or manage a VBA Application? Learn how to (https://www.theexcelvbahandbook.com/) build 10 Excel VBA applications from scratch.)

Get the Free eBook



(https://excelmacromastery.lpages.co/leadbox/14791da73f72a2%3A106f25298346dc/5636318331

Please feel free to subscribe to my newsletter and get exclusive VBA content that you cannot find here on the blog, as well as free access to my eBook, **How to Ace the 21 Most Common Questions in VBA** which is full of examples you can use in your own code.

DOWNLOAD NOW

(https://excelmacromastery.lpages.co/leadbox/14791da73f72a2%3A106f25298346dc/5636318331

(https://excelmacromastery.lpages.co/leadbox/14791da73f72a2%3A106f25298346dc/5636318331

(/#facebook) (/#twitter) (/#pinterest) (/#linkedin) (/#reddit)

(https://www.addtoany.com/share#url=https%3A%
vba-worksheet%2F&title=The%20Complete%20Gu

Sheets (https://excelmacromastery.com/tag/sheets/) VBA (https://excelmacromastery.com/tag/vba/) Workbooks (https://excelmacromastery.com/tag/workbooks/) Worksheets (https://excelmacromastery.com/tag/worksheets/)

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

Next
The Complete Guide to Ranges and Cells in Excel VBA
(https://excelmacromastery.com/excel-vba-range-cells/)

45 COMMENTS

Peter

September 15, 2015 at 8:30 am (https://excelmacromastery.com/excel-vba-worksheet/#comment-89)

Thank you for the excellent information for a VBA novice of some 10 years

Reply

Paul Kelly

September 15, 2015 at 9:31 am (https://excelmacromastery.com/excel-vba-worksheet/#comment-90)

You're welcome Peter.

Reply

George Cook

March 12, 2016 at 10:47 pm (https://excelmacromastery.com/excel-vba-worksheet/#comment-91)

Hi Paul

I've just come across your blog and it looks most informative. I've not had time to read through it yet and the answer to my question may be in there somewhere but in the meantime Is there a shorter way (in a macro) to write If Cell.Value = "A" Or Cell.Value = "B" Or Cell.Value = "C" Then

Cell.Value = 2

Many thanks

Reply

Paul Kelly

March 13, 2016 at 11:32 am (https://excelmacromastery.com/excel-vba-worksheet/#comment-92)

Hi George,

You can use the Select Case statement to do it

```
Select Case Cells(1,1).Value
    Case "A" To "C"
         Cells(1,1).Value = 2
End Select
```

Reply

George Cook

March 13, 2016 at 10:13 pm (https://excelmacromastery.com/excel-vba-worksheet/#comment-93)

Thanks for that Paul.

In the time between my query and your answer I did find a way except a bit longer than what you have suggested.

This is what I found

For Each oCell In Range("B11:M32")
Select Case oCell.Value
Case Is = "A", "B", "C": oCell.Value = 2
Case Is = "D", "E", "F": oCell.Value = 3
Case Is = "G", "H", "I": oCell.Value = 4
Case Is = "J", "K", "L": oCell.Value = 5
Case Is = "M", "N", "O": oCell.Value = 6

```
Case Is = “P”, “Q”, “R”, “S”: oCell.Value = 7
Case Is = “T”, “U”, “V”: oCell.Value = 8
Case Is = “W”, “X”, “Y”, “Z”: oCell.Value = 9
End Select
Next
```

Anyway thanks again. Much appreciated.

Reply

mike denley

August 9, 2016 at 7:04 pm (https://excelmacromastery.com/excel-vba-worksheet/#comment-94)

I use the code name to identify the different types of worksheets that are being created/used; allowing users to rename the worksheet as they see fit. From your experience what is the best way to set the code name ?

Reply

Paul Kelly

August 10, 2016 at 8:25 am (https://excelmacromastery.com/excel-vba-worksheet/#comment-95)

Hi Mike,

I normally set it manually in the property windows. It can be done using the code but normally this is not necessary.

Paul

Reply

harsha547

April 4, 2017 at 4:08 pm (https://excelmacromastery.com/excel-vba-worksheet/#comment-4891)

Thanks for sharing !!

Reply

Jorge C

April 13, 2017 at 3:10 am (https://excelmacromastery.com/excel-vba-worksheet/#comment-5341)

Hi Paul,
I am really really really thankful for your very informative, didactic and amusing posts. I have learned a lot since I began to read and rewrite your code, while adding slight changes and experimenting.
I just have a comment, I found that into the Sub AllSheetNames(), the second “next” asks for an “i” variable where the proper one should be “wrk”
Thank you again for everything and keep making live videos.

Reply

Paul Kelly

April 13, 2017 at 3:54 am (https://excelmacromastery.com/excel-vba-worksheet/#comment-5345)

Thanks Jorge, I have update the code. Glad you enjoy the website posts.

Reply

Afzal khan

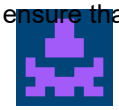
April 26, 2017 at 1:12 pm (https://excelmacromastery.com/excel-vba-worksheet/#comment-5764)

Thanks for your great effort Paul Kelly.
It’s very helpful to developing towards VBA programmer.

But one question, you have not shown how the “codename” for the worksheet is made. Only “codename” is displayed in the property screen shot.

Thanks for sharing knowledge…. I am glad i found this site.

Reply



Afzal khan

April 26, 2017 at 1:15 pm (https://excelmacromastery.com/excel-vba-worksheet/#comment-5765)

Ya i got it, its already mentioned there.
Sorry i missed it.

Reply

Paul Kelly

April 27, 2017 at 3:34 am (https://excelmacromastery.com/excel-vba-worksheet/#comment-5782)

No problem Afzal.

Reply

Curzio

June 19, 2017 at 8:04 am (https://excelmacromastery.com/excel-vba-worksheet/#comment-8105)

Hallo Paul, your whole site is the mandatory start lane and pit stop for every vba writer.
I get confused by example “Code Name in other Workbooks”: I miss somewhere something like

Dim iWB As Workbook
For Each iWB In Application.Workbooks
‘Loop thru Worksheets and check condition
Next

Thank again and kind regards

Reply

Chris K.

September 27, 2017 at 2:17 am (https://excelmacromastery.com/excel-vba-worksheet/#comment-16881)

Hi Paul,

You stated that codenames can only be used in “Thisworkbook”, but in my testing the module that I’m running writes to two workbooks that I have opened at the same if the other workbook is active. Here’s what I did. I declared a public codename in workbook1 and set it in ThisWorkbook object using Auto_Open. (I’ve tested setting it in a module as well and I get the same results) The module which contains my code starts with “With ThisWorkbook” before

calling the public variable. As a test, I renamed a sheet from each workbook with the same codename. I would expect that my code would only run in workbook1 as the public variable is written, set and called in workbook1; on top of that I declared “With Thisworkbook” at the start of my module.

Do you know how I can prevent the module from running in the wrong workbook? I know I can simply rename the codename in workbook1 to something highly unlikely, but for theoretical purposes I’d like not to do that.

Thanks,
Chris

Reply

Chris K.

September 27, 2017 at 2:45 am (https://excelmacromastery.com/excel-vba-worksheet/#comment-16882)

Upon further testing regardless of the other workbooks codename if it was active my module would write to it. I changed the codenames to differ and I had the same results; even if the other workbook was an xlsx file the module still wrote to it. For now, I’m going with a workaround to state in my code Thisworkbook.active. This prevented writing to the other workbook even if I clicked on it during runtime.

Reply

Chris K.

September 27, 2017 at 3:11 am (https://excelmacromastery.com/excel-vba-worksheet/#comment-16884)

Okay, so what I found out is that in my module I have lines of code such as range(“A2”).value, etc. As a result, excel or vba reads it like ActiveSheet.Range(“A2”).value. I had thought that starting the code with “With Thisworkbook” I would have avoided the code from writing or reading other workbooks.

What I did to resolve this is instead of range(“A2”).value and I used VBsht.Range(“A2”).value with VBsht being my public variable or object.

Reply

DangBui

February 23, 2018 at 7:28 am (https://excelmacromastery.com/excel-vba-worksheet/#comment-24325)

I am new to VBA and am learning from beginning. Thank you for your useful sharing!

Reply

Paul Kelly

February 25, 2018 at 10:14 am (https://excelmacromastery.com/excel-vba-worksheet/#comment-24396)

You're welcome.

Reply

Guity

March 6, 2018 at 8:19 pm (https://excelmacromastery.com/excel-vba-worksheet/#comment-24894)

OMG, I have been looking for someone to teach me macros as how to write macros. I found this site, it seems to me a new world has opened its door to me. It is unbelievable!!!

Reply

Paul Kelly

March 7, 2018 at 4:56 am (https://excelmacromastery.com/excel-vba-worksheet/#comment-24907)

Hi Guity,

Glad you like it so much:-)

Paul

Reply

Nick Jensen

June 17, 2019 at 12:03 pm (https://excelmacromastery.com/excel-vba-worksheet/#comment-65350)

Hello Paul,

Great stuff you have here. I really appreciate your thoroughness and clarity. For a nitpicker like me, it is great to have clear explanations for many things you write in code.

In this article, there is a section called “Code Name in other Workbooks” which I have a question about. You write, “There is another way of getting the sheet name of an external workbook using the code name. You can use the VBProject element of that Workbook… I would recommend using the method in the previous example rather than this one”. Could you explain why you recommend the looping technique over using VBProject elements?

Reply

Paul Kelly

June 21, 2019 at 5:09 am (https://excelmacromastery.com/excel-vba-worksheet/#comment-65679)

Using the VBProject requires that the user has checked the “Trust Access” box in the security settings. Which is fine if you are just using it yourself.

Reply

Loki

July 26, 2019 at 5:19 am (https://excelmacromastery.com/excel-vba-worksheet/#comment-69014)

I don’t know how I stumbled upon your site, but I am confident that this will hands down be the greatest thing in my VBA journey.

I believe in the sub UseSheet of ‘Code Name in other Workbooks’ section, the argument ThisWorkbook must be replaced with the workbook you are trying to access the sheets of, like Workbooks(“Book2.xlsx”) .

Thanks

Reply

Paul Kelly

August 1, 2019 at 8:32 am (https://excelmacromastery.com/excel-vba-worksheet/#comment-69533)

Thanks Loki. ThisWorkbook refers to the current workbook i.e. the workbook containing the VBA code.

Reply

Ashish

August 28, 2019 at 6:29 am (https://excelmacromastery.com/excel-vba-worksheet/#comment-73603)

Dear Paul,
Thanks for this informative article. I tried to set two worksheetsheet variables to access two sheets of a new workbook but VBA is throwing error on assigning second sheet to second variable. Below is the code

```
Sub test27()
Dim temp_wkb As Workbook
Dim temp_sheet1, temp_sheet2 As Sheets
Set temp_wkb = Workbooks.Add
Set temp_sheet1 = temp_wkb.Sheets(1)
Set temp_sheet2 = temp_wkb.Sheets(2)
End Sub
```

I get Error 13 type mismatch on “Set temp_sheet2 = temp_wkb.Sheets(2)”

Could you please guide as to what am I doing wrong here?

Regards
Ashish

Reply

Paul Kelly

August 30, 2019 at 3:24 am (https://excelmacromastery.com/excel-vba-worksheet/#comment-73884)

temp_sheet2 As Sheets

This code should be “As Sheet” or better still “As Worksheet”

Reply

Dušan

October 26, 2019 at 12:49 pm (https://excelmacromastery.com/excel-vba-worksheet/#comment-83726)

Why this Sub procedure doesn’t work for me. Please answer.
Public Sub UseCodeName2()

‘ Using the code name of the worksheet
Debug.Print CodeName.Name
CodeName.Range(“A1”) = 45
CodeName.Visible = True

End Sub

Goodby ! Dušan

Reply

Paul Kelly

October 30, 2019 at 8:47 am (https://excelmacromastery.com/excel-vba-worksheet/#comment-84539)

You must have a worksheet with the code name set to **CodeName**

Reply

Jérôme Turmel

December 11, 2019 at 3:41 pm (https://excelmacromastery.com/excel-vba-worksheet/#comment-94307)

Paul, I read all the articles on your site and I would like to show you all my gratitude for your information sharing. Having a simple structure with multiple examples, easy to copy and to understand, makes the student’s job very simple. You are an excellent teacher.

Thank you for the nice work.

Reply

Paul Kelly

December 13, 2019 at 8:38 am (https://excelmacromastery.com/excel-vba-worksheet/#comment-94651)

Thanks Jérôme

Reply

ludovic

March 14, 2020 at 10:35 am (https://excelmacromastery.com/excel-vba-worksheet/#comment-112376)

hi,
is it possible to refer with a variable to the sheets codename;

dim sht as worksheet, strCName as string
strCName = “cnTestCodeName”
set sht = sheets(strCName) ‘this does not fly for me when i want to refer to the sheets CodeName

in general impressive tutorials / on udemy it could had gone some deeper for me, but good.

Reply

Paul Kelly

March 14, 2020 at 11:37 am (https://excelmacromastery.com/excel-vba-worksheet/#comment-112389)

Why would you want to? When you use the codename you are directly accessing the worksheet object.
You can do this:
Dim sht As Worksheet
Set sht = sheets(cnSheet1.Name)

But it’s the same as
Set sht = cnSheet1

Reply

ludovic

March 15, 2020 at 1:02 pm (https://excelmacromastery.com/excel-vba-worksheet/#comment-112805)

Thanks for reply, will try to explain my 'idea'.
I have a hidden 'classSheet' that holds a range of predefined zoom values for a set of sheets in the workbook. A function setting all those predefined zooms to the 'default' by looping true all sheets referred to by codename. Now if I use the sheetname instead the user could have changed this sheetname. For the codename I wanted to check if the sheet still exists (Excel will crash referring to a sheets codename that does not exist) and loop true the collection. As a result, of this approach, I thought to be pretty sure all goes as intended to.
It is probably not a capital case but I thought I miss out on something easy once you know. Thanks.

Reply

Paul Kelly

March 18, 2020 at 6:14 am (https://excelmacromastery.com/excel-vba-worksheet/#comment-113661)

You can use the *SheetFromCodeName()* function in this section (https://excelmacromastery.com/excel-vba-worksheet/#Code_Name_in_other_Workbooks) of the post to get the worksheet object from the codename text.

-Paul

Reply

ludovic

March 21, 2020 at 6:03 am (https://excelmacromastery.com/excel-vba-worksheet/#comment-114342)

sorry for late reply, i only have time in my weekends. thanks for the tip, prefect workable approach.
Ludovic

SHIN DONGJUN

March 19, 2020 at 12:03 am (https://excelmacromastery.com/excel-vba-worksheet/#comment-113838)

If I found you earlier, I could dig treasure in Excel. Until now, I just have used simple function of it. Anyway, I keep turning on your tutorial everyday. Again, Thanks for your great lecture.

Reply

Paul Kelly

March 20, 2020 at 2:47 am (https://excelmacromastery.com/excel-vba-worksheet/#comment-114084)

Thanks Shin.

Reply

Rajeev Shukla

April 11, 2020 at 5:47 pm (https://excelmacromastery.com/excel-vba-worksheet/#comment-118714)

Sir,
I have a workbook in which there is a worksheet in which we had used some formulas. I want to make a duplicate sheet by a command button with only values (paste special).
Please Guide me

Reply

Lutta

April 15, 2020 at 12:54 am (https://excelmacromastery.com/excel-vba-worksheet/#comment-118739)

Hi Paul and all,

Thanks for the great resources herein.

I would like to copy the values in cell C1 from a number of worksheets and paste in range C2 : C13 of my summary worksheet in the same workbook, and subsequently in the empty cells that follow. But instead of getting a set of 12 unique values, I get a long list with the same values from one worksheet.

Here is my code:
Sub Tabulates()

Dim erow As Long

For i = 1 To Worksheets.Count
If Worksheets(i).Name “Data” Then

erow = Sheet1.Cells(Sheet1.Rows.Count, 1).End(xlUp).Offset(1, 0).row + 1

Worksheets(i).Range(“C1”).Copy
Sheet1.Range((“C2:C13”) & erow).Select

Selection.PasteSpecial Paste:=xlPasteValues, Operation:=xlNone, SkipBlanks:=False, Transpose:=True
Application.CutCopyMode = False

End If
Next
End Sub

Kindly assist. Thank you.

Lutta

Reply

Reply

Rupa

June 25, 2020 at 9:52 pm (https://excelmacromastery.com/excel-vba-worksheet/#comment-118944)

Can you please help me to solve this problem? Eight number line shows error.

```
Sub Macro1()
Dim c, m, n, f As Integer
c = 4 'Total room number
m = 1
For f = 1 To c
If (f <= c – 3) Then
Worksheets("Bedroom" & m).Activate
Worksheets("Load Factor").Cells(6, f + 2).Formula = "="Bedroom"&m!B2"
m = m + 1
End If
Next f
End Sub
```

Reply

Paul Kelly

July 6, 2020 at 1:58 pm (https://excelmacromastery.com/excel-vba-worksheet/#comment-118965)

What is the error?

Reply

Abdul Majeed

August 19, 2020 at 4:51 am (https://excelmacromastery.com/excel-vba-worksheet/#comment-119102)

Dear Sir,
I am looking for a vba code to find whether a worksheet is “Protected” or “Unprotected”.
Thanks & Kind Regards
Majeed

Reply

Marc Sanschagrin

January 20, 2021 at 9:12 pm (https://excelmacromastery.com/excel-vba-worksheet/#comment-119431)

Hi,
Great resources, specially for an absolute beginner like me.

I almost know nothing about Excel, but I'm really motivated to learn.

I'm on a project to collect software inventory from a list of computers, for this part I have managed to make individual inventory and merge the different worksheets in a single workbook.

In my workbook I also have the content of the base image, which I use as a reference to remove the lines that are in this specific sheet from the other sheets, that's working fine.

But as it sometimes happens That on certain computers, there is no other software than the base image I get empty inventory sheets after the cleanup and that's causing me trouble as sometimes my account doesn't have access to certain computers and instead of crashing my entire tool, I issue an empty sheet and that's my way to find computers on which I have no access.

But after cleaning up my sheets I can't figure out which are empty because the lack of access or the absence of software added to the base image.

So, here's where my problem begins, I just can't figure out how to instead of deleting those lines, just change the background colors for these rows.

So here is my actual code and any help would be really appreciated.

****

```
Option Explicit

Sub NettoyeurDeFeuilles()
Application.ScreenUpdating = False

Dim ws As Worksheet
Dim startRow As Integer

startRow = 1
Dim row As Integer

row = startRow

Dim bRow As Integer

For Each ws In ThisWorkbook.Worksheets
With ws
'Ne pas chercher dans la feuille Image_De_Base_W7!
```

```
If ws.Name = “Image_De_Base_W7” Then GoTo myNext

Do While (Worksheets(ws.Name).Range(“A” & row).Value “”)

Dim aVal As String

aVal = Worksheets(ws.Name).Range(“A” & row).Value

bRow = startRow

Do While (Worksheets(“Image_De_Base_W7”).Range(“A” & bRow).Value “”)

Dim aVal2 As String

aVal2 = Worksheets(“Image_De_Base_W7”).Range(“A” & bRow).Value

If (aVal = aVal2) Then

Worksheets(ws.Name).Rows(row).Delete ‘ Une entrée à supprimé à été trouvé
row = row – row

‘ Worksheets(ws.Name).Rows(row).Cells.Interior.ColorIndex = 3

Exit Do

End If

bRow = bRow + 1

Loop

row = row + 1
Loop

myNext:
End With

Next ws

End Sub

**********
```

Reply

Anil

June 28, 2021 at 3:42 am (https://excelmacromastery.com/excel-vba-worksheet/#comment-119707)

Suppose you have a 7 sheets.

I want to protect sheet 2,4,6 by immediate window vba code.

how?

Reply

LEAVE A REPLY

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

Name *

Email *

Website

I'm not a robot

reCAPTCHA
Privacy - Terms

Post Comment