Return to VBA Code Examples

VBA Not Equal to, Greater Than or Equal To & Other Comparison Operators

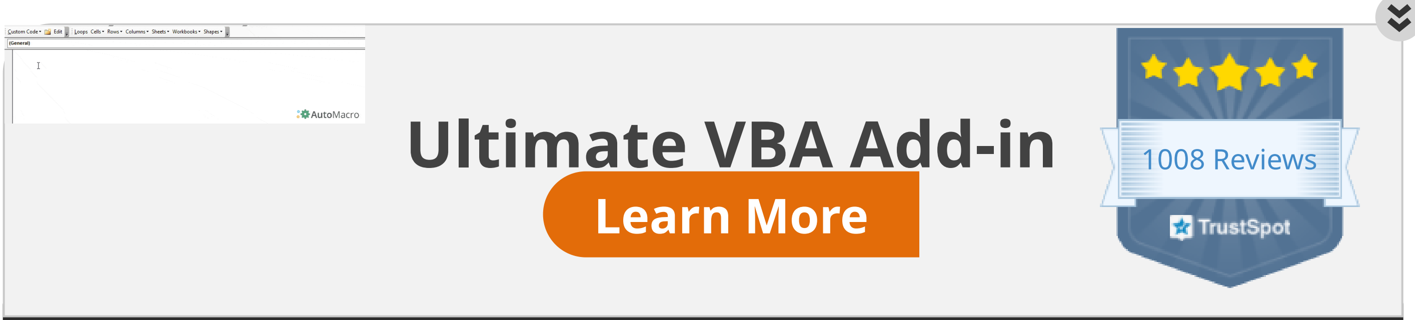
IN THIS ARTICLE

VBA allows you to use comparison operators to compare values. By using these operators, you can compare values and return a Boolean True or False as a result.

These are the main comparison operators used in VBA:

Comparison Operator	Explanation
=	Equal to
<>	Not Equal to

<	Less than
<=	Less than or Equal to



```
MsgBox 5 <> 3
```

Cell Value Not Equal To

There are several ways you might compare numbers. In the previous example, we hard-coded 5 and 3 into our code. Let’s demonstrate two other ways to compare values.

This example will test if two cell values are not equal to each other:

```
MsgBox Range("A1").Value <> Range("B1").value
```

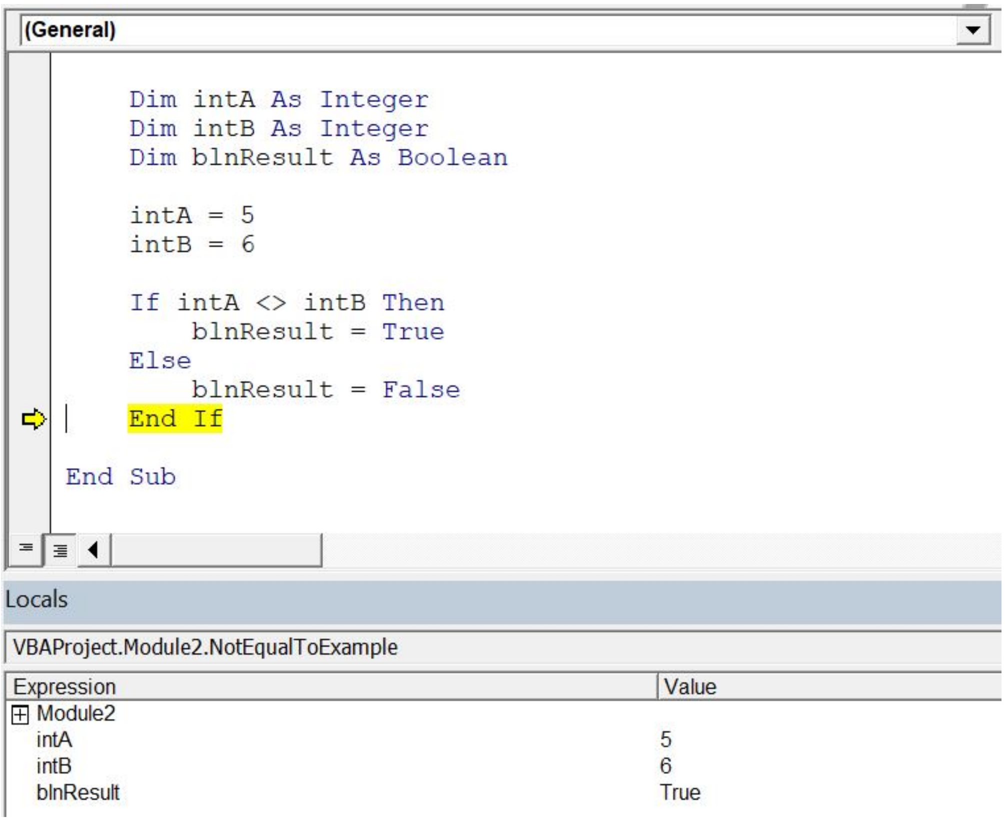
Another way to compare values is with Variables.

```
Sub NotEqualTo ()
  Dim intA As Integer
  Dim intB As Integer
  Dim blnResult As Boolean

  intA = 5
  intB = 6

  If intA <> intB Then
    blnResult = True
  Else
    blnResult = False
```

In this example, we want check if Integer Variable *intA* is not equal to *intB*. If this is true, the value of Boolean variable *blnResult* will be True, otherwise, it will be False.

We use the **<>** operator in the If Statement to check if the values of *intA* and *intB* are different:

The value of *intA* is 5 and the value of *intB* is 6, the variables are not equal, therefore the *blnResult* returns True:



```
    Dim intA As Integer
    Dim intB As Integer
    Dim blnResult As Boolean

    intA = 5
    intB = 6

    If intA <> intB Then
        blnResult = True
    Else
        blnResult = False
    End If

End Sub
```

```
VBAProject.Module2.NotEqualToExample
Expression          Value
⊞ Module2
  intA              5
  intB              6
  blnResult         True
```

Equal To

The ***Equal to*** operator works exactly the same. It checks if two values are equal and returns True or False. Here is the example code:

```
Sub EqualTo()
```

```
  intA = 5
  intB = 5

  If intA = intB Th
```

In this example, we want to check if *intA* is equal to *intB*. If this is true, the value of Boolean *blnResult* will be True, otherwise, it will be False.

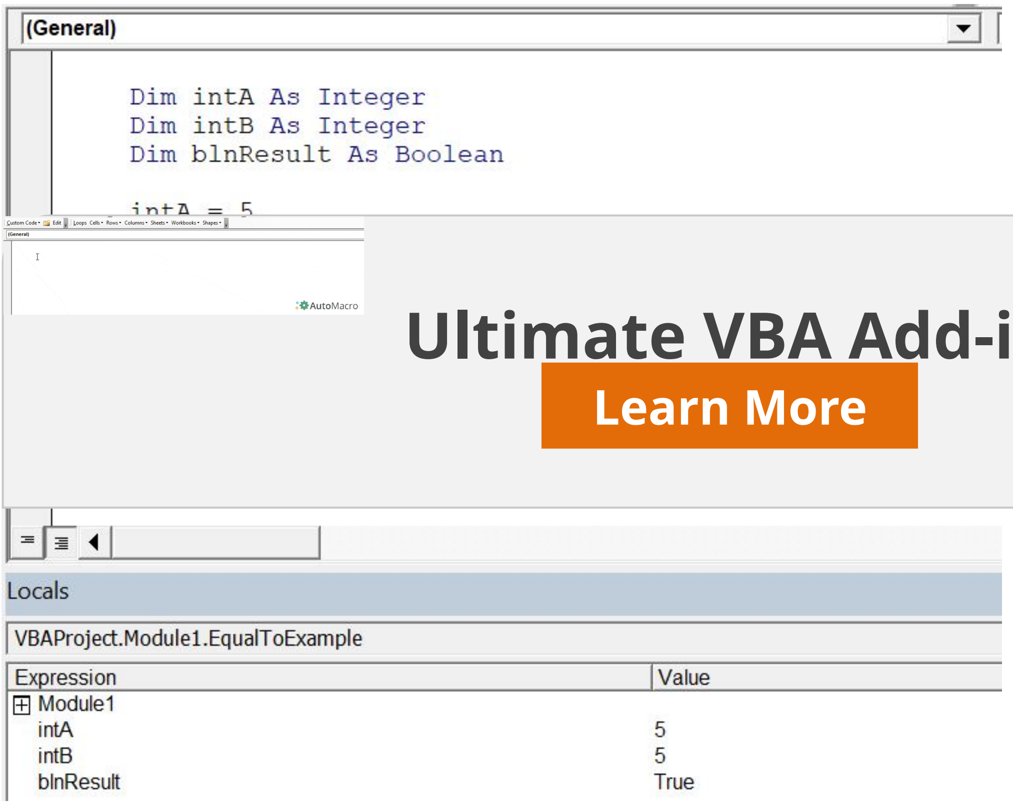
First, we set values of *intA* and *intB* to 5:

```
intA = 5
intB = 5
```

After that, we use the *=* operator in the If statement to check if the values of *intA* and *intB* are equal:

```
If intA = intB Then
    blnResult = True
Else
    blnResult = False
End If
```

Both variables are equal to 5, therefore the *blnResult* returns True:



```
    Dim intA As Integer
    Dim intB As Integer
    Dim blnResult As Boolean

    intA = 5
```

VBAProject.Module1.EqualToExample

```
Expression                  Value
⊞ Module1
  intA                      5
  intB                      5
  blnResult                 True
```

Greater Than / Less Than

The ***Greater Than*** (>) and ***Less Than*** (<) operators work exactly the same way.

Greater Than

```
MsgBox 5 > 3
```

Less Than

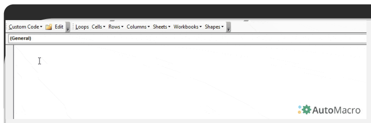
```
MsgBox 5 < 3
```

Greater Than or Equal To / Less Than or Equal To

To test if a value is ***Greater Than* or *Equal To***, combine the Equal To (=) operator and the Greater Than (>) / Less Than (<) operators.

```
MsgBox 5 >= 3
```

Less Than or Equal To

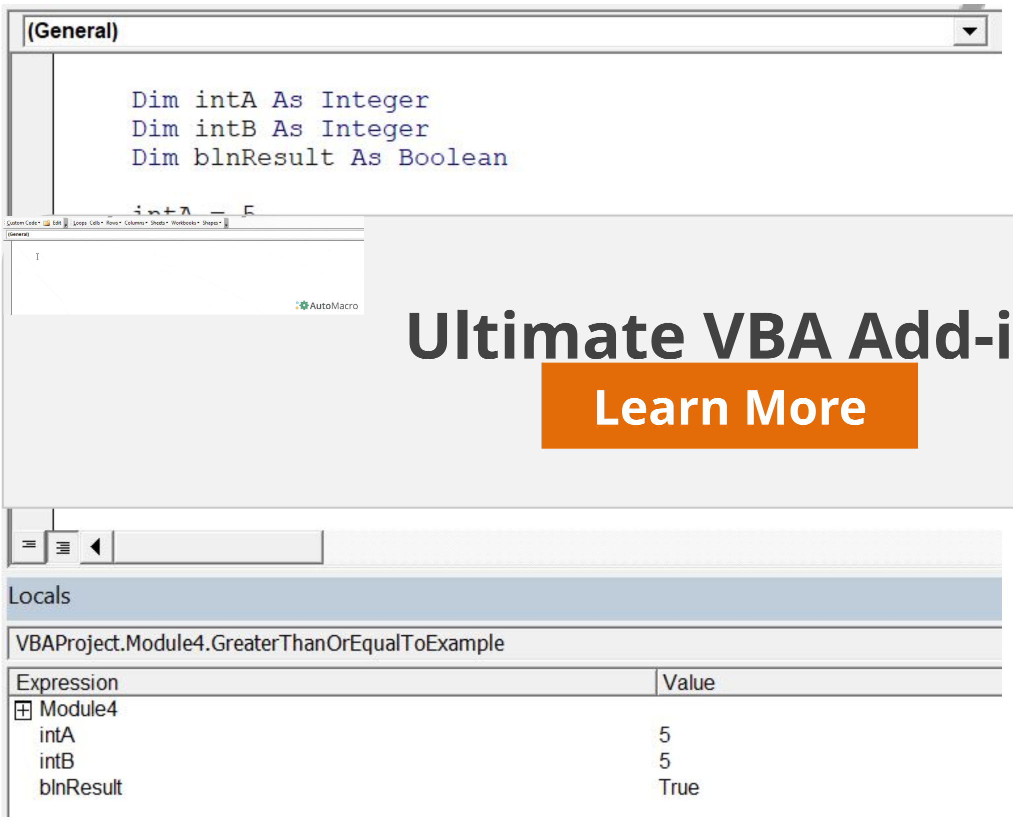


```
Sub GreaterThanEqualTo ()
  Dim intA As Integer
  Dim intB As Integer
  Dim blnResult As Boolean

  intA = 5
  intB = 5

  If intA >= intB Then
    blnResult = True
  Else
    blnResult = False
  End If

End Sub
```

In this example, we want to check if *intA* is greater than or equal to *intB*. If this is true, the value of Boolean *blnResult* will be True, otherwise, it will be False.

We use the *>=* operator in the If statement to check if the value of *intA* is greater than or equal to *intB*:

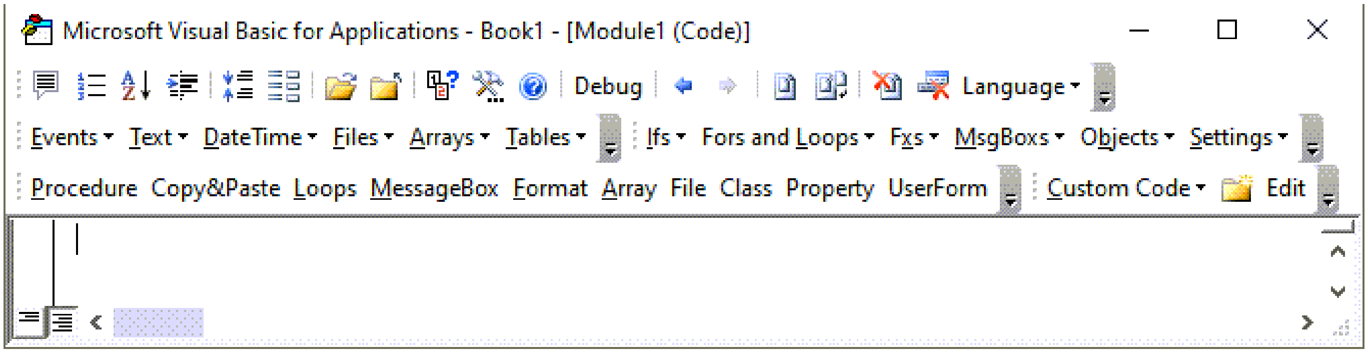
```
If intA >= intB Then
    blnResult = True
Else
    blnResult = False
End If
```



```
    Dim intA As Integer
    Dim intB As Integer
    Dim blnResult As Boolean

    intA = 5
```

VBAProject.Module4.GreaterThanOrEqualToExample

Expression	Value
⊞ Module4	
intA	5
intB	5
blnResult	True



Microsoft Visual Basic for Applications - Book1 - [Module1 (Code)]
Debug
Language
Events Text DateTime Files Arrays Tables
Ifs Fors and Loops Fxs MsgBoxs Objects Settings
Procedure Copy&Paste Loops MessageBox Format Array File Class Property UserForm
Custom Code Edit

Is Operator

```
Set ws1 = Sheets("Sheet1")
Set ws2 = Sheets("Sheet2")

If ws1 Is ws2 Th
```

```
End Sub
```

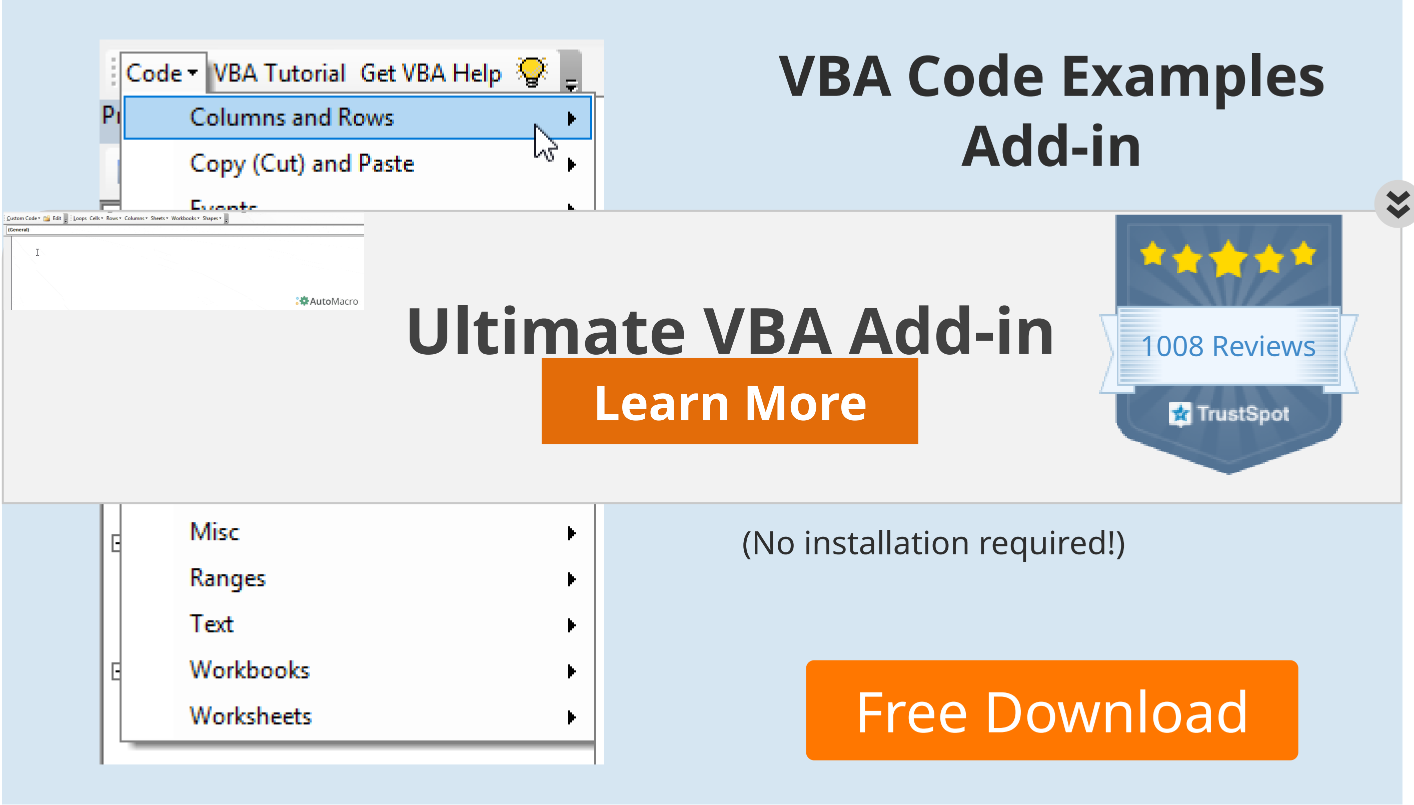
Like Operator

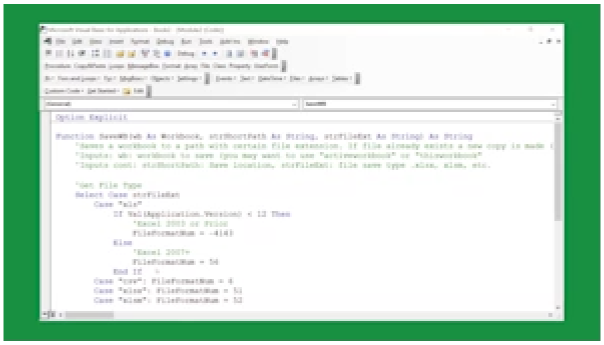
The Like Operator can be used to find inexact text matches. This example will test if a string starts with "Mr."

```
Sub LikeDemo()

Dim strName As String
Dim blnResult As Boolean

strName = "Mr. Michael James"

If strName Like "Mr*" Then
    blnResult = True
Else
    blnResult = False
End If

End Sub
```

If you want to learn how to compare strings, click here: VBA Compare Strings – StrComp

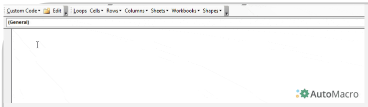
If you want to learn more about how to use logical operators, click here: VBA Logical Operators

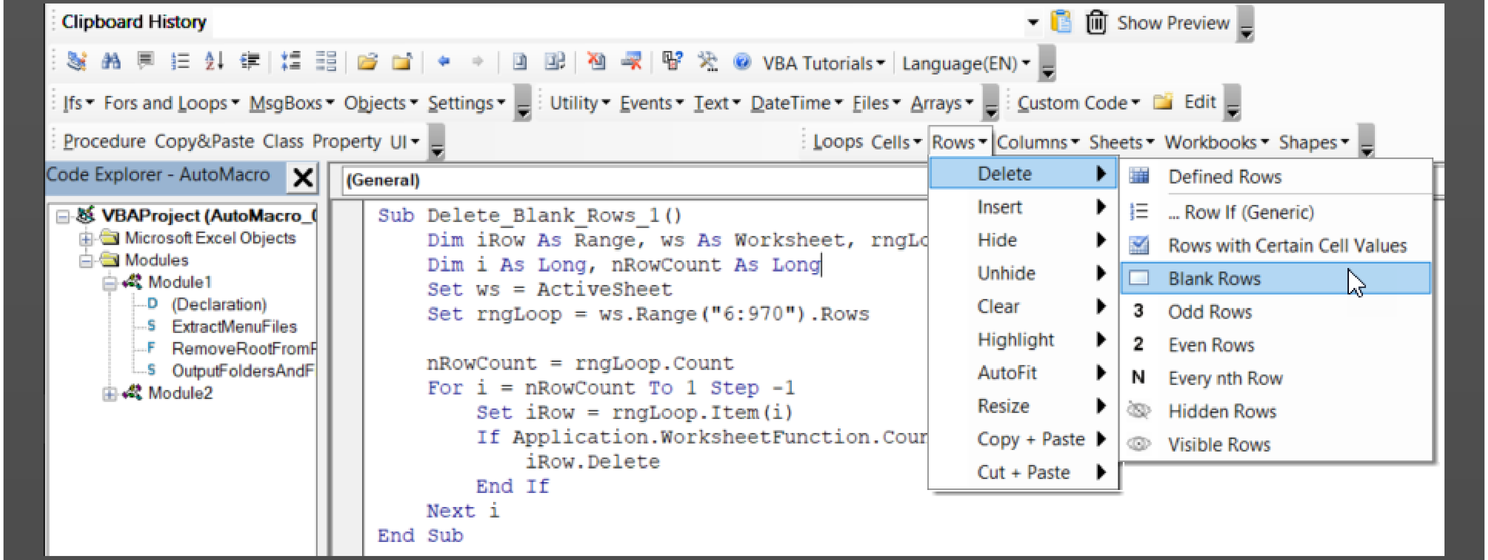




```
Option Explicit

Function SaveWB(wb As Workbook, strShortPath As String, strFileExt As String) As String
    'Saves a workbook to a path with certain file extension. If file already exists a new copy is made (
    'Inputs: wb: workbook to save (you may want to use "activeworkbook" or "thisworkbook"
    'Inputs cont: strShortPath: Save location, strFileExt: file save type .xlsx, xlsm, etc.

    'Get File Type
    Select Case strFileExt
        Case "xls"
            If Val(Application.Version) < 12 Then
                'Excel 2003 or Prior
                FileFormatNum = -4143
            Else
                'Excel 2007+
                FileFormatNum = 56
            End If
        Case "csv": FileFormatNum = 6
        Case "xlsx": FileFormatNum = 51
        Case "xlsm": FileFormatNum = 52
```

Return to VBA Code Examples





```
Sub Delete_Blank_Rows_1()
    Dim iRow As Range, ws As Worksheet, rngLo
    Dim i As Long, nRowCount As Long
    Set ws = ActiveSheet
    Set rngLoop = ws.Range("6:970").Rows

    nRowCount = rngLoop.Count
    For i = nRowCount To 1 Step -1
        Set iRow = rngLoop.Item(i)
        If Application.WorksheetFunction.Cour
            iRow.Delete
        End If
    Next i
End Sub
```

VBA CODE GENERATOR

EXCEL BOOT CAMP

FORMULAS TUTORIAL

EXCEL FORMULAS EXAMPLES LIST

FUNCTIONS LIST

SHORTCUT COACH

SHORTCUT TUTORIAL

SHORTCUTS LIST

DISCLAIMER

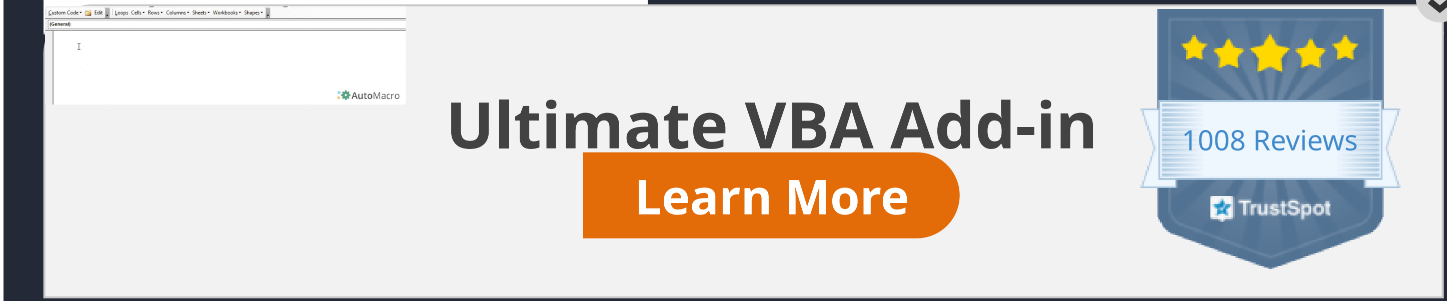
PRIVACY POLICY

COOKIE POLICY

YOUR PRIVACY CHOICES

NOTICE AT COLLECTION

Register for FREE and get:


AutoMacro
Ultimate VBA Add-in
Learn More
1008 Reviews
TrustSpot