Article: Excel - VBA - How To use Looping
Date: 24/01/2012
Posted by: HeelpBook Staff
Source: Link
Permalink: Link

Excel - VBA - How To use Looping

When you create macros or applications in **Microsoft Visual Basic for Applications (VBA)**, it is often necessary to run through a section of code several times. **Visual Basic for Applications (VBA)** provides several methods with which to repeat, or "**loop**" through, a section of code.

Prior to Begin

Microsoft provides programming examples for illustration only, without warranty either expressed or implied. This includes, but is not limited to, the implied warranties of merchantability or fitness for a particular purpose.

This article assumes that you are familiar with the programming language that is being demonstrated and with the tools that are used to create and to debug procedures. Microsoft support engineers can help explain the functionality of a particular procedure, but they will not modify these examples to provide added functionality or construct procedures to meet your specific requirements. When you are deciding which looping structure to use, there are several considerations to be aware of. For example:

- ***Do I know exactly how many times I want to loop through the code?***
- ***If so, how many?***
- ***If not, is there a specific condition on which I want the macro to exit the loop? If the loop is conditional, do I want to test the condition before or after the code is executed?***

Sample Visual Basic Procedures

The following examples demonstrate the various looping structures available in **Visual Basic for Applications (VBA)**. Unless otherwise indicated, the examples assume a cell or range of cells is selected. Some other things to note:

- Within each example, the message box function (**MsgBox**) is used to display information. In these examples, wherever a **MsgBox** function occurs is where you should enter the code to be repeated through each iteration of the loop.
- Some of the comments in the code describe alternative methods for accomplishing a particular task or for doing different tasks with similar code.
- The tilde **(~)** symbol used within some of the comments should be replaced with the name of the indicated object [the object enclosed in quotation marks, for example **worksheets("sheet1")**], or the object's index number.

For...Each...Next

This example uses a **For**...**Each**...**Next** statement to loop through all the cells in a selected range (***the comments show an alternative method***):

```
' To show the contents of each cell in a selection or specific range.
Sub for_each_demo()

   ' Or, use "In Worksheet(~).Range(~)" to specify a specific range.
   For Each cell In Selection

      ' Displays cell contents in message box.
      MsgBox cell.Value

      ' Reset cell to next object.
   Next
End Sub
```

For <variable> = <n1> to <n2>

```
' This example loops through the code a specified number of times.
   Sub for_demo()

      ' Sets x to 1, adds 1 to x until x = 5, loops 5 times.
      For x = 1 To 5 Step 1

         ' Displays value of x in msgbox.
         MsgBox x

         ' Returns to top of loop 4 times.
      Next
   End Sub
```

Do...Until with Test at Beginning of Loop

This example uses a **Do**...**Until** loop to "***walk***" down a column of cells until the first empty cell is reached. Because the macro performs the test at the beginning of the loop, ***if the first cell is empty***, the code inside the loop will not be run.

```
' Performs Do loop, testing at top of loop.
' Loops until empty cell is reached.
' Note that you can also use a Do While Not IsEmpty(ActiveCell) loop.

Sub test_before_do_loop()

   ' Test contents of active cell; if active cell is empty, exit loop.
   Do Until IsEmpty(ActiveCell)

      ' Displays cell contents in message box.
      MsgBox ActiveCell.Value

      ' Step down 1 row to the next cell.
      ActiveCell.Offset(1, 0).Select

     ' Return to top of loop.
   Loop
End Sub
```

Do...Until with Test at End of Loop

This example also uses a Do loop, but it tests at the bottom of the loop. The first line of the macro tests the first cell. Without the first line of code to test the initial cell, the code would execute at least one time, because the loop tests at the bottom.

```
' Performs Do loop, testing at bottom of loop.

Sub test_after_do_loop()

   ' Test to see if first cell is empty.
   If IsEmpty(ActiveCell) Then Exit Sub

      ' Begin loop.
      Do

         ' Displays cell contents in message box.
         MsgBox ActiveCell.Value

         ' Steps down one row to the next cell.
         ActiveCell.Offset(1, 0).Select

         ' Test contents of active cell; if empty, exit loop
         ' or Loop While Not IsEmpty(ActiveCell).

      Loop Until IsEmpty(ActiveCell)
   End Sub
```

CAUTION: Do not branch into the body of a **While**...**Wend** loop without executing the **While** statement. Doing so may cause run-time errors or other problems that are difficult to locate.
The **Do**...**Loop** statement provides a more structured and flexible way to perform looping.

The **While**...**Wend** loop is included in **Visual Basic for Applications (VBA)** for backward compatibility.

While...Wend Loop

NOTE: The **While**...**Wend** loop is included in **Visual Basic for Applications** for backward compatibility.
This example uses a **While**...**Wend** loop. This loop tests at the top of the loop only.

```
 ' Performs While loop, testing at top of the loop.
Sub While_loop_demo()

   ' Sets condition of loop, while active cell is not empty.
   While Not IsEmpty(ActiveCell)

      ' Displays cell contents in message box.
      MsgBox ActiveCell.Value

      ' Step down one row to the next cell.
      ActiveCell.Offset(1, 0).Select

   ' End While loop.
   Wend
End Sub
```

If...Then..GoTo Loop

This example creates a loop by using "***If <condition> Then GoTo <line label>***" structure. This structure is tested at the bottom of the loop.

```
     Sub loop_using_goto()

      ' Test to see if first cell is empty.
      If IsEmpty(ActiveCell) Then Exit Sub

      ' Line label indicating top of loop.
   top:

      ' Displays cell contents in message box.
      MsgBox ActiveCell.Value

      ' Step down one row to the next cell.
      ActiveCell.Offset(1, 0).Select

      ' Test to see if new cell is empty.
      If Not IsEmpty(ActiveCell) Then GoTo top

   End Sub
```

For more information about looping structures, in the **Visual Basic Editor (VBE)**, click the **Office Assistant**, type ***loops***, and then click Search to view the topics returned.