How to use VBA procedures to generate a list of sheet names in an Excel workbook

by **Susan Harkins** in **Software**
on May 25, 2021, 6:32 AM PDT

Use one or both of these simple Microsoft Excel macros to list all the sheets in an Excel workbook.

Image: Aajan Getty Images/iStockphoto

Whether your Microsoft Excel workbook has three sheets or 50, knowing what you have is important. Some complex applications require detailed documentation that includes sheet names. For those times when a quick look isn't good enough, and you must have a list, you

have a couple of options: You can go the manual route, which would be tedious and need updating every time you made a change. Or you can run a quick VBA procedure. In this article, I'll show you easily modified code that generates a list of sheet names and hyperlinks. There are other ways to generate a list, but I prefer the VBA method because it's automated and easy to modify to suit individual requirements.

SEE: 69 Excel tips every user should master (TechRepublic)

I'm using Microsoft 365 on a Windows 10 64-bit system, but you can use earlier versions. To save time, download the .xlsx, .xls and .cls files. Macros aren't supported by the online version. This article assumes you have basic Excel skills and are familiar with VBA, but even a beginner should be able to follow the instructions to success.

How to enter and run code in VBA

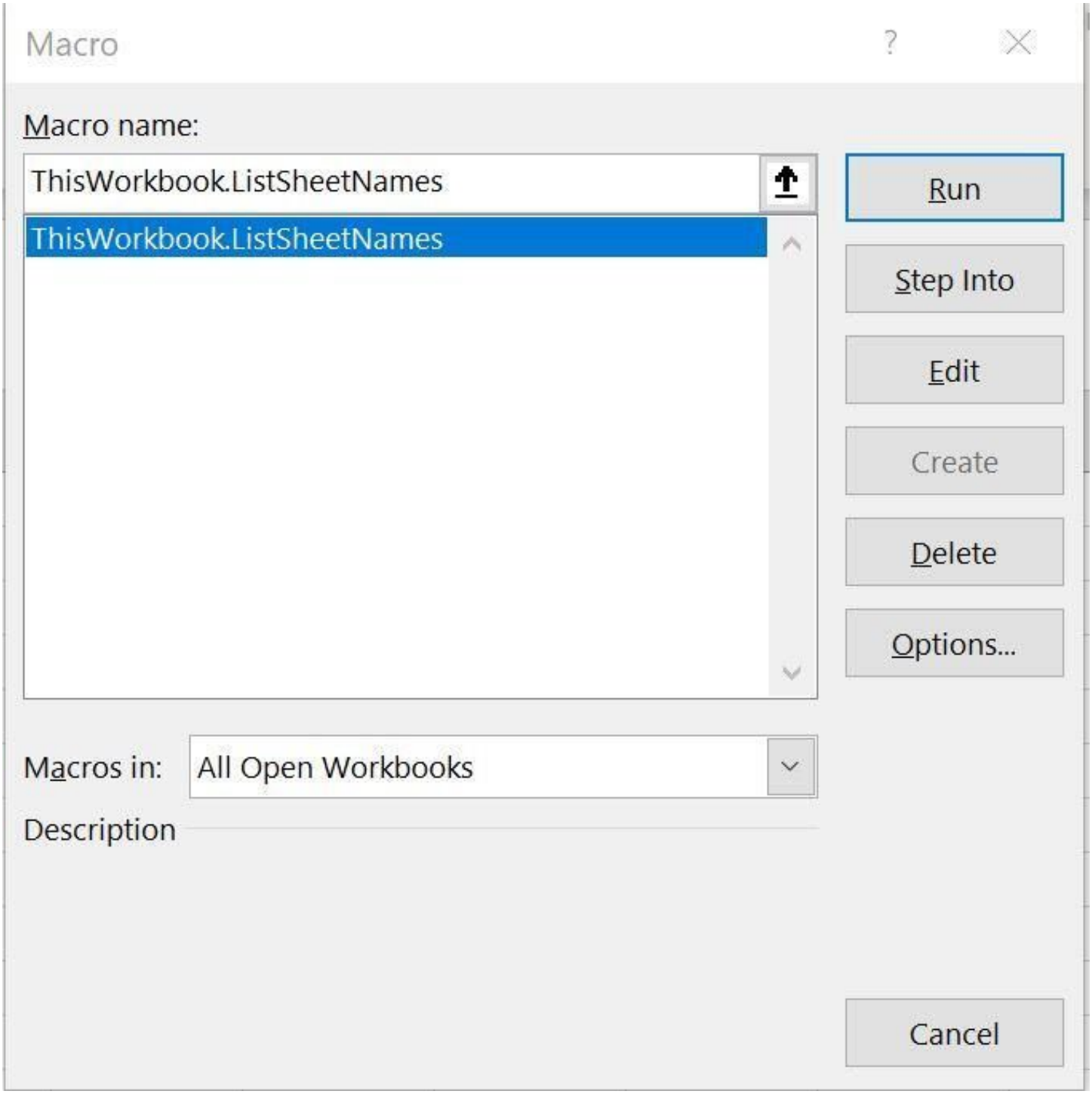
If you're new to VBA code, you might be wondering about the terms procedure and macro. You'll see them used interchangeably when VBA is the language used. This isn't true for all languages. In VBA, both a procedure and a macro are a named set of instructions that are preformed when called. Some developers refer to a sub procedure as a macro and a function procedure as a procedure because a procedure can accept arguments. Many use the term macro for everything. Some, like me, tend to use the term procedure for everything. Furthermore, Access has a macro feature that is separate from any VBA code. Don't get too hung up on the terms.

SEE: Windows 10: Lists of vocal commands for speech recognition and dictation (free PDF) (TechRepublic)

To enter VBA code, press Alt + F11 to open the Visual Basic Editor. In the Project Explorer to the left, choose ThisWorkbook and enter the code. If you're using a ribbon version, you must save the file as a macro-enabled file to use macros. You can also import the downloadable .cls file that contains the code. Or, you can work with either of the downloadable Excel workbooks. If you're entering the code yourself, please don't copy it from this web page. Instead, enter it manually or copy it to a text editor and then paste that code into a module.

When in the VBE, press F5 to run a procedure, but be sure to click inside the procedure you want to run. When in an Excel sheet, click the Developer tab, click Macros in the Code group, choose the procedure in the resulting dialog shown in **Figure A**, and then click Run.

Figure A



The bare bones VBA code

A simple list of sheet names is easy to generate using VBA thanks to the Worksheets collection. **Listing A** shows a simple For Each loop that cycles through this collection. For each sheet, the code uses the Name property to enter that name in column A, beginning at A1, in Sheet1.

Listing A

```
Sub ListSheetNames()

'List all sheet names in column A of Sheet1.

'Update to change location of list.

Sheets("Sheet1").Activate

ActiveSheet.Cells(1, 1).Select

'Generate a list of hyperlinks to each sheet in workbook in Sheet1.

For Each sh In Worksheets

ActiveCell = sh.Name

ActiveCell.Offset(1, 0).Select 'Move down a row.

Next

End Sub
```

When adapting this code, you might want to change the location of the list; do so by changing the first two lines accordingly. The For Each loop also offers possibilities for modifying. You might want to add header names or values to number the sheet names.

This code will list hidden and very hidden sheets, which you might not want. When that's the case, you'll need to check the Visible properties xlSheetVisible and xlSheetVeryHidden. In addition, because we're actively selecting A1 on Sheet1, the cursor moves to that location. If you don't want the active cell to change, use implicit selection statements. To learn more about implicit and explicit references, read Excel tips: How to select cells and ranges efficiently using VBA.

There are many modifications you might want to make. For example, instead of an ordinary list of text, you might want a list of hyperlinks.

How to generate a list of hyperlinks in Excel

It's not unusual for a complex workbook to include a list of hyperlinks to each sheet in the workbook. The procedure you'll use, shown in **Listing B**, is similar to **Listing A**, but this code uses the Name property to create a hyperlink.

Listing B

```
Sub ListSheetNamesAsHyperlinks()

'Generate list of hyperlinks to each sheet in workbook in Sheet1, A1.

Sheets("Sheet1").Activate

ActiveSheet.Cells(1, 1).Select

'Generate a list of hyperlinks to each sheet in workbook in Sheet1.

For Each sh In Worksheets

ActiveSheet.Hyperlinks.Add Anchor:=Selection, _

Address:="", SubAddress:="'" &amp;amp; sh.Name &amp;amp; "'" &amp;amp; "!A1", _

TextToDisplay:=sh.Name

ActiveCell.Offset(1, 0).Select 'Moves down a row

Next

End Sub
```

The first two lines select cell A1 in Sheet1. Update these two statements to relocate the list. The For Each loop cycles through all the sheets, using the Name property to create a hyperlink for each sheet.

The Hyperlinks.Add property in the For Each uses the form

.Add Anchor, Address, [SubAddress], [ScreenTip], [TextToDisplay]

The parameter information is listed below:

- Anchor: A Range or Shape object.
- Address: The address of the hyperlink.
- SubAddress: The subaddress of the hyperlink
- ScreenTip: Information displayed when the mouse pointer pauses over the hyperlink.
- TextToDisplay: The hyperlink text.

Our procedure's SubAddress argument

SubAddress:="'" & sh.Name & "'" & "!A1"

builds a string using the current sheet's name and the cell reference A1. For instance, if the current sheet is Sheet1, this evaluates to 'Sheet1'!A1. Subsequently, when you click this hyperlink, it takes you to cell A1 on Sheet1. As before, you can easily modify this procedure to reflect the way you want to use this list.

Both sub procedures are easy to use. Both can be easily modified to change the list's position or to add more information to the simple list.

By Susan Harkins

Susan Sales Harkins is an IT consultant, specializing in desktop solutions. Previously, she was editor in chief for The Cobb Group, the world's largest publisher of technical journals.

MICROSOFT SOFTWARE