Making Everything Easier!™

4th Edition

Excel® VBA Programming

FOR DUMMIES®

A Wiley Brand

Learn to:

- Use the essential tools and operations for VBA
- Get the most out of Excel's macro recording feature
- Handle errors and eliminate bugs in your code
- Develop custom worksheet functions and user-oriented applications

John Walkenbach

Excel® VBA Programming

FOR DUMMIES®
A Wiley Brand

4th Edition

by John Walkenbach

FOR DUMMIES®
A Wiley Brand

Excel® VBA Programming For Dummies®, 4th Edition

Published by: **John Wiley & Sons, Inc.,** 111 River Street, Hoboken, NJ 07030-5774, www.wiley.com

Published simultaneously in Canada

LIMIT OF LIABILITY/DISCLAIMER OF WARRANTY: THE PUBLISHER AND THE AUTHOR MAKE NO REPRESENTATIONS OR WARRANTIES WITH RESPECT TO THE ACCURACY OR COMPLETENESS OF THE CONTENTS OF THIS WORK AND SPECIFICALLY DISCLAIM ALL WARRANTIES, INCLUDING WITHOUT LIMITATION WARRANTIES OF FITNESS FOR A PARTICULAR PURPOSE. NO WARRANTY MAY BE CREATED OR EXTENDED BY SALES OR PROMOTIONAL MATERIALS. THE ADVICE AND STRATEGIES CONTAINED HEREIN MAY NOT BE SUITABLE FOR EVERY SITUATION. THIS WORK IS SOLD WITH THE UNDERSTANDING THAT THE PUBLISHER IS NOT ENGAGED IN RENDERING LEGAL, ACCOUNTING, OR OTHER PROFESSIONAL SERVICES. IF PROFESSIONAL ASSISTANCE IS REQUIRED, THE SERVICES OF A COMPETENT PROFESSIONAL PERSON SHOULD BE SOUGHT. NEITHER THE PUBLISHER NOR THE AUTHOR SHALL BE LIABLE FOR DAMAGES ARISING HEREFROM. THE FACT THAT AN ORGANIZATION OR WEBSITE IS REFERRED TO IN THIS WORK AS A CITATION AND/OR A POTENTIAL SOURCE OF FURTHER INFORMATION DOES NOT MEAN THAT THE AUTHOR OR THE PUBLISHER ENDORSES THE INFORMATION THE ORGANIZATION OR WEBSITE MAY PROVIDE OR RECOMMENDATIONS IT MAY MAKE. FURTHER, READERS SHOULD BE AWARE THAT INTERNET WEBSITES LISTED IN THIS WORK MAY HAVE CHANGED OR DISAPPEARED BETWEEN WHEN THIS WORK WAS WRITTEN AND WHEN IT IS READ. FULFILLMENT OF EACH COUPON OFFER IS THE SOLE RESPONSIBILITY OF THE OFFEROR.

For general information on our other products and services, please contact our Customer Care Department within the U.S. at 877-762-2974, outside the U.S. at 317-572-3993, or fax 317-572-4002. For technical support, please visit www.wiley.com/techsupport.

Wiley publishes in a variety of print and electronic formats and by print-on-demand. Some material included with standard print versions of this book may not be included in e-books or in print-on-demand. If this book refers to media such as a CD or DVD that is not included in the version you purchased, you may download this material at http://booksupport.wiley.com. For more information about Wiley products, visit www.wiley.com.

Library of Congress Control Number: 2015950121

ISBN 978-1-119-07739-8 (pbk); ISBN 978-1-119-07741-1 (ebk); ISBN 978-1-119-07744-2 (ebk)

Manufactured in the United States of America

10 9 8 7 6 5 4 3 2 1

Contents at a Glance

Table of Contents

Chapter 18: UserForm Techniques and Tricks 295

Chapter 19: Accessing Your Macros through the User Interface . . .321

Part V: Putting It All Together 333

Chapter 20: Creating Worksheet Functions — and Living to Tell about It 335

Chapter 21: Creating Excel Add-Ins 353

Introduction

Greetings, prospective Excel programmer. . . .

Thanks for buying my book. I think you'll find that it offers a fast, enjoyable way to discover the ins and outs of Microsoft Excel programming. Even if you don't have the foggiest idea of what programming is all about, this book can help you make Excel jump through hoops in no time. (Well, it will take *some* time.)

Unlike most programming books, this one is written in plain English, and even normal people can understand it. Even better, it's filled with information of the "just the facts, ma'am" variety — not the drivel you might need once every third lifetime.

About this Book

Go to any large bookstore (in-person or online), and you'll find many Excel books (far too many, as far as I'm concerned). A quick overview can help you decide whether this book is really right for you. This book

- Is designed for intermediate to advanced Excel users who want to get up to speed with Visual Basic for Applications (VBA) programming.
- Requires no previous programming experience.
- Covers the most commonly used commands.
- Is appropriate for Excel 2013 or Excel 2016.
- Just might make you crack a smile occasionally.

If you're using Excel 2000, XP, or 2003, this book is not for you. If you're using Excel 2007 or 2010, it *might* be okay, but some things have changed. You'd probably be better off with the previous edition.

Oh, yeah — this is *not* an introductory Excel book. If you're looking for a general-purpose Excel book, check out either of the following books, which are all published by Wiley:

- *Excel 2016 For Dummies,* by Greg Harvey
- *Excel 2016 Bible,* by John Walkenbach (yep, that's me)

These books are also available in editions for earlier versions of Excel.

Notice that the title of this book isn't *The Complete Guide to Excel VBA Programming For Dummies.* I don't cover all aspects of Excel programming — but then again, you probably don't want to know *everything* about this topic. If you consume this book and find that you're hungry for a more comprehensive Excel programming book, you might try *Microsoft Excel 2016 Power Programming with VBA,* by John Walkenbach, also published by Wiley. And yes, editions for older versions of Excel are also available.

Obligatory Typographical Conventions Section

All computer books have a section like this. (I think some federal law requires it.) Read it or skip it.

Sometimes, I refer to key combinations — which means you hold down one key while you press another. For example, Ctrl+Z means you hold down the Ctrl key while you press Z.

For menu commands, I use a distinctive character to separate items on the Ribbon or menu. For example, you use the following command to create a named range in a worksheet:

Formulas ➪ Defined Names ➪ Define Name

Formulas is the tab at the top of the Ribbon, Defined Names is the Ribbon group, and Define Name is the actual command.

The Visual Basic editor still uses old-fashioned menus and toolbars. So I might tell you to choose Tools ➪ Options. That means choose the Tools menu and then choose the Options menu item.

Excel programming involves developing *code* — that is, the instructions Excel follows. All code in this book appears in a monospace font, like this:

```
Range("A1:A12").Select
```

Some long lines of code don't fit between the margins in this book. In such cases, I use the standard VBA line-continuation character sequence: a space followed by an underscore character. Here's an example:

```
Selection.PasteSpecial Paste:=xlValues, _
   Operation:=xlNone, SkipBlanks:=False, _
   Transpose:=False
```

When you enter this code, you can type it as written or place it on a single line (omitting the space and underscore combination).

Check Your Security Settings

It's a cruel world out there. It seems that some scam artist is always trying to take advantage of you or cause some type of problem. The world of computing is equally cruel. You probably know about computer viruses, which can cause some nasty things to happen to your system. But did you know that computer viruses can also reside in an Excel file? It's true. In fact, it's relatively easy to write a computer virus by using VBA. An unknowing user can open an Excel file and spread the virus to other Excel workbooks and to other systems.

Over the years, Microsoft has become increasingly concerned about security issues. This is a good thing, but it also means that Excel users need to understand how things work. You can check Excel's security settings by choosing the File ➪ Options ➪ Trust Center ➪ Trust Center Settings command. There is a plethora of options in there, and people have been known to open that dialog box and never be heard from again.

If you click the Macro Settings tab (on the left side of the Trust Center dialog box), your options are as follows:

- **Disable all macros without notification.** Macros will not work, regardless of what you do.
- **Disable all macros with notification.** When you open a workbook with macros, you see the Message Bar open with an option you can click to enable macros, or (if the Visual Basic Editor window is open) you get a message asking if you want to enable macros.
- **Disable all macros except digitally signed macros.** Only macros with a digital signature are allowed to run (but even for those signatures you haven't marked as trusted, you still get the security warning).
- **Enable all macros.** All macros run with no warnings. This option is not recommended because potentially dangerous code can be executed.

Consider this scenario: You spend a week writing a killer VBA program that will revolutionize your company. You test it thoroughly and then send it to your boss. He calls you into his office and claims that your macro doesn't do anything at all. What's going on? Chances are, your boss's security setting does not allow macros to run. Or maybe he chose to go along with Microsoft's default suggestion and disable the macros when he opened the file.

Bottom line? Just because an Excel workbook contains a macro, it is no guarantee that the macro will ever be executed. It all depends on the security setting and whether the user chooses to enable or disable macros for that file.

To work with this book, you need to enable macros for the files you work with. My advice is to use the second security level. Then, when you open a file that you've created, you can simply enable the macros. If you open a file from someone you don't know, you should disable the macros and check the VBA code to ensure that it doesn't contain anything destructive or malicious. Usually, it's pretty easy to identify suspicious VBA code.

Another option is to designate a trusted folder. Choose File ➪ Options ➪ Trust Center ➪ Trust Center Settings. Select the Trusted Locations option and then designate a particular folder to be a trusted location. Store your trusted workbooks there, and Excel won't bug you about enabling macros. For example, if you download the sample files for this book, you can put them in a trusted location.

Foolish Assumptions

People who write books usually have a target reader in mind. For this book, my target reader is a conglomerate of dozens of Excel users I've met over the years (either in person or out in cyberspace). The following points more or less describe my hypothetical target reader:

- You have access to a PC at work — and probably at home. And those computers are connected to the Internet.
- You're running Excel 2013 or Excel 2016.
- You've been using computers for several years.
- You use Excel frequently in your work, and you consider yourself to be more knowledgeable about Excel than the average bear.
- You need to make Excel do some things that you currently can't make it do.
- You have little or no programming experience.
- You understand that the Help system in Excel can actually be useful. Face it — this book doesn't cover everything. If you get on good speaking terms with the Help system, you'll be able to fill in some of the missing pieces.
- You need to accomplish some work, and you have a low tolerance for thick, boring computer books.

Icons Used in This Book

TIP

Don’t skip information marked with this icon. It identifies a shortcut that can save you lots of time (and maybe even allow you to leave the office at a reasonable hour).

This icon is also used to let you know that the code being discussed is available on the web. Download it to eliminate lots of typing. See “Beyond the Book” for more information.

REMEMBER

This icon tells you when you need to store information in the deep recesses of your brain for later use.

TECHNICAL STUFF

This icon flags material that you might consider technical. You may find it interesting, but you can safely skip it if you’re in a hurry.

WARNING!

Read anything marked with this icon. Otherwise, you may lose your data, blow up your computer, cause a nuclear meltdown — or maybe even ruin your whole day.

Beyond the Book

Hungry for more?

- **Sample files:** This book has its very own website where you can download the example files. To get these files, point your web browser to

  ```
  http://dummies.com/extras/excelvbaprogramming
  ```

  Please note that this URL is case-sensitive and uses all lowercase letters. If you don’t type it exactly, it won’t work.

  Having the sample files will save you a lot of typing. Better yet, you can play around with them and experiment with various changes. In fact, I highly recommend playing around with these files. Experimentation is the best way to master VBA.

- **Cheat Sheet:** The Cheat Sheet is a handy list of common VBA statements for Excel programming. You can find it at

  ```
  http://dummies.com/extras/excelvbaprogramming
  ```

- **Updates:** Occasionally, we have updates to our technology books. If this book does have technical updates, they will be posted at

  ```
  http://dummies.com/extras/excelvbaprogramming
  ```

Again, this URL is case-sensitive.

Where to Go from Here

Reading this introduction was your first step. Now it's time to move on and become a programmer. (There's that *p* word again!)

If you're a programming virgin, I strongly suggest that you start with Chapter 1 and progress through the book until you've discovered enough to do what you want to do. Chapter 2 gives you some immediate hands-on experience, so you have the illusion that you're making quick progress.

But it's a free country (at least, it was when I wrote these words); I won't sic the Computer Book Police on you if you opt to thumb through randomly and read whatever strikes your fancy.

I hope you have as much fun reading this book as I did writing it.

Part I

Getting Started with Excel VBA Programming

Visit `www.dummies.com` to explore more great Dummies content online.

In this part . . .

- ✓ Get to know Visual Basic for Applications.
- ✓ See examples of some of the things you can do with VBA.
- ✓ Travel back in time and see what Excel was like in the olden days.
- ✓ Work through a real-live Excel programming session.
- ✓ Get a handle on how Excel deals with macro security.

Chapter 1

What Is VBA?

In This Chapter

- Getting a conceptual overview of VBA
- Finding out what you can do with VBA
- Discovering the advantages and disadvantages of using VBA
- Getting the lowdown on what VBA is
- Taking a mini lesson on the history of Excel

If you're eager to jump into VBA programming, hold your horses. This chapter is completely devoid of any hands-on training material. It does, however, contain some essential background information that assists you in becoming an Excel programmer. In other words, this chapter paves the way for everything else that follows and gives you a feel for how Excel programming fits into the overall scheme of the universe. It's not as boring as you might think, so please try to resist the urge to jump to Chapter 2.

Okay, So What Is VBA?

VBA, which stands for *Visual Basic for Applications,* is a programming language developed by Microsoft — you know, the company that tries to get you to buy a new version of Windows every few years. Excel, along with the other members of Microsoft Office, includes the VBA language (at no extra charge). In a nutshell, VBA is the tool that people like you and me use to develop programs that control Excel.

Imagine an intelligent robot that knows all about Excel. This robot can read instructions, and it can also operate Excel very fast and accurately. When you want the robot to do something in Excel, you write up a set of robot instructions by using special codes. Then you tell the robot to follow your instructions while you sit back and drink a glass of lemonade. That's kind of what VBA is all about — a code language for robots. Note, however, that Excel does not come with a robot or lemonade.

TECHNICAL STUFF

A few words about terminology

Excel programming terminology can be a bit confusing. For example, VBA is a programming language, but it also serves as a macro language. What do you call something written in VBA and executed in Excel? Is it a *macro,* or is it a *program?* Excel's Help system often refers to VBA procedures as *macros,* so I use that terminology. But I also call this stuff a *program.*

I use the term *automate* throughout this book. This term means that a series of steps are completed automatically. For example, if you write a macro that adds color to some cells, prints the worksheet, and then removes the color, you have *automated* those three steps.

By the way, *macro* does not stand for **M**essy **A**nd **C**onfusing **R**epeated **O**peration. Rather, it comes from the Greek *makros,* which means large — which also describes your paycheck after you become an expert macro programmer.

What Can You Do with VBA?

You're probably aware that people use Excel for thousands of different tasks. Here are just a few examples:

- Analyzing scientific data
- Budgeting and forecasting
- Creating invoices and other forms
- Developing charts from data
- Keeping lists of things such as customers' names, students' grades, or holiday gift ideas (a nice fruitcake would be lovely)
- Yadda, yadda, yadda

The list could go on and on, but I think you get the idea. My point is simply that Excel is used for a wide variety of tasks, and everyone reading this book has different needs and expectations regarding Excel. One thing virtually every reader has in common is the *need to automate some aspect of Excel.* That, dear reader, is what VBA is all about.

For example, you might create a VBA program to import some numbers and then format and print your month-end sales report. After developing and testing the program, you can execute the macro with a single command, causing Excel to automatically perform many time-consuming procedures.

Rather than struggle through a tedious sequence of commands, you can click a button and then hop on over to Facebook and kill some time while your macro does the work.

In the following sections, I briefly describe some common uses for VBA macros. One or two of these may push your button.

Inserting a bunch of text

If you often need to enter your company name, address, and phone number in your worksheets, you can create a macro to do the typing for you. You can extend this concept as far as you like. For example, you might develop a macro that automatically types a list of all salespeople who work for your company.

Automating a task you perform frequently

Assume you're a sales manager and you need to prepare a month-end sales report to keep your boss happy. If the task is straightforward, you can develop a VBA program to do it for you. Your boss will be impressed by the consistently high quality of your reports, and you'll be promoted to a new job for which you are highly unqualified.

Automating repetitive operations

If you need to perform the same action on, say, 12 different Excel workbooks, you can record a macro while you perform the task on the first workbook and then let the macro repeat your action on the other workbooks. The nice thing about this is that Excel never complains about being bored. Excel's macro recorder is similar to recording live action on a video recorder. But it doesn't require a camera, and the battery never needs to be recharged.

Creating a custom command

Do you often issue the same sequence of Excel menu commands? If so, save yourself a few seconds by developing a macro that combines these commands into a single custom command, which you can execute with a single keystroke or button click. You probably won't save *that* much time, but you'll probably be more accurate. And the guy in the next cubicle will be really impressed.

Creating a custom button

You can customize your Quick Access toolbar with your own buttons that execute the macros you write. Office workers tend to be very impressed by buttons that perform magic. And if you *really* want to impress your fellow employees, you can even add new buttons to the Ribbon.

Developing new worksheet functions

Although Excel includes hundreds of built-in functions (such as SUM and AVERAGE), you can create *custom* worksheet functions that can greatly simplify your formulas. I guarantee you'll be surprised by how easy this is. (I show you how to do this in Chapter 20.) Even better, the Insert Function dialog box displays your custom functions, making them appear built-in. Very snazzy stuff.

Creating custom add-ins for Excel

You're probably familiar with some of the add-ins that ship with Excel. For example, the Analysis ToolPak is a popular add-in. You can use VBA to develop your own special-purpose add-ins. I developed my Power Utility Pak add-in by using only VBA, and people all around the world pay me real money so they can use it.

Creating complete, macro-driven applications

If you're willing to spend some time, you can use VBA to create large-scale applications complete with a custom Ribbon tab, dialog boxes, onscreen help, and lots of other accoutrements. This book doesn't go quite that far, but I'm just telling you this to impress you with how powerful VBA really is.

Advantages and Disadvantages of VBA

In this section, I briefly describe the good things about VBA — and I also explore its darker side.

VBA advantages

You can automate almost anything you do in Excel. To do so, you write instructions that Excel carries out. Automating a task by using VBA offers several advantages:

- Excel always executes the task in exactly the same way. (In most cases, consistency is a good thing.)
- Excel performs the task much faster than you can do it manually (unless, of course, you're Clark Kent).
- If you're a good macro programmer, Excel always performs the task without errors (which probably can't be said about you or me).
- If you set things up properly, someone who doesn't know anything about Excel can perform the task by running the macro.
- You can do things in Excel that are otherwise impossible — which can make you a very popular person around the office.
- For long, time-consuming tasks, you don't have to sit in front of your computer and get bored. Excel does the work while you hang out at the water cooler.

VBA disadvantages

It's only fair that I give equal time to listing the disadvantages (or *potential* disadvantages) of VBA:

- You have to know how to write programs in VBA (but that's why you bought this book, right?). Fortunately, it's not as difficult as you might expect.
- Other people who need to use your VBA programs must have their own copies of Excel. It would be nice if you could press a button that transforms your Excel/VBA application into a stand-alone program, but that isn't possible (and probably never will be).

- Sometimes, things go wrong. In other words, you can't blindly assume that your VBA program will always work correctly under all circumstances. Welcome to the world of debugging and, if others are using your macros, technical support.
- VBA is a moving target. As you know, Microsoft is continually upgrading Excel. Even though Microsoft puts great effort into compatibility between versions, you may discover that the VBA code you've written doesn't work properly with older versions or with a future version of Excel.

VBA in a Nutshell

Just to let you know what you're in for, I've prepared a quick-and-dirty summary of what VBA is all about. Of course, I describe all this stuff in semi-excruciating detail later in the book.

- **You perform actions in VBA by writing (or recording) code in a VBA module.** You view and edit VBA modules by using the Visual Basic Editor (VBE).
- **A VBA module consists of Sub procedures.** A Sub procedure has nothing to do with underwater vessels or tasty sandwiches. Rather, it's a chunk of computer code that performs some action on or with objects (discussed in a moment). The following example shows a simple Sub procedure called AddEmUp. This amazing program, when executed, displays the result of 1 plus 1:

  ```
  Sub AddEmUp()
      Sum = 1 + 1
      MsgBox "The answer is " & Sum
  End Sub
  ```

  A Sub procedure that doesn't perform properly is said to be substandard.
- **A VBA module can also have Function procedures.** A Function procedure returns a single value. You can call it from another VBA procedure or even use it as a function in a worksheet formula. An example of a Function procedure (named AddTwo) follows. This Function accepts two numbers (called arguments) and returns the sum of those values:

  ```
  Function AddTwo(arg1, arg2)
      AddTwo = arg1 + arg2
  End Function
  ```

  A Function procedure that doesn't work correctly is said to be dysfunctional.