LEARNING MADE EASY

6th Edition

Excel® VBA Programming

for dummies®

A Wiley Brand

Work faster and smarter by automating with VBA

Create applications and be the office hero

Learn VBA to get the most out of Excel

Dick Kusleika

Excel® VBA Programming

6th Edition

by Dick Kusleika

for dummies®
A Wiley Brand

Excel® VBA Programming For Dummies®

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

Published simultaneously in Canada

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 https://hub.wiley.com/community/support/dummies.

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: 2021951691

ISBN 978-1-119-84307-8 (pbk); ISBN 978-1-119-84308-5 (ebk); ISBN 978-1-119-84309-2 (ebk)

Contents at a Glance

Table of Contents

Introduction

Greetings, prospective Excel programmer. . .

You no doubt have your reasons for picking up a book on VBA programming. Maybe you got a new job (congratulations). Maybe you're trying to automate some of the repetitive data crunching tasks you have to do. Maybe you're just a nerd at heart. Whatever the reason, thank you for choosing this book.

Inside, you find everything you need to get up and running with VBA fast. Even if you don't have the foggiest idea of what programming is all about, this book can help. Unlike most programming books, this one is filled with information designed to include just what you need to know to quickly ramp your VBA programming skillset.

About This Book

Go to any large bookstore (in person or online), and you'll find many Excel books. 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 recent versions of Excel.
- Just might make you crack a smile occasionally.

If you're using an older version of Excel, this book *might* be okay, but some things have changed. You'd probably be better off with the preceding 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 both published by Wiley:

- *Excel 2019 For Dummies,* by Greg Harvey
- *Excel Bible,* by Michael Alexander and Dick Kusleika

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.* This book doesn'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 2019 Power Programming with VBA,* also published by Wiley. And yes, editions for older versions of Excel are also available.

To make the content more accessible, I divided this book into six parts:

- **Part 1, Starting with Excel VBA Programming**
- **Part 2, Employing VBA with Excel**
- **Part 3, Programming Concepts**
- **Part 4, Communicating with Your Users**
- **Part 5, Putting It All Together**
- **Part 6, The Part of Tens**

Typographical conventions

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 Ribbon tool you click.

The Visual Basic Editor still uses old-fashioned menus and toolbars. So Tools➪ Options means choose the Tools menu and then choose the Options menu item.

Excel programming involves developing *code* — that is, the instructions VBA 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).

Macro security

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 File➪ Options➪ Trust Center➪ Trust Center Settings. 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 VBA macros without notification.** Macros will not work, regardless of what you do.
- **Disable VBA 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 VBA 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 VBA 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. They call you into their office and claim that your macro doesn't do anything at all. What's going on? Chances are, your boss's security setting doesn't allow macros to run. Or maybe they chose to go along with Microsoft's default suggestion and disable the macros when they opened the file.

Bottom line? Just because an Excel workbook contains a macro does not 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 as 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. The following points more or less describe the hypothetical target reader for this book:

- You have access to a PC at work — and probably at home. And those computers are connected to the internet.
- You're running a fairly recent version of Excel.
- 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

Throughout this book, icons in the margins highlight certain types of valuable information that call out for your attention. Here are the icons you'll encounter and a brief description of each.

The Tip icon marks tips and shortcuts that can save you a great deal of time (and maybe even allow you to leave the office at a reasonable hour).

Remember icons mark the information that's especially important to know. To siphon off the most important information in each chapter, just skim through these icons.

The Technical Stuff icon marks information of a highly technical nature that you can normally skip over.

WARNING

The Warning icon tells you to watch out! It marks important information that may save you from losing data and ruining your whole day.

Beyond the Book

This book has its very own website where you can download the sample files. To get these files, point your web browser to

`https://www.dummies.com/go/excelvbaprogrammingfd6e`

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, experimentation is the best way to master VBA.

In addition, this book comes with a free access-anywhere Cheat Sheet that includes keyboard shortcuts related to Excel VBA programming. To get this Cheat Sheet, simply go to `www.dummies.com` and type **VBA Excel Programming For Dummies Cheat Sheet** in the Search box and click on the Cheat Sheets tab.

Where to Go from Here

This book contains everything you need to learn VBA programming at a mid-advanced level. The book starts off with the basics of recording macros and builds, chapter by chapter.

If you're completely new to Excel macros, start with Part 1 to get a refresher on the fundamentals of recording macros. If you have experience recording macros, but want to better understand the VBA behind them, read to Parts 2 and 3. There, you gain a concise understanding of how VBA works, along with the basic foundation you need to implement your own code.

Finally, if you're familiar with programming concepts and just want to get a quick run-through of some of the more advanced techniques like creating your custom functions and add-ins, feel free to jump to Part 4.

1 Starting Excel VBA Programming

IN THIS PART . . .

Get to know Visual Basic for Applications.

Work through a real-live Excel programming session.

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
- Staying Excel compatible

Chapter 1
Getting to Know VBA

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. Just like a great heart surgeon has to take freshman biology, you must learn some basic concepts and terminology so that the more practical aspects you do later make sense.

Understanding VBA Basics

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 use to write programs that automate Excel, such as a program to format a spreadsheet into a monthly report. In the next section, I discuss in more detail the types of tasks you can automate with VBA.

Imagine a robot that knows all about Excel. This robot can read instructions, and it can also operate Excel quickly and accurately. When you want the robot to do something in Excel, you write 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. With VBA, you don't need the extra chair at your desk for an actual robot. Just feed the special codes into VBA and it does the work.

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. In this context, a *macro* is a set of instructions Excel performs to imitate keystrokes and mouse actions. 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 that terminology is used in this book. But you can also call this stuff a *program.*

You'll see 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* doesn't 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.

Knowing What VBA Can Do

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
- Building charts from data
- Keeping lists of things such as customers' names, students' grades, or holiday gift ideas (a nice fruitcake would be lovely)

The list could go on and on, but you get the idea. The 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 writing 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 simply click a button and then hop on over to TikTok and kill some time while your macro does the work.

The following sections 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 enters 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. However, 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 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 sitting 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. You'll be surprised by how easy this is. (You can explore 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 might be 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. Add-ins are just like regular Excel workbooks except they don't have any worksheets the users can see. They're all VBA and they usually automate tasks on other workbooks.

Getting the Most from VBA

VBA provides a ton of benefits that are tempered with a few disadvantages you'll want to keep in mind. One advantage of VBA, for example, is that you can use it to automate almost anything you do in Excel. All you have to do is write instructions in VBA, and Excel carries out those instructions when instructed to do so. The previous section, "Knowing What VBA Can Do," describes some specific tasks you can accomplish by using VBA. The following sections helps you decide whether VBA is the best tool to achieve the results you want.

Knowing what VBA does best

Here are some benefits of automating a task by using VBA:

- 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, no matter how careful you are).
- If you set up things 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.

Recognizing the disadvantages of using VBA

Using VBA can present some disadvantages (or *potential* disadvantages). Understanding these limitations upfront helps you avoid running down rabbit trails to nowhere. Most people use VBA to save time, not waste it!

Keep the following points in mind when deciding whether to use VBA:

- 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* (fixing errors) and, if others are using your macros, providing 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 might discover that the VBA code you've written doesn't work properly with older versions or with a future version of Excel. For more information on the importance of Excel compatibility, see the section "Ensuring Excel Compatibility," later in this chapter.

Understanding VBA Concepts

Just to let you know what you're in for, here's a quick-and-dirty summary of what VBA is all about.

- **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.*

- **VBA manipulates objects.** Excel provides dozens and dozens of objects that you can manipulate. Examples of objects include a workbook, a worksheet, a cell range, a chart, and a shape. You have many more objects at your disposal, and you can manipulate them by using VBA code.
- **Objects are arranged in a hierarchy.** Objects can act as *containers* for other objects. At the top of the object hierarchy is Excel. Excel itself is an object called *Application*. The Application object contains other objects, such as Workbook objects and Add-In objects. The Workbook object can contain other objects, such as Worksheet objects and Chart objects. A Worksheet object can contain objects such as Range objects and PivotTable objects. The term *object model* refers to the arrangement of these objects. (Object-model mavens can find out more in Chapter 4.)