PART I
Understanding the BASICs

- **LESSON 1:** Introducing VBA
- **LESSON 2:** Getting Started with Macros
- **LESSON 3:** Introducing the Visual Basic Editor
- **LESSON 4:** Working in the VBE

1

Introducing VBA

Welcome to your first lesson in *Excel VBA 24-Hour Trainer*! A good place to start is at the beginning, where you'll find it useful to get an understanding of where Visual Basic for Applications (VBA) comes from and what VBA is today. After you get a feel for how VBA fits into the overall Excel universe, you find out how to use VBA to manipulate Excel in ways you might never have thought possible.

WHAT IS VBA?

VBA is a programming language created by Microsoft to automate operations in applications that support it, such as Excel. VBA is an enormously powerful tool that enables you to control Excel in countless ways that you cannot do—or would not want to do—manually.

In fact, VBA is also the language that manipulates Microsoft Office applications in Access, Word, PowerPoint, and Outlook. For the purposes here, VBA is the tool you use to develop macros and manipulate objects to control Excel and to control other Office applications from Excel.

You do not need to purchase anything more than the Office suite (or the individual application) to also own VBA. If you have Excel on your computer, you have VBA on your computer.

WHAT IS A "MACRO," ANYWAY?

Back in the day, a programming language was often called a "macro language" if its capabilities included the automation of a sequence of commands in spreadsheet or word-processing applications. With Microsoft's release of Office 5, VBA set a new bar for how robust a programming language can be, with capabilities extending far beyond those of earlier programming languages, such as the ability to create and control objects within Excel or to have access to disk drives and networks.

continues

continued

So VBA is a programming language, and it is also a macro language. Confusion of terminology arises when referring to VBA code that is a series of commands written and executed in Excel. Is it a macro, a procedure, or a program? Microsoft commonly refers to its VBA procedures as macros, so that's good enough for me to call them macros also. Outside of a few exceptions that I explain when the time comes, I refer to VBA procedures as macros.

A BRIEF HISTORY OF VBA

VBA is a present-day dialect of the BASIC (Beginner's All-purpose Symbolic Instruction Code) programming language that was developed in the 1960s. BASIC became widely used in many software applications throughout the next two decades because it was easy to learn and understand.

Over the years, BASIC has evolved and improved in response to advancing technology and increased demands by its users for greater programming flexibility. In 1985, Microsoft released a much richer version of BASIC, named QuickBASIC, which boasted the most up-to-date features found in programming languages of the day. In 1992, Microsoft released Visual Basic for Windows, designed to work within the burgeoning Windows environment.

Meanwhile, various software publishers were making their own enhancements to BASIC for their products' programming languages, resulting in a wide and confusing range of functionality and commands among software applications that were using BASIC. Microsoft recognized the need for developing a standardized programming language for its software products, and created Visual Basic for Applications.

VBA was first released by Microsoft with Excel 5 in the Office 1995 suite. Since then, VBA has become the programming language for Microsoft's other popular Office applications, as well as for external software customers of Microsoft to whom VBA has been licensed for use.

THERE'S A BIG DIFFERENCE BETWEEN VB AND VBA!

With all the acronyms bandied about in the world of computing, it's easy to get some terms confused. VB stands for Visual Basic, and it is not the same as VBA. Though both VB and VBA are programming languages derived from BASIC and created by Microsoft, they are otherwise very different.

VB is a language that enables you to create standalone executable applications that do not even require its users to have Office or Excel loaded onto their computers. VBA cannot create standalone applications, and it exists within a host application such as Excel and the workbook containing the VBA code. For a VBA macro to run, its host application workbook must be open. This book is about VBA and how it controls Excel.

WHAT VBA CAN DO FOR YOU

Everyone reading this book uses Excel for their own needs, such as financial budgeting, forecasting, analyzing scientific data, creating invoices, or charting the progress of their favorite football team. One thing all readers have in common is the need to automate some kind of frequently encountered task that is either too time-consuming or too cumbersome to continue doing manually. That's where VBA comes in.

The good news is that utilizing VBA does not mandate that you first become a world-class professional programmer. Many VBA commands are at your disposal, and are relatively easy to implement and customize for your everyday purposes.

Anything you can do manually you can do with VBA, but VBA enables you to do it faster and with a minimized risk of human error. Many things that Excel does not allow you to do manually, you can do with VBA. The following sections describe a handful of examples of what VBA can do for you.

Automating a Recurring Task

If you find yourself needing to produce weekly or monthly sales and expense reports, a macro can create them in no time flat, in a style and format you (and more importantly, your boss) will be thrilled with. And if the source data changes later that day and you need to produce the updated report again, no problem—just run the macro again!

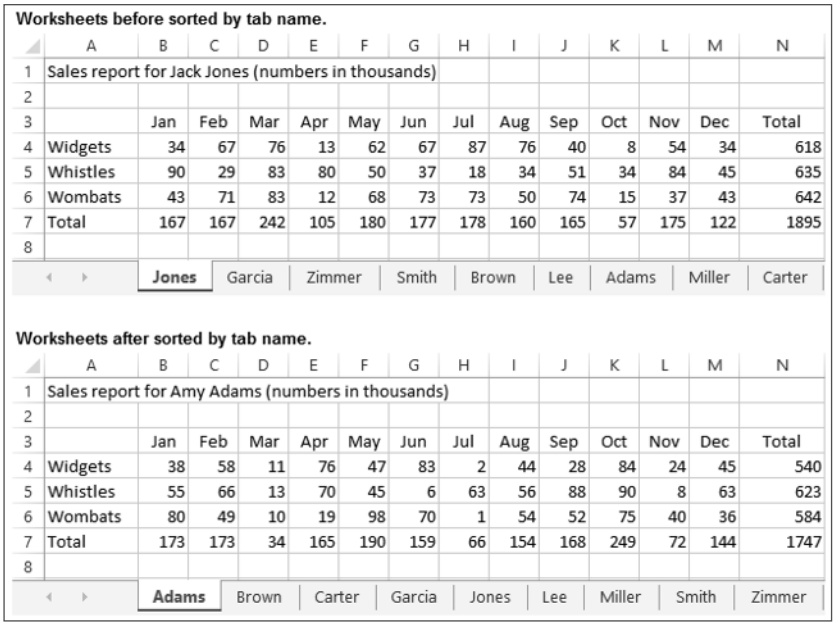
Automating a Repetitive Task

When faced with needing to perform the same task on every worksheet in your workbook, or in every workbook in a particular file folder, you can create a macro to "loop" through each object and do the deed. You find out how to repeat actions with various looping methods in Lesson 10. Figure 1-1 shows an example of worksheets that were sorted in alphabetical order by a macro that looped through each tab name, repositioning each sheet in the process.

Running a Macro Automatically if Another Action Takes Place

In some situations, you want a macro to run automatically so you don't have to worry about remembering to run it yourself. For example, to automatically refresh a pivot table the moment its source data changes, you can monitor those changes with VBA, ensuring that your pivot table always displays real-time results. This is called "event" programming, which is cool stuff, and is discussed in Lessons 13 and 14.

An event can also be triggered and programmed anytime a cell or range of cells is selected. A common request I've received from Excel users is to highlight the active cell, or the row and column belonging to the active cell, automatically when a cell is selected. Figure 1-2 shows three options to easily locate your active cell as you traverse your worksheet.



Worksheets before sorted by tab name.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	Sales report for Jack Jones (numbers in thousands)													
2														
3		Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec	Total
4	Widgets	34	67	76	13	62	67	87	76	40	8	54	34	618
5	Whistles	90	29	83	80	50	37	18	34	51	34	84	45	635
6	Wombats	43	71	83	12	68	73	73	50	74	15	37	43	642
7	Total	167	167	242	105	180	177	178	160	165	57	175	122	1895
8														

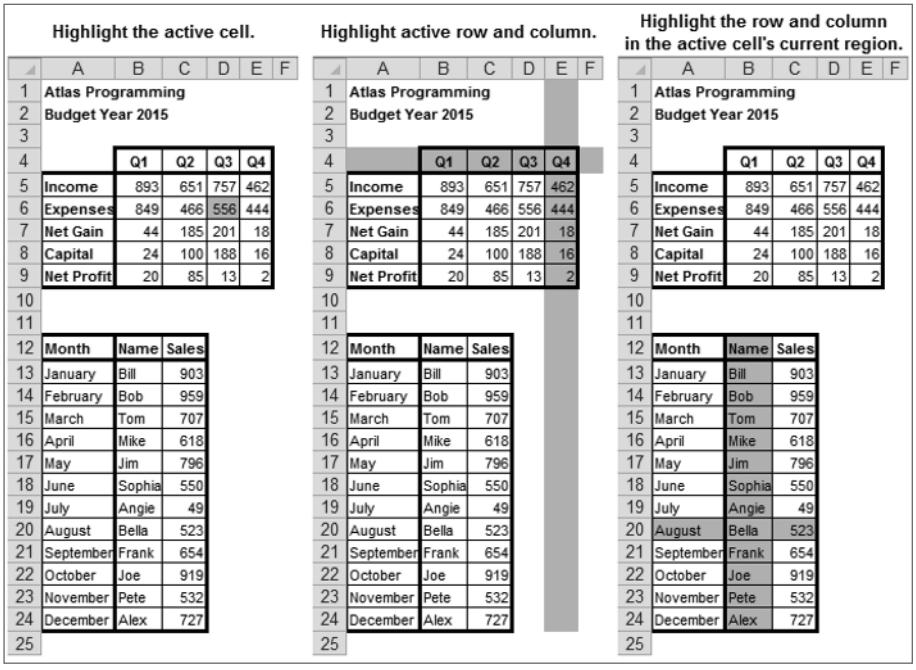
Jones | Garcia | Zimmer | Smith | Brown | Lee | Adams | Miller | Carter

Worksheets after sorted by tab name.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	Sales report for Amy Adams (numbers in thousands)													
2														
3		Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec	Total
4	Widgets	38	58	11	76	47	83	2	44	28	84	24	45	540
5	Whistles	55	66	13	70	45	6	63	56	88	90	8	63	623
6	Wombats	80	49	10	19	98	70	1	54	52	75	40	36	584
7	Total	173	173	34	165	190	159	66	154	168	249	72	144	1747
8														

Adams | Brown | Carter | Garcia | Jones | Lee | Miller | Smith | Zimmer

FIGURE 1-1



Highlight the active cell.

	A	B	C	D	E	F
1	Atlas Programming					
2	Budget Year 2015					
3						
4		Q1	Q2	Q3	Q4	
5	Income	893	651	757	462	
6	Expenses	849	466	556	444	
7	Net Gain	44	185	201	18	
8	Capital	24	100	188	16	
9	Net Profit	20	85	13	2	
10						
11						
12	Month	Name	Sales			
13	January	Bill	903			
14	February	Bob	959			
15	March	Tom	707			
16	April	Mike	618			
17	May	Jim	796			
18	June	Sophia	550			
19	July	Angie	49			
20	August	Bella	523			
21	September	Frank	654			
22	October	Joe	919			
23	November	Pete	532			
24	December	Alex	727			
25						

Highlight active row and column.

	A	B	C	D	E	F
1	Atlas Programming					
2	Budget Year 2015					
3						
4		Q1	Q2	Q3	Q4	
5	Income	893	651	757	462	
6	Expenses	849	466	556	444	
7	Net Gain	44	185	201	18	
8	Capital	24	100	188	16	
9	Net Profit	20	85	13	2	
10						
11						
12	Month	Name	Sales			
13	January	Bill	903			
14	February	Bob	959			
15	March	Tom	707			
16	April	Mike	618			
17	May	Jim	796			
18	June	Sophia	550			
19	July	Angie	49			
20	August	Bella	523			
21	September	Frank	654			
22	October	Joe	919			
23	November	Pete	532			
24	December	Alex	727			
25						

Highlight the row and column in the active cell's current region.

	A	B	C	D	E	F
1	Atlas Programming					
2	Budget Year 2015					
3						
4		Q1	Q2	Q3	Q4	
5	Income	893	651	757	462	
6	Expenses	849	466	556	444	
7	Net Gain	44	185	201	18	
8	Capital	24	100	188	16	
9	Net Profit	20	85	13	2	
10						
11						
12	Month	Name	Sales			
13	January	Bill	903			
14	February	Bob	959			
15	March	Tom	707			
16	April	Mike	618			
17	May	Jim	796			
18	June	Sophia	550			
19	July	Angie	49			
20	August	Bella	523			
21	September	Frank	654			
22	October	Joe	919			
23	November	Pete	532			
24	December	Alex	727			
25						

FIGURE 1-2

Creating Your Own Worksheet Functions

You can create your own worksheet functions, known as *user-defined functions*, to handle custom calculations that Excel's built-in functions do not provide, or would be too complicated to use even if such native functions were available. For example, later in the book you see how to add up numbers in cells that are formatted a certain color. UDFs, as these custom functions are called, are covered in Lesson 19, "User-Defined Functions."

Simplifying the Workbook's Look and Feel for Other Users

When you create a workbook for others to use, there will inevitably be users who know little to nothing about Excel, but who will still need to work in that file. You can build a customized interface with user-friendly menus and informational pop-up boxes to guide your novice users throughout their activities in the workbook. You might be surprised at how un-Excel-looking an Excel workbook can be, with VBA providing a visually comfortable and interactive experience for users unfamiliar with Excel, enabling them to get their work done. Figure 1-3 shows an example of accomplishing this with UserForms, which are discussed in Lessons 21, 22, and 23.

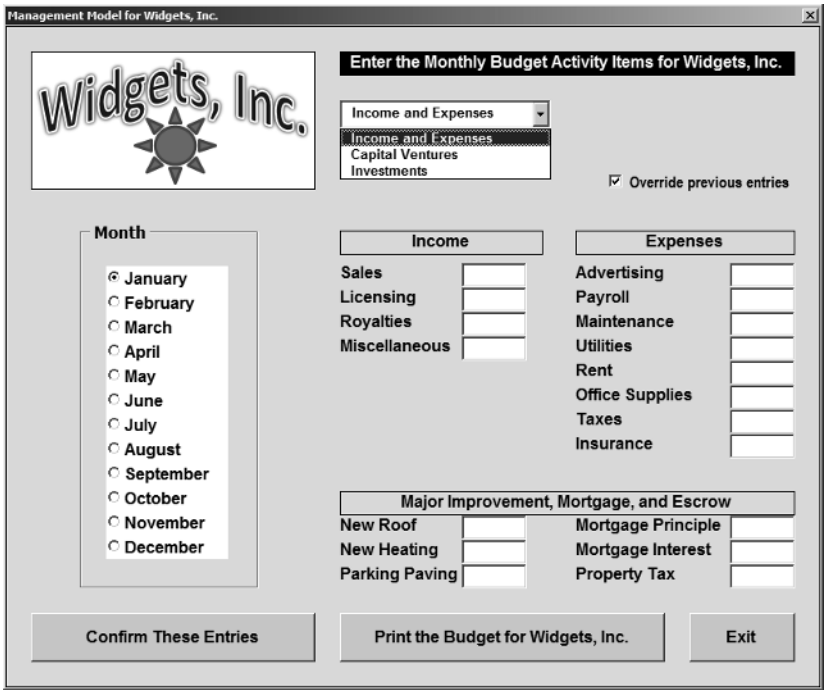


FIGURE 1-3

Controlling Other Office Applications from Excel

If you create narrative reports in Word that require an embedded list of data from Excel, or if you need to import a table from Access into an Excel worksheet, VBA can automate the process. VBA is the programming language for Microsoft's other Office applications, enabling you to write macros

in Excel to perform tasks in those other applications, with the users being none the wiser that they ever left Excel while the macro was running.

As you might imagine, the list of advantages to using VBA could fill the capacity of your average flash drive. The point is, you are sure to have tasks in your everyday dealings with Excel that can be accomplished more quickly and efficiently with VBA, and this book shows you how.

LIABILITIES OF VBA

Although VBA is a tremendously useful and versatile tool, it is not a 100 percent perfect programming language—but then, no programming language anywhere can truthfully claim infallibility. The pros of VBA far outweigh its cons, but learning and using VBA does come with a few objective caveats that you should be aware of:

- With each version release of Excel, Microsoft may add new VBA commands or stop supporting existing VBA commands, sometimes without advance warning. Surprises do happen, as was especially the case when Office 2007 was released with all its added features. Such is life in the world of Excel VBA. You will probably learn of coding errors from people who have upgraded to a newer version and are using the workbook you created in an earlier version.
- VBA does not run uniformly in all computer operating environments. Sometimes, no matter how extensively you test your code and how flawlessly the macros run on your computer as you develop a project, there will be users of your workbook who will eventually report an error in your code. It won't be your fault or VBA's fault, it's just the idiosyncrasies of how programming languages such as VBA mix with various operating systems, Office versions, and network configurations. Debugging your code is the subject of Lesson 20.
- Programming languages, including VBA, are not warmly received by all workplace IT departments. Many companies have set internal policies that forbid employees from downloading malicious software onto workplace computers. This is an understandable concern, but the corporate safety nets are sometimes cast far and wide to include Excel workbooks with VBA code. The tug of war in companies between the security interests of IT and the work efficiency needs of management can determine whether the VBA code you install will actually be allowed for use in some company venues.
- Finally, VBA is a large program. It has thousands of keywords and the language library is only getting larger. Actually, I see this as a good thing, because the more VBA you learn, the more productivity and control you will have with Excel. Just as with any language, be it spoken or programming, there is a level of rolling-up-your-shirtsleeves commitment that'll be needed to learn VBA. Even the longest journey starts with a first step, and this book gets you on your way.

> **NOTE** *VBA has a bright, stable future. An occasional rumor makes the rounds on the Internet, claiming the imminent demise of VBA. Do not believe it. VBA is here to stay, and Microsoft has publicly said so, time and again. The facts are, in 2007, Microsoft closed its VBA licensing program to new customers, and VBA was not supported in the 2008 version of Office for the Mac, though VBA has been supported by Mac versions after that. Microsoft has consistently made very clear its plan for supporting VBA in future versions of Excel for Windows.*

TRY IT

With the introductory nature of this first lesson, there's nothing specific to try with VBA. What you can do is to get a jump on the rest of the lessons in this book by making a list of some of your most frequent everyday manual Excel tasks, especially the dreaded, time-consuming ones you wish would go away. Tasks such as those will become good candidates for you to apply the VBA macros and automated solutions skills that the following lessons will teach you.

REFERENCE *There is no video to accompany this lesson.*