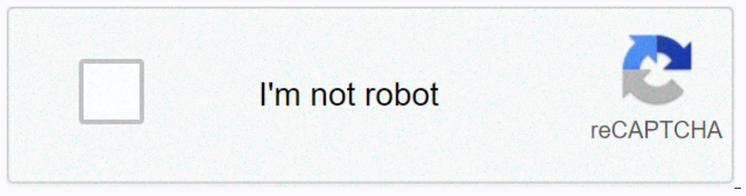

Continue

Backcourt violation nba

Top reviews Latest Top Reviews Alexander M. Wiley, 2015. — 290 pp. — ISBN: 1119089344, 97811119089346Learn to: Save time, automate, and be more productive with Excel macros. Instantly deploy and use more than 70 Excel macros. Understand how each macro works and how to customize it. Reuse the macros in other workbooks or with other macros. Maximize these out-of-the-box Excel macros to make your workflow smarter and faster. Macros automate aspects of Excel so you can streamline your workflow and minimize errors. Even if you're not a VBA guru and don't have time to become one, take your heart! This book contains some of the most widely used Excel macros and shows you exactly how to put them to work to solve problems you encounter while working. Macros 101 - get an overview of macro foundations, meet VBA and Visual Basic Editor, and learn how macros are constructed Workbook Workshop - explore macros that can automate a range of tasks, from creating workbooks and backing them up to printing all worksheets at once Navigation and manipulation - discovering macros that work together to help you navigate and manipulate your spreadsheets , exfoliate, and shape data Macro report magic - find macros to automate PivotTable and chart tasks and automatically email reports as attachmentsOver the AuthorMichael Alexander is a Microsoft Certified Application Developer and a Microsoft MVP. He has written numerous books on advanced business analytics with Microsoft Access and Excel. His tutorial website www.datapigtechnologies.com provides tips on Excel and Access for the Microsoft Office community. Go to the accompanying website to download the sample files for each macro in the book. Lets you see the macro at work and copy and paste the code. The file will be sent to your email address. It may take up to 1-5 minutes for you to receive it. Book name: Excel Macros For Dummies Author: Michael Alexander ISBN-10: 1119089344 Years: 2015 Pages: 288 Language: English File Size: 6.05MB File Format: PDF Ready-made Excel Macros that streamline your Excel Macros workflow for Dummies, helps you save time, automate and be more productive, even without programming experience. Each chapter offers practical macros that you immediately implement, with practical exercises that expand your knowledge and help you understand the mechanics at work. You'll find the most effective Excel macros for solving common problems, and explain why and where to use them all, plus valuable guidelines and step-by-step instructions to apply them effectively. Learn how to customize your applications to look and work exactly like this, with simple, friendly directly applicable to real-world tasks. Follow it from start to finish, or quickly pick up problems when they occur; the clear layout of the book and and makes it an irreplaceable desk reference, and all macro code is available for download from the accompanying website. Microsoft Excel is the world's leading spreadsheet application and supports VBA macros that allow you to customize the program and automate many common tasks. This book helps you use macros to get more done, and get it done better. Understand the basics of VBA and macros Work with workbooks, worksheets, and achieve Clean Data, automate reporting, and send email from Excel Use tips and tricks that streamline your workflow If you have an Excel problem, there is a macro to fix it. You don't have to be a programmer and you don't have to spend months learning code. Excel Macros For Dummies gives you the recipes you need, and the knowledge to apply them effectively. Download PDF NOTICE:IF DOWNLOAD LINK IS BROKEN REPORT US AT Do you track what percentage of the time you spend working on Excel goes away in small and relatively unimimportant, but repetitive, tasks? If you are (and maybe even if you haven't), you've probably noticed that routine things like formatting or inserting standard text usually take a significant amount of time. Even if you practice in performing these activities and you are able to complete them relatively quickly, those 5 minutes you spend almost every day inserting the name and details of your business into all the Excel worksheets you add to customers/colleagues in time. In most (not all) cases, investing a lot of time on these common but repetitive operations does not produce proportional results. In fact, most of them are great examples of the 80/20 principle in action. They are part of the majority of efforts that have little impact on output. If you're reading this Excel macro tutorial for beginners, you're probably already aware of how macros are one of Excel's most powerful features and how they can help you automate repetitive tasks. As a result, you're probably looking for a basic beginner guide that explains how to create macros in an easy-to-follow way. Macros are an advanced subject, and if you want to become an advanced programmer, you'll encounter complex materials. This is why some training resources on this topic are sometimes difficult to follow. However, this does not mean that the process for setting up a macro in Excel is impossible to learn. In fact, in this Excel Macro Tutorial for Beginners, I explain how to start making basic macros in 7 simple steps. In addition to the step-by-step process of setting up a macro, this guide includes a step-by-step example. More In this Excel tutorial, I'll show you how to set up a macro that does the following: Type This is the best Excel tutorial in the active cell. Automatically adjusts the column width of the active cell, cell, that the typed text fits in one column. Color the active cell red. Change the font color of the active cell to blue. This Excel Macro Tutorial for Beginners is accompanied by an Excel workbook with the data and macros I use (including the macro I describe above). You can immediately get free access to this sample workbook by clicking on the button below. The 7 steps I'm laying below are enough to get you on your way to producing basic Excel macros. However, if you're interested in fully unleashing the power of macros and interested in learning to program Excel macros with VBA, the second part of this Excel Macro tutorial for beginners allows you to learn more advanced topics by: Introducing you to Visual Basic for Applications (or VBA) and the Visual Basic Editor (or VBE). Explain how to see the actual programming instructions behind a macro and how to use it to start learning to write Excel macrocode. Give you some tips you're about to use to improve and accelerate the process of learning about macros and VBA programming. You use the following table of contents to go to a section. However, I suggest that you don't actually skip all sections ☺. Are you ready to create your first Excel macro? Then let's start with the preparations... Before you start creating macros: Show the Developer tab Before you create your first Excel macro, make sure you have the right tools. In Excel, most useful commands when working with Excel macros and Visual Basic for applications are on the Developer tab. By default, excel hides the Developer tab. Unless you (or someone else) added the Developer tab to the ribbon, you'll need to show excel to access the appropriate tools when setting up a macro. In this section, I explain how to add the Developers tab to the ribbon. At the end of the step-by-step explanation, there is an image with the whole process. Keep in mind that you need to ask Excel to display the Developer tab once. Assuming the set-up doesn't be rolled back later, Excel will continue to display the tab in future opportunities. 1. Step #1. Open the Excel Options dialog box using one of the following methods: Method #1.Step #1: Click the ribbon with the mouse. Step #2: Excel displays a context menu. Step #3: Click Customize the ribbon.... The following image shows the following three steps: Method #2.Step #1: Click the File Ribbon tab.Step #2: On the navigation bar on the left side of the screen, click Options. The following image shows how to do this: Method keyboard shortcuts such as 'Alt + T + O' or 'Alt + F + T'. 2. Step #2. Once you're in the Excel Options dialog box, make sure you're on the Ribbon tab by clicking this tab on the navigation bar on the left 3. Step #3. View the Customize the Ribbon box, the list box on the right side of the Excel Options dialog box, and find Developer. This is the Developer tab that by default is the third tab at the bottom of the list (just above Add-ins and Background Removal). By default, the Developer box to the left is empty. In this case, the Developer tab does not appear on the ribbon. If this box has a check mark, the Developer tab appears on the ribbon. 4. Step #4. If the Developer left box is empty, click it to add a check mark. If the box already has a check mark, you don't need to do anything (you should already have the Developer tab on the ribbon). 5. Step #5. Click the OK button in the lower-right corner of the Excel Options dialog box. In Excel, go back to the worksheet you were working on and the Developer tab appears on the ribbon. How to enable the Developers in Pictures tab The image below takes you step by step through the process described above: Excel excel-creating excel perks using one of the following tools: The macro recorder, which helps you capture the actions you perform in an Excel workbook. The Visual Basic Editor, which requires you to write the instructions you want Excel to follow in the Visual Basic for Applications programming language. The second option (which requires programming) is more complex than the first, especially if you're a newcomer to the world of macros and you have no programming experience. Since this guide is aimed at beginners, I explain below how to record an Excel macro using the recorder. If your goal is to only record and play macros, this tutorial probably includes most of the knowledge you need to achieve your goal. As explained by John Walkenbach (one of the main authorities in Microsoft Excel) in the Excel 2013 Bible, if your goal is to only record and play macros: (...) you don't need to be involved in the language itself (although a basic understanding of how things work doesn't hurt). However, if you want to take full advantage of Excel macros and fully use their power, you'll ultimately need to learn VBA. As Mr Excel (Bill Jelen) (one of the main Excel wizards) and Tracy Syrstad (an Excel and Access consultant) say in Excel 2013 VBA and Macros, recording a macro is useful if you are a beginner and have no experience in macro programming, but ... (...) if you gain more knowledge and experience, you start to include lines of code less often. Therefore, I cover a number of topics related to Visual Basic for applications deeper in other tutorials. However, for the time being, I explain below how to record an Excel macro using the recorder: The 7 steps to creating your first macro OK... In the meantime, you've added the Developer tab ribbon and you are aware that there are two different tools that you use to produce a macro, including the recorder. You're ready to create your first Excel macro. To do it, just follow the 7 simple steps I explain below. 1. Step #1. Click the Developer tab. 2. Step #2. Make sure the relative reference recording is turned on by checking Use Relative References. If the relative reference recording is not enabled, as in the case of the screenshot below, click Use Relative References. If the relative reference recording is turned on, as in the case of the screenshot below, you don't need to click anything. I can explain the use of relative and absolute references further in future tutorials. However, make sure you have the relative reference recording turned on. When the relative recording is disabled (which is standard), absolute/exact cell addresses are recorded. When relative reference recording is enabled, all actions recorded by Excel are relative to the active cell. In other words, the absolute recording, as explained by Bill Jelen in Excel 2013 in Depth, is extremely literal. Let's say you include an Excel macro that types, This is the best Excel tutorial in the active cell. Copy the text you just typed and paste it directly into the cell below. If at the time of recording, the active cell is A3 and you don't turn on the relative recording, you'll include the macro records to run: Type This is the best Excel tutorial in the active cell. Copy the text and paste it into cell A4, the cell directly below the active cell at the time of the macro's recording. As you imagine, this macro does not work very well if, when using it, you are in a cell other than A3. The following image shows what this would look like if you worked in cell H1 and activated the macro with absolute references explained above. 3. Step #3. Click Record Macro on the Developer tab, or the Include Macro button that appears on the left side of the status bar. Step #4. The Record Macro dialog box appears. This dialog box allows you to assign a name to the macro. Excel assigns a default name to macros: Macro1, Macro2, Macro3, and so on. However, as explained by John Walkenbach in Excel VBA Programming for Dummies, it is generally more useful to use a descriptive name. The most important rules for macro names are that they must start with a letter or an underscore (_) (no number), cannot have spaces or special characters, except for underscore (which is allowed) and should not be in violation of previously existing names. I cover it of macro naming in detail here (for Sub procedures) and here (for Function Procedures). For example, Best Excel Tutorial is not an acceptable name, but Best_Excel_Tutorial works: assign a shortcut to the step is optional. You set a macro without a shortcut, but by selecting a shortcut, you can run the macro by simply pressing the chosen keyboard shortcut. The shortcut to be pointed is from the Ctrl + key combination form. In this context, the key combination (i) means a letter in itself or (ii) a combination of a letter plus the Shift key. When you create keyboard shortcuts for macros, you want to be careful about the exact key combination you choose. If you choose a shortcut that was previously assigned (for example, a built-in shortcut key), the keyboard shortcut choice for the Excel macro transfers and turn off the pre-existing shortcut. Since Excel has several built-in keyboard shortcuts from the Ctrl + Letter form, the risk of disabling built-in keyboard shortcuts is not so small. For example, take the Ctrl + B shortcut, the built-in shortcut for the Bold command. However, if you assign the Ctrl + B shortcut to a particular macro, the built-in shortcut for the Bold command is disabled. Pressing Ctrl + B runs the macro, but does not make the font of the selected text bold. One way to address this problem, which generally works, is to assign keyboard shortcuts from the Ctrl + Shift + Letter form. The risk of overriding and disabling a previously existing shortcut is smaller, but at least I suggest that you continue to be careful with the exact key combination you choose. This means, for example, that instead of choosing Ctrl + B as a shortcut, we can assign Ctrl + Shift + B: Determine where you want to save the macro. You save the macro in the workbook you're working on (This workbook), a new Excel file (New workbook) or a personal macro workbook (Personal Macro Workbook). The default selection is to save the macro in the workbook you're working on. In this case, you only use that macro when that particular workbook is open. If you choose New workbook, Excel opens a new file. You record and save the macro in that new workbook, but just like when selecting This workbook, the macro only works in the file where it was created. The more advanced storage option is Personal Macro Workbook. In Excel 2013 In Depth, Bill Jelen defines a personal macro workbook as (...) a special workbook designed for general-purpose macros that may apply to any workbook. The main advantage of storing macros in the personal macro workbook is that these macros can be used later in future Excel files, because all of those macros are available when you use Excel on the same computer where you saved them, regardless of whether you're working on a new or different Excel file than the file created the macro. Create a macro description. Having a macro description is optional. However, as explained by Greg Harvey in Excel 2013 All-in-One for Dummies: It's a good idea to get into the habit of recording this information every time you build a new macro so that you and your colleagues can always know what to expect from the macro when one of you run it. Harvey also suggests that you include the date the macro was saved and who created the macro. 5. Step #5. Once you've assigned a name, set the location where you want to save the macro and (if desired) assigned a shortcut and created a macro description, click the OK button to close the Include Macro dialog box. 6. Step #6. Perform the actions that the macro should record and save. 7. Step #7. Click Stop Recording on the Developer tab, or click the Stop Recording button that appears on the left side of the status bar. That's it... it really only takes these 7 simple steps to record your first macro. Example of creating an Excel macro If you follow the 7 simple steps above, you already start creating basic macros. However, I promised that this Excel Macro Tutorial for beginners would include an example. That's why we set up a macro in this section that does the following three things: Type This is the best Excel tutorial in the active cell. Automatically adjusts the column width of the active cell. Color the active cell red. Change the font color of the active cell to blue. I've already explained how to make the Developers tab appear in Excel. Since you only need to ask Excel once to view the Developer tab, the image below shows only the actual image of the macro. For this particular example, I used the parameters described above when working with the Record Macro dialog box. More precisely: The name assigned to the macro is Best_Excel_Tutorial, the keyboard shortcut is Ctrl + Shift + B, and the Excel macro is stored in the Excel workbook I was working on. Are you ready? When you're done... Congratulations! You created your first Excel macro! Amazing! You now continue to run your new macro using the shortcut that has been assigned (in this case Ctrl + Shift + B). As you become a more advanced macro user, you'll see that there are several other ways to run a macro, such as the Best_Excel_Tutorial macro above. I hope you've found it easy to create your first Excel macro. At the very least, I hope you realize that the basics of Excel macros aren't as complicated as they seem at first glance. I know the macro we have included above is a very example and, in other posts about VBA and macros, I dig deeper into more complicated topics that allow you to set up more complex and powerful macros. However, it is true that the information in the previous previous Tutorial for beginners is enough to set up a relatively wide range of macros. In the Excel 2013 Bible, John Walkenbach explains: In most cases, you record your actions as macro and then simply replay the macro; you don't have to look at the code that is automatically generated. Congratulations again for creating your first Excel macro! The next step in creating Excel Macros: Enter VBA I've quoted twice how John Walkenbach, one of the most prolific authors on the subject of spreadsheets, implies that casual users of Excel macros don't necessarily need to learn programming. However, this does not mean that you should not learn to code. If you're committed to unleashing the power of Excel macros, you need to learn Visual Basic for applications. Programming Excel macros with VBA is more powerful than just recording the macros for a variety of reasons, with the most important being that you use VBA code to perform tasks that cannot be recorded with the macro recorder. For example, in the Excel 2013 Bible, John Walkenbach provides some examples of tasks that cannot be included, such as displaying custom dialog boxes or processing data in a series of workbooks and even additional special-purpose add-ons. In Excel 2013 VBA and Macros, Bill Jelen and Tracy Syrstad tell us that is important to recognize that the Macro Recorder will never correctly record the intent of the AutoSum button. If I had the space, I could continue making a really long list of examples of how programming with VBA is a superior way to create macros than using the Macro Recorder. Start by learning how to write Excel macrocode You've already learned how to set up a macro in Excel and, as you saw in the most recent sections, the macro works. To start learning to program macros, it's helpful to look at the actual instructions (or code) behind you when recording the macro. To do this, you need to activate the Visual Basic Editor. Let's open the VBE by clicking Visual Basic on the Developer tab or using the Alt + F11 shortcut. Excel opens the Visual Basic Editor that looks something like this: The VBE window is customizable, so it's (very) possible that the window displayed in your computer looks slightly different from the screenshot above. The first time I saw this window several years ago, the next one where my first two questions: What am I looking at? Perhaps more importantly, where is the code of my macro? Since you have these same questions, let us answer them. What Are You Looking At In The Visual Basic Editor Divide the VBE into 6 main sections: 1. Item #1: The Menu Bar. The visual basic editor is, pretty much, like the menu bars you use in other programs. More precisely, a menu bar contains the drop-down menus where you most commands you use to give instructions to and interact with the VBE. If you're working with more recent versions of Excel (2007 or later), Excel itself may not have a menu bar, but a ribbon. The reason for this is that microsoft has replaced the menus and toolbars of some programs with the ribbon from Microsoft Office 2007. 2. Item #2: The toolbar. The VBE Toolbar, just as it happens with the menu bar, is similar to the toolbars you may have encountered when using other types of software. To be precise, the toolbar contains items such as on-screen buttons, icons, menus, and similar elements. The toolbar shown in the screenshot above is the default and default toolbar of the Visual Basic Editor. As explained by John Walkenbach in Excel VBA Programming for Dummies, most people (including Walkenbach himself) just leave them as they are. As explained above, the Excel window doesn't show a newer version of Excel (starting with 2007) or a toolbar or menu bar because Microsoft has replaced both items with the ribbon. 3. Item #3: Project Window (or Project Explorer). The project window is the part of the VBE where you can find a list of all open Excel workbooks and the add-ins that are loaded. This section is useful for navigation purposes. As you can see in the image below, you use the Visual Basic Editor to expand or collapse the different sections of the list by clicking on the + or - (as may be the case) that appears on the left side of the correct branching. When VBAProject is expanded, the various folders that are currently loaded are displayed. There may be different folders for different types of items, such as sheets, objects, forms, and modules. I explain what all these are in other VBA and macro tutorials. When a folder is expanded, you see the individual components in that folder. For example, in the image above, there are 2 folders (Microsoft Excel objects and modules) and the Microsoft Excel Objects folder (which has been expanded) two items (Sheet1 and ThisWorkbook). If you don't see the Project Explorer, it might be hidden. To view the project window, use the Ctrl + R shortcut, click the Project explorer icon on the toolbar, or go to the View menu, and then click Project Explorer: 4. Item #4: The Properties window. The Properties window is the portion of the VBE that you use to edit the properties of anything you might have selected in the project window. It is possible to hide or make the property sheet visible. If your Visual Basic Editor does not currently display the Properties window, use the F4 shortcut, On the Property Sheet icon on the toolbar, or expand the View menu, and then click Property Window: 5. Item #5: Programming Programming Code) Window. The programming window displays the VBA code that you include. I'll explain how to let the Visual Basic Editor display the code of your macros in the next section. In addition to displaying code, the code window is where you actually write or edit VBA code. 6. Item #6: The instant window. The direct window is useful for detecting errors, checking or detecting them. You may have noticed that, in the first screenshot of the VBE I included above, there is no direct window. There are two main reasons for this: this window is hidden by default. As explained by John Walkenbach in Excel VBA Programming for Dummies, this window is not that useful for beginners and therefore it may be better to keep it hidden or, if currently shown, hide it. To make the Instant View window visible, use the Ctrl + G shortcut, or open the View menu, and then click Direct Window. Now that you know what you're looking at when you're working with the Visual Basic Editor, let's move on and learn how to see the actual code of the macro you've created... Where is your VBA macro code The portion of the VBE that you usually use for navigation purposes is the project window. Let's go back to it and take a look at the screenshot above: In the screenshot above, VBAProject is extensive and shows two folders: Microsoft Excel Objects and Modules. You see the elements in the first folder (Microsoft Excel Objects), but not in the second (modules). To expand the Modules folder and view its components, click +: The Project Explorer looks something like this: The items that appear in the Microsoft Excel Objects folder may look familiar. However, you wonder... What is a module? A module is, in the words of John Walkenbach in the Bible of Excel 2013, a container for VBA code. In other words, a module is where the VBA code is actually stored. If you followed the example in this Excel Macro Tutorial for Beginners, your macro code is in a module, specifically Module1: To display the Visual Basic Editor the VBA code, double-click Module1, or right-click Module1, and then select Show Code: And the VBE displays the macro code in the programming window. If you followed the example in this beginner guide and created the Best_Excel_Tutorial macro, your code looks something like this: Does this make sense to you? The good news is that, to some extent, it probably makes a bit of sense. However, you feel that you don't fully understand all the instructions in the Excel macro you've created. You also wonder... why does something as simple as writing text, automatically adjusting a column, coloring a and changing the font color as much programming? All these feelings and questions are his Let's take a look at the macro code to understand all this. Learn VBA from scratch Using an example of Basic Excel Macro Code Good News first. As you may have noticed, VBA code (kind) is similar to English. In VBA for Excel Made Simple, Keith Darlington (an experienced programming teacher) explains how structured English (which is similar to plain English) can be a useful intermediate step to think about the instructions a macro needs to follow before actually writing those instructions in Visual Basic for Applications. Therefore, you are probably able to understand some of the words, and maybe even some of the instructions above. For example, you can recognize or partially understand the following rules from the Best_Excel_Tutorial macro created in this beginner's guide: ActiveCell.Select.De active cell is the cell currently selected in a worksheet. I assume that even if you are not familiar with Excel or Visual Basic for applications, you know what the word select means. That's right, as you probably imagine, this piece of code selects the current active cell. Selection.Columns.Autofit.This code line starts again with a selection. However, it refers to columns and to automatic assembly. If you remember, the second thing that the Best_Excel_Tutorial macro had to do is to automatically fit the column width of the active cell so that the text that was typed (this is the best Excel tutorial) fits into a single cell. Given the above, you (correctly) thought that the purpose of this piece of VBA code is to automatically fit into the column where the active cell is located so that the text of the macro types fits into a column. However, if you currently know more about Excel macros, you may want to understand what each line of code means, so let's understand some of these basics of VBA code. Excel Macro Code Basics To understand each of the instructions behind the macro you've included, we check the full code line-by-line and item-by-item, which is how Excel runs the macro. Don't worry if you don't understand every line below now. The purpose of this part of the guide is not to make you an expert in Visual Basic for applications, but to give you a basic idea of how VBA works and, more importantly, show you what are the instructions that Excel performs to write This is the best Excel tutorial, automatically fit into the column, color the active cell red and change the font color blue. You'll notice (not just this time, but generally when recording macros) that the VBA code can perform some actions that you don't actually perform. According to John Walkenbach, in the Excel 2013 Bible, this is just a byproduct of the method that Excel uses to take actions in code. In other words, at the moment you don't have to worry about the That seems to be useless. I can explain, in future tutorials, how to remove them. The programming window that contains the code of the macro you created has the following components: Item #1: Sub Best_Excel_Tutorial () Sub stands for Subprocedure. This is one of two types of macros or procedures you use in Excel. Subprocedures perform certain actions or activities in Excel. The other type of procedure is a function procedure. Job procedures are used to perform calculations and return a value. So... What does this rule do? It simply tells Excel that you are writing a new subprocedure. Subprocedures should always start with: The word Sub. The name of the procedure, in this case Best_Excel_Tutorial. Parentheses. In addition, subprocedures should always end with the words End Sub, as you see in the last line of code shown in the screenshot above (signaled by the number 8). Item #2: Lines Of VBA Code In Green Font That Begin With ' I'm referr to the following lines: ' Best_Excel_Tutorial Macro ' Types This is the best Excel tutorial. Column Auto-fits. Cell color red. Font color blue. Keyboard shortcut: Ctrl+Shift+B These are simple comments. The comments have the following main characteristics: They start (or are indicated by) an apostrophe ('). Visual Basic for Applications basically ignores the text that comes after the apostrophe to the end of the line. Therefore, when running a macro, Excel simply ignores the comments. As a result of the previous point, the main purpose of a comment is to display information about that particular macro and help you understand it. Comments can explain things, such as the purpose of a procedure or what are the most recent changes made to the procedure. Item #3: ActiveCell.Select As explained above, this rule specifies that Excel must select the current active cell. More precisely: ActiveCell refers to the current active cell in the active window. Select activates an object on the current active Excel worksheet, in this case the current active cell that ActiveCell referred to. Item #4: ActiveCell.FormulaR1C1 = This is the best Excel tutorial In this statement, Excel is instructed to write This is the best Excel tutorial in the active cell. Let's check each of the individual parts of the rule: you already know what activecell's purpose is. FormulaR1C1 tells Excel to set the formula for the object, in this case the current active cell to which ActiveCell refers. The last part (R1C1) refers to the R1C1 format in which cell references are relative rather than absolute. I put R1C1 notation more in depth in this tutorial. Remember, however, that at the beginning of this guide, I explain: Why you should relative reference recording; and How to do this. Do. is the best Excel tutorial indicates what the formula is (in this case the text) that should be placed in the object, in this case the active cell. Item #5: Selection.Columns.AutoFit As I explain above, this specific instruction ensures that Excel automatically fits into the column of the active cell, so that the text that typed the macro fully fits in. The following is the purpose of the different parts of this instruction: Selection simply represents the current selection, in this case the active cell. Columns selects the columns in the selection, in this case the column where the active cell is located. AutoFit speaks for itself; sets the width of the selected columns (as in this case) or the height of the selected rows to the size reaches the best fit. Item #6: With... End with statement 1 I refer to the next group of lines, which is known as a Met ... End with statement: With Selection.Interior . Pattern = xlSolid . PatternColorIndex = xlAutomatic . Colour = 255 . TintAndShade = 0 . PatternTintAndShade = 0 End With The Excel macro you created, it has already performed two of the four actions it should perform: it typed This is the best Excel tutorial in the active cell. It has automatically mounted the column width so that the text it has typed fits well. As you probably expect, the next activity excel performs is to color the active cell red. You would expect that coloring the active cell is a simple step. However, it turns out that Excel needs to take many steps to perform this action. This is why with... End with explanations exist. The main goal of a Met... End with the statement is to simplify the syntax by running different instructions that all refer to the same object without referring to that object each time. In the case of the example used in this beginner guide, this object is the active cell. As you can see in the screenshot below, the basic macro you've included has two with... End with statements: With... End with instructions have the following structure: In the beginning, they must be objectExpression. I research, in other tutorials, what objectExpression means. At this time, is enough to know that in the case of the example included in this guide, objectExpression is Selection.Interior (for the first Met ... End with statement) and Selection.Font (for the second Met... End with statement), as I explain below. They may have one or more lines of code, which are the instructions that are executed on the object referenced. In the end they have to say End with. In the case of the first Met... End with statement, each of these objects here's how: Now that you have a basic knowledge of what a Met... End with statement does, let's take a look at the first one line-by-line: 1. Line #1: #1: Selection.Interior. This rule tells Excel that it should always refer to the interior of the active cell when performing the instructions that are part of the Met... End with statement. How does it achieve this? With is the beginning of the Met ... End with instruction and inform Excel that the following lines of code refer (work with) the object mentioned in this row. Selection.Interior is the objectExpression that I mentioned above when explaining the structure of Met... End with statements. Selection represents the current selection, which in this example is the active cell, while Interior indicates the interior of an object, in this case the inside of the active cell. 2. Rule #2: . Pattern = xlSolid. This is the met's first rule... End with a statement that refers to the interior of the active cell. It tells Excel to set the inner pattern of the active cell to a solid color. It does this as follows: Pattern sets the interior pattern. xlSolid indicates that the pattern must be a solid color. 3. Rule #3: . PatternColorIndex = xlAutomatic. This rule specifies the automatic pattern for the inside of the active cell as follows: PatternColorIndex sets the color of the interior pattern. xlAutomatic indicates that the color must be an automatic color. 4. Line #4: . Color = 25. This is the statement that actually announces to Excel what is the color it should use to fill the interior of the active cell. Color assigns the cell color, while the number (in this case 255) indicates the color that is red in the Best_Excel_Tutorial macro. 5. Rule #5: . TintAndShade = 0. This rule recommends Excel not to lighten or darken the color chosen for the active cell fill. TintAndShade sets the lighting or darkening of a color. When TintAndShade is set equal to 0 (as in this case), the property is fixed to neutral and therefore there is no lighting or darkening of the color selected for the active cell. 6. Line #6: . PatternTintAndShade = 0. As you imagine, this line brings to Excel that it should not set a tint or shadow pattern for the interior of the active cell. PatternTintAndShade sets the tint and shadow pattern for the inside of an object, in this case the selected cell. 7. Line #7: End with. This line signals to Excel the end of the Met... End with statement. Therefore, the following lines of code refer to an object other than that on which it is... End with statement did. In the case of the example used in this Excel Macro Tutorial for beginners, the end of the first Met ... End with instruction indicates that the following instructions do not refer to the interior of the active cell. Item #7: With... End with statement 2 You learned above what is a Met... End with statement and what is the general structure. That's why I'm going straight ahead. a line-by-line explanation of the second Met... End with the statement of the macro that, as you probably expect, performs the last of the instructions you gave when creating it: changing the font color from the active cell to blue. You'll probably be happy to read that this second Met... End with statement is shorter than the first. Let's start with the line-by-line explanation... 1. Rule #1: With Selection.Font. As I explain above, this is the opening of the Met... End with the statement, where the Met Excel tells you that the following instructions work with the object that appears here. In this case, this object is Selection.Font. So what is Selection.Font.? Selection is the current selection, which is the active cell in the Best_Excel_Tutorial macro, while Font (not surprisingly) is the font. In other words, Selection.Font means the font of the text in the active cell. Therefore, Met Selection.Font basically inform Excel that all the rules of code that are part of the Met ... End with the statement refer to the font of the active cell. 2. Rule #2: . Color = -4165632. This line of code, as you might expect given the very similar line in the first Met ... End with the above statement, Excel tells what color to use for the font in the active cell. Color assigns the color, while the number (in this case -4165632) is the actual color code that is blue in this case. 3. Rule #3: . TintAndShade = 0. This statement is exactly the same as any of the lines in the Met... End with statement above. It instructs Excel not to lighten or darken the color of the font. Since TintAndShade defines the lighting or darkening of a color, when it equals 0 (as is here), it illuminates or darkens the font color of the active cell. 4. Line #4: End with. This is the end of the Met... End with statement. Therefore, lines of code below this do not refer to the font of the active cell. Item #8: Sub-end summaries end the execution of something, in this case a subprocedure. This means that once Excel executes this line of code, the macro you created will run. In other words, this is the end of the code of your first Excel macro. A few latest tips on how to learn more about Excel macros If you want to take an extra step to speed up your learning process on Excel macros, here's some of the latest tips. You try most of it in the Excel workbook example that guides this Excel macro study for beginners. You can immediately get free access to this sample workbook by clicking on the button below. Change parts of the VBA code to try new things. For example, change ActiveCell.FormulaR1C1 = This is the best for ActiveCell.FormulaR1C1 = I love Microsoft Excel. You also change the numbers that cese and font colour. For example, change. Color = 255 for . Colour = 10 and . Color = –4165632 for . Color = 200. Go back to the Excel window head and redo the macro (for example, using the Ctrl + Shift + B shortcut you assigned) and see what's happening. The results changed significantly, right? Isn't it interesting how much difference a few small items in the VBA code can make? Remove certain instructions from the code to see how they affect the macro. For example, what would happen if you removed 'Selection.Columns.AutoFit'? Try it out. Return to Excel again and run the macro one more time with this deletion. What happened? Was it what you expected? One of the best ways to learn Excel macro code is to repeat the exercise in this guide, so I encourage you to do it. How?1. Include Excel macros differently than the one shown in this Excel macro study for beginners. Try new things and see what happens.2. Open the VBE and go through the VBA code line-by-line to understand what the purpose of each statement is. Perhaps even better, if you have a big enough screen (or two monitors), is to follow the advice of John Walkenbach in the Excel 2013 Bible and ... (...) set your screen so that you see the code that is generated in the VB Editor windows. Reading and studying. For example, you go through the Archives of Power Spreadsheets to find all the Excel tutorials I've written about VBA and macros. Conclusion Again, congratulations! After going through this Excel Macro Tutorial for beginners, you've created your first macro and understood the VBA code behind it. As you may have seen, setting up a macro using Excel's recorder is relatively easy and can be done in seven simple steps. If your main goal is to just record excel macros and play it, you're ready to go! If your goal is to become a macro expert, I hope this basic guide has you a good idea of how to include Excel macros and a basic introduction to VBA programming. Furthermore, I hope this Excel Macro Tutorial for beginners gives you some confidence about your Excel programming capabilities. Put into practice the latest tips on how to learn about macros I've provided in the section above. Produce macros, study the VBA code behind them and try different things to see what happens. If you continue to study and practice Visual Basic for Applications, including the topics I go into other Excel VBA tutorials in Power Spreadsheets, you will soon include: Much better understand what you did while creating your first macro and how the Best_Excel_Tutorial macro works. Write and use much more complex and advanced Excel macros. referenced in this Excel Macro tutorial for beginner books in the Power Spreadsheets Library. Spreadsheets. Keith (2004), VBA for Excel Made Simple. Burlington, MA: Simple Books Made. Harvey, Greg (2013). Excel 2013 All-in-One for Dummies. Hoboken, NJ: John Wiley & Sons Inc. Jelen, Bill (2013). Excel 2013 in Depth. United States of America: Que Publishing. Jelen, Bill and Syrstad, Tracy (2013). VBA and Macros from Excel 2013. United States of America: Pearson Education, Inc. Walkenbach, John (2013). Excel 2013 Bible. Indianapolis, IN: John Wiley & Sons Inc. Walkenbach, John (2013). Excel VBA programming for Dummies. Hoboken, NJ: John Wiley & Sons Inc. Inc.