Tutorial macro excel pdf

I'm not robot

reCAPTCHA

Continue

Do you track how much time you spend working on Excel going into small and relatively unimportant but repetitive tasks? If you have (and perhaps even if you don't), you've probably noticed that ordinary things such as formatting or inserting standard text usually takes a considerable amount of time. Even if you have practice in holding these activities and you can complete them relatively quickly, those 5 minutes that you spend almost every day inserting your company's name and details in all Excel sheets you send to customers/colleagues start saving over time. In most (not all) cases, investing a lot of time on these common but repetitive operations does not yield proportional results. In fact, most of them are excellent examples of the 80/20 principle in action. They are part of most efforts that have little impact on the exit. However, if you're reading this Excel macro lesson for beginners, you probably already know how macros are one of Excel's most powerful features and how they can help you automate repetitive tasks. As a result of this, you are probably looking for a basic beginner's guide that explains, in an easy-to-follow way, how to create macros. Macros are an advanced topic, and if you want to become an advanced programmer, you will come across complex materials. This is why some training resources on this topic are sometimes difficult to follow. However, this does not mean that the macro setting process in Excel cannot be studied. In fact, in this Excel Macro Beginners tutorial, I'll explain how you can start creating basic macros now in 7 simple steps. In addition to step-by-step through the macro creation process, this guide includes a step-by-step example. More precisely, in this Excel tutorial I'll show you how to customize the macro that does the following: Type Is the best Excel tutorial in active cell. Automatic fit to the width of the active cell column, so that the machine-intensive text is placed in one column. The active cell is red. Change the color of the active cell font to blue. This Excel Macro Tutorial for beginners is accompanied by an Excel work book containing the data and macros I use (including the macro I describe above). You can get immediate free access to this approximate book by clicking on the button below. The 7 steps that I'll explain below are enough to set you on your way to producing major Excel macros. However, if you are interested in fully unleashing the power of macros and are interested in learning how to program Excel macros with VBA, the second part of this Excel Macro Tutorial for Beginners sets you on your way to learn more advanced topics: Introducing you Visual Basic for Apps (or VBA) and Visual Basic (or VBE). Explaining how you can see the actual programming instructions for the macro, the macro, how you can use this to start learning how to write the macrococ code Excel. Give you some tips that you can start using now to improve and speed up the process of learning macros and VBA programming. You can use the following content table to go to any section. However, I suggest that you don't really miss any ☺. Are you ready to create your first Excel macro? Then let's start with the preparation ... Before you start building macros: Show the developer tab before you create your first Excel macro, you need to make sure that you have the right tools. In Excel, most of the useful commands you can do with Excel and Visual Basic for apps are in the Developer tab. The developer tab, by default, is hidden by Excel. So if you (or someone else) hasn't added a Developer tab to the tape, you should make Excel show it in order to have access to the appropriate tools when setting up the macro. In this section, I'll explain how you can add a developer tab to the feed. At the end of the step-by-step explanation, an image appears showing the whole process. Please note that you only need to ask Excel to display the developer tab once. Assuming the setting isn't canceled later, Excel continues to display the tab in future features. Step #1. Open the Excel Options dialogue using one of the following methods: #1.Step #1: Using a mouse, right click on Ribbon.Step #2: Excel displays contextual menus. Step #3: Click on the Button Tune Tape.... The following image illustrates these 3 steps: #2.Step #1: Click on the Tab.Step #2: Click on options on the navigation bar on the left side of the screen. The following image shows you how to do it: Method #3.Use keyboard shortcuts such as Alt and T O or Alt and F T. 2. Step #2. Once you're in the Excel Options dialogue, make sure you're on the tab to set up the tape by clicking on this tab on the navigation bar on the left side. Step #3. Take a look at the Customize the Ribbon list field, which is a list field located on the right side of the Excel Options dialogue, and find the Developer. This is the Developer tab, which, by default, is the third tab at the bottom of the list (just above Add-Ins and Background Removal). The box to the left of the Developer is empty by default. In this case, the Developer tab does not appear in The Feed. If there is a check mark in this field, the Developer tab appears in The Feed. Step #4. If the field to the left of the Developer is empty, click on it to add a tick. If you already have a tick in the box, you don't have to do anything (you should already have a developer tab in the feed). Step #5. Click ON OK bottom right corner of the Excel Options dialogue. Excel brings you back to the sheet you've been working on and the Developer Tab In the tape. How to incorporate the developer tab in the images The image below takes you, step by step, through the process described above: Tools to create Excel Macros Excel allows you to create macros using any of the following tools: Macro Recorder, which allows you to record the actions you're doing in Excel Workbook. A visual basic editor that requires you to write instructions that you want Excel to follow the Visual Basic programming language for apps. The second option (which requires programming) is more complex than the first, especially if you are new to the world of macros and you have no programming experience. Since this guide is for beginners, I'll explain below how to write down an Excel macro with a recorder. If your goal is to only record and play macros, this tutorial probably covers most of the knowledge needed to achieve your goal. As explained by John Walkerbach (one of the leading bodies in Microsoft Excel) in the Excel Bible 2013, if your goal is only to record and play macros: (...) you don't have to worry about the language itself (although a basic understanding of how things work doesn't do any harm). However, if you want to make the most of Excel macros and use their power to the fullest, you will eventually need to study VBA. As Mr. Excel (Bill Helen) (another of the advanced Masters Excel) and Tracy Sirstad (Consultant Excel and Access) say in Excel 2013 VBA and Macros, recording macros is useful when you're new and have no experience in macro programming, but... (...) as you gain more knowledge and experience, you start recording lines of code less frequently. So I'm more deeply covering some of the topics related to Visual Basic for applications in other tutorials. However, for now, I'll explain below how you can record a macro Excel using a recorder: 7 easy steps to building your first Macro OK... By now, you've added a developer tab to the tape, and you know that there are two different tools you can use to create a macro, including a voice recorder. You are ready to make your first macro Excel. To do this, just follow the 7 simple steps that I will explain below. Step #1. Click on the developer tab. Step #2. Make sure the relative link record is on by checking use of relative links. If the relative reference is not included, as is the case with the screenshot below, click on the Use of Relative Links button. If you have a relative reference record enabled, as with 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, for now, make sure you have included relative entries links. When turned off records (which are by default) absolute/exact addresses of cell cells When you turn on a relative reference record, any actions recorded by Excel are associated with an active cell. In other words, the absolute record, as explained by Bill Elen in Excel 2013 in Depth, is extremely literal. For example, suppose you're recording an Excel macro that: Types Is the best Excel tutorial in an active cell. Copies the text you just typed in and insert it into the cell directly below. If at the time of recording is an active A3 cell and you can't include a relative record, the macro records it should: Type is the best Excel tutorial in an active cell. Copy the text and insert it into the A4 cell, which is a cell directly under the active cell when the macro recording begins. As you can imagine, this macro doesn't work very well if, when you use it, you are in any cell except the A3. The following image shows how it will look if you work in the H1 cell and activate the macro with the absolute links explained above. Step #3. Click on the Macro Record button on the Developer tab or on the Record Macro button that appears on the left side of the status bar. Step #4. Record Macro dialogue appears. This dialogue allows you to: Assign a macro name. Excel assigns the default name to macros: Macro1, Macro2, Macro3, and so on. However, as john Walkenbach explained in Excel VBA programming for Dummies, it is generally more helpful to use a descriptive name. The basic rule for macro names is that they should start with a letter or emphasize (I) (not a number), cannot have spaces or special characters, except to emphasize (which is allowed) and should not fit in with previously existing names. I cover the macro-naming topic in detail here (for Sub procedures) and here (for functional procedures). For example, the Best Excel tutorial is not an acceptable name, but Best_Excel_Tutorial works: Assign a keyboard shortcut to a macro. This step is optional. You can customize the macro without the keyboard shortcut, but the choice of shortcuts allows you to perform a macro by simply pressing the chosen combination of keys. The keyboard label to be assigned has a Ctrl shape and a combination of keys. In this context, the key combination means either (i) the letter itself or (ii) the letter combination plus the Shift key. When creating shortcuts for macros, you want to be careful about the exact combination of keys that you choose. If you choose a keyboard shortcut that was previously assigned (such as a built-in keyboard shortcut), your choice of shortcut for the macro excel overrides and disables the existing keyboard shortcut. Since Excel has several built-in Ctrl-shaped keyboard shortcuts and writing, the risk of disabling built-in keyboard shortcuts isn't so small. Take, for Ctrl and B keyboard shortcut, which is a built-in keyboard shortcut for the Bold team. If, however, you assign the Ctrl and B keyboard shortcut to a specific macro, the built-in keyboard shortcut for the Bold team is disabled. As a result, if you click Ctrl and B, the macro will be executed, but the font of the chosen text will not be bold. One way to solve this problem, which usually works, is to assign a shortcut to the shape of Ctrl and Shift Letter. The risk of rewriting and disabling the previously existing keyboard shortcut is less, but, in any case, I suggest you continue to be careful about the exact combination of the keys that you choose. This means that, for example, instead of choosing Ctrl and B as a keyboard shortcut, we could assign Ctrl and Shift B: Decide where you want to store the macro. You can store the macro in the work book you're working on (This work book), a new Excel file (New Workbook) or a personal macro work book (Personal Macro Workbook). The default choice is to store the macro in the work book you're working on. In this case, you can only use this macro when this particular work book is open. If you choose New Workbook, Excel opens a new file. You can write down and save the macro in this new work book, but as with This Workbook, the macro only works in the file where it was created. A more advanced storage option is the Personal Macro Workbook. In Excel 2013 In Depth, Bill Yellen defines the Personal Macro Workbook as (...) a special work book designed for general purpose macros that can be applied to any work book. The main advantage of saving macros in your personal macro workbook is that these macros can be later used in future Excel files, because all of these macros are available when you use Excel in the same computer where you saved them, regardless of whether you're working on a new or other Excel file from where you 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 you and your colleagues can always know what to expect from a macro when any of you run it. Harvey also proposes to include the date in which the macro was preserved and who created the macro. Step #5. Once you've assigned a name, set the place where you want to store the macro and (if you want) assigned a keyboard shortcut and created a macro description, click on the OK button to close the Macro Record dialogue. Step #6. Perform the steps you want the macro to record and store. Step #7. Click on the Stop entry button on the tab or on the Stop Recording Macro 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. An example of how to create an Excel Macro If you follow the 7 simple steps explained above, you can already start creating basic macros. However, I promised that this Excel Macro beginner tutorial would include an example. So in this section, we're adjusting the macro that does the following three things: This is the best Excel tutorial in the active cell. Automatic fit of the width of the active cell column. The active cell is red. Change the color of the active cell font to blue. I've already explained how you can get a developer tab to show in Excel. Since you only need to ask Excel once to display the developer tab, the image below shows only the actual macro entry. For this particular example, I used the settings above when working with The Record Macro. More precisely, the name assigned to the macro is Best_Excel_Tutorial, the keyboard label is Ctrl and Shift b, and the Excel macro was saved in the Excel work book I was working on. Are you done? If you did ... Congratulations! You created your first Excel Macro! Amazing! Now you can go ahead and run your new macro using the keyboard shortcut that has been assigned (in this case Ctrl and Shift B). As you become a more advanced macrout user, you'll see that there are several other ways to perform macro, such as Best_Excel_Tutorial macro above. I hope you had the easy to create your first macro Excel. At least I hope you understand that the basics of Excel macros are not as complex as it may seem at first glance. I know that the macro we recorded above is a very prime example, and in other posts about VBA and macros, I dig deeper into more complex topics that allow me to create more complex and powerful macros. However, it is true that the information in the previous sections of this Macro tutorial is enough to create a relatively wide range of macros. In the Bible Excel 2013 John Walkerbach explains that: In most cases you can record your actions as a macro and then simply reproduce the macro; you don't need to look at the code that is automatically generated. So once again I congratulate you on creating the first Macro Excel! The next step in creating Excel Macros: Enter VBA I quoted twice as John Walkenbach, one of the most prolific authors on the topic of spreadsheets, implied that casual users of Excel macros do not necessarily need to learn programming. However, this does not mean that you don't have to learn programming. If you're committed to excel macro capabilities, you'll have to explore Visual Basic for apps. Programming Excel macros with VBA is more than just recording macros for a few The main one is that using VBA code allows you to perform tasks that cannot be recorded with macro Recorder. For example: In the 2013 Excel Bible, John Walkerbach lists some examples of tasks that cannot be recorded, such as displaying user dialog boxes or processing data in a series of workbooks, and even creating special-purpose add-ons. In Excel 2013, VBA and Macros' Bill Yellen and Tracy Sirstad tell us that it's important to recognize that macro recorder will never properly record the target of the AutoSum button. If I had space, I could continue to create a very long list of examples of how programming with VBA is an excellent way to create macros than with Macro Recorder. Since writing the macro code Excel, you've already learned how to get macro in Excel, and as you've seen in the last sections, the macro works. In order to start learning how to program macros, it's helpful to take a look at the actual instructions (or code) you've prepared when writing the macro. To do this, you need to activate visual Basic Editor. Let's open the VBE by clicking on the Visual Basic button in the developer tab or using the Alt s F11 keyboard shortcut. Excel opens a visual Basic Editor that looks something like this: the VBE window is configurable so it's (pretty) possible that the window that is displayed in your computer looks a little different from the above screenshot. The first time I saw this box a few years ago, the following is where my first two questions are: What am I looking at? Maybe more importantly, where is the code of my macro? Since you may have the same questions, let's answer them. What you look at in the visual base editor you can divide the VBE into 6 main sections: 1. Item #1: Menu Bar. Visual Basic Editor bar menus are pretty much like the menu bars that you use in other programs. More precisely, the menu contains a retiring menu where you can find most of the commands you use to give instructions, and interact with the VBE. If you're working with later versions of Excel (2007 or later), you may have noticed that Excel itself doesn't have a bar menu, but rather a tape. The reason for this is that with Microsoft Office 2007, Microsoft has replaced the menus and toolbars of some programs with tape. 2. Item #2: Toolbar. The VBE toolbar, as is the case with the menu bar, is similar to the toolbars you may have encountered when using other types of software. More precisely, the toolbar contains elements such as buttons on the screen, icons, menus and similar elements. The toolbar displayed in the screenshot above is the standard and default dashboard of visual Basic Editor. As john Walkenbach explained VBA programming for dummies, most people (including Walkenbach themselves) just leave to leave as they are. As explained above, if you have a newer version of Excel (since 2007), you won't see either the toolbar or the menu bar in the Excel window because Microsoft has replaced both of these items with tape. 3. Item #3: Project Window (or Project Researcher). The project window is part of the VBE, where you can find a list of all Excel open workbooks and downloadable add-ons. This section is useful for navigation purposes. As you can see in the picture below, Visual Basic Editor allows you to expand or roll down different sections of the list by clicking on the I or - as it may be) button that is displayed on the left side of the corresponding branch. When VBAProject expands, it shows different folders that are being downloaded. There may be several folders for different types of items, such as sheets, objects, shapes, and modules. I'll explain all this to other VBA and macro lessons. When the folder is expanded, you can see the individual components inside the folder. For example, in the image above, there are 2 folders (Microsoft Excel Objects and Modules) and Microsoft Excel Objects folder (which is expanded) has two items (Leaf1 and ThisWorkbook). If you don't see Project Explorer, it can be hidden. To show the project window, use the Ctrl s R keyboard shortcut, tap the Project Explorer icon in the toolbar, or go to the View menu and click on Project Explorer: 4. Item #4: Property Window. The Property Window is a VBE section that is used to edit the properties of everything you may have chosen in the project window. You can hide or untangle the Property window. If your visual Basic Editor doesn't show a property window at the moment, use the F4 keyboard shortcut, click on the Property Window icon in the toolbar, or expand the View menu and click on the Property Window: 5. Item #5: Programming Window (or Code). A programming window is where the VBA code you're recording is displayed. I explain how you can get a Visual Basic Editor to display the code of your macros in the next section. In addition to displaying the code, a code window where you can actually write or edit the VBA code. 6. Point #6: Immediate window. An immediate window is useful to notice errors, check or fix. You may have noticed that in the first VBE screenshot that I included above, there is no immediate window. There are two main reasons for this: this window is, by default, hidden. As explained by John Walkenbach in Excel VBA Programming for Dummies, this window is not as useful for beginners and therefore it may be more appropriate to keep it hidden or if it is being shown by hiding it. To untangle the immediate window, use the Ctrl s G keyboard shortcut or get access to it Browse the menu and click on the Immediate Window button. Now that you know what you're looking at when working with Visual Basic Editor, let's go ahead and find out how you can see the actual macro code you've created... Where is your VBA Macro Code Section VBE, which you usually use for navigation purposes is a project window. Let's go back to it and take a closer look at the screenshot above: In the screenshot above, VBAProject expands and shows two folders: Microsoft Excel objects and modules. Items can be seen inside the first folder (Microsoft Excel Objects), but not inside the second (modules). To expand the module folder and see its components, click on K: Project Explorer: Items that appear in the Microsoft Excel Objects folder may look familiar. However, you may wonder... What is a module? The module, according to John Walkenbach in the 2013 Excel Bible, is a container for VBA code. In other words, the module is the place where the VBA code is actually stored. If you've followed suit in this Excel Macro Tutorial for beginners, your macrocode is in the module, more specifically in Module1: In order to get a Visual Basic Editor to display VBA code, double tap Module1 or right click on Module1 and select View Code: And VBE displays the macrocode in the programming window. If you followed suit in this beginner's guide and created a macro Best_Excel_Tutorial, your code looks something like this: does it make any sense to you? The good news is that, to a certain extent, it probably makes a bit of a sense. However, you may feel that you don't fully understand all the instructions in the Excel macro you've created. You may also be surprised... why is something as simple as writing text, automatically typing, coloring the cell and changing font color require so much programming? All these feelings and questions are normal. Let's take a closer look at the macrocoff to understand all of this. Exploring VBA from scratch using the example of a basic Excel Macrocode Good News in the first place. As you may have noticed, the VBA code (kind) is similar to English. In VBA for Excel Made Simple Keith Darlington (experienced programming teacher) explains how structured English (which is similar to regular English) can be a useful intermediate step to think that the instructions that the macro should follow before actually writing these instructions in Visual Basic for applications. So you're probably able to understand some words, and maybe even some of the instructions above. For example, you can recognize or partially understand the following lines from the macro Best_Excel_Tutorial created in this beginner guide: ActiveCell.Select.The active cell is the cell that is currently selected in the sheet. suppose that, even if you're not familiar with Excel or Visual Basic for apps, you know what the word means to choose. This is true, as you probably imagine, this part of the code chooses the current active cell. Selection.Columns.Autofit.This line of code starts, once again, with a choice. However, it then refers to columns and automatic installation. If you remember, the second thing the macro had to do Best_Excel_Tutorial is to automatically match the width of the active cell column so that the text that was hired (This is the best Excel tutorial) is placed in one cell. With that said, you may (correctly) think that the purpose of this part of the VBA code is to automatically place the column where the active cell is, so that the text that macro types fits in one column. However, if you're currently studying Excel macros, you can understand what each line of code means, so let's understand some of these basics of VBA code. The basics of the Excel code macrocos To understand each of the instructions you've written down the macro, let's check the entire line of code by line and point by point, which is how Excel performs 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 the VBA works and, more importantly, to show you what instructions that Excel performs in order to write This is the best Excel tutorial, automatically fit the speakers, color the active cell red and change the color of the font to blue. You'll notice (not just this time, but generally when recording macros) that the VBA code may include some actions that you don't actually perform. According to John Walkenbach of Excel Bible 2013, this is just a step-by-step product of the method that Excel uses to translate actions into code. In other words, at this point, you don't have to worry about lines of code that seem useless. I can explain in future tutorials how you can delete them. The programming window, which contains the code of the created macro, contains the following parts: Item #1: Sub Best_Excel_Tutorial means Sub procedure. This is one of two types of macros or procedures that can be used in Excel. Sub-procedures perform certain actions or actions in Excel. Another type of procedure is the function procedure. Function procedures are used to make calculations and return value. So... What does this line do? It just tells Excel that you are writing a new Sub procedure. Sub procedures should always start with: Word Sub. The name of the procedure, in this case Best_Excel_Tutorial. Brackets. In addition, sub-procedures should always end with End Sub words, as you can see in the last line of code shown in the above (signal number 8). Item #2: VBA lines of code in green, which begin with 'I mean the following lines: 'Best_Excel_Tutorial Macro' Types This is the best Excel tutorial. Auto-fit column. Cellular color is red. The color of the font is blue. 'Keyboard Shortcut: Ctrl'Shift 'B' It's just comments. Comments have the following basic characteristics: they start (or are indicated) apostrophe ('). Visual Basic for applications basically ignores the text that comes after the apostrophe right up to the end of the line. Therefore, when performing the Excel macro, it simply ignores the comments. As a consequence of the previous paragraph, the main purpose of the comment is to display information about this particular macro and help you understand it. Comments can explain things like the purpose of the procedure or what the most recent changes that have been made to the procedure. Item #3: ActiveCell.Select As stated above, this line tells Excel to select the current active cell. More precisely: ActiveCell refers to the current active cell in the active window. Select activates the object on the current Excel active sheet, in this case the current active cell that ActiveCell was referring to. Item #4: ActiveCell.FormulaR1C1 - This is the best Excel tutorial This statement instructs Excel to write This is the best Excel tutorial in active cell. Let's check each of the individual parts of the line: You already know what ActiveCell's purpose is. FormulaR1C1 tells Excel to establish a formula for the object, in this case the current active cell to which ActiveCell refers. The last part (R1C1) refers to the R1C1 notation, in which cell references are relative, not absolute. I explain R1C1 notation in more detail in this tutorial. Remember, however, that at the beginning of this guide, I explain why you should include a relative reference record; and how to do it. This is the best Excel tutorial states that it is a formula (in this case text) that should be placed in an object, in this case an active cell. Item #5: Selection.Columns.AutoFit As I explain above, this particular statement makes Excel automatically fit the active cell column so that the text that the macro typed fits fully into it. The following is the purpose of the various parts of this statement: Choice simply represents the current choice, in this case the active cell. Columns select columns in choice, in this case a column where the active cell is located. AutoFit is a kind of self-evident; it sets the width of the selected columns (as in this case) or the height of the selected rows up to any size, achieves the best fit. Item #6: with... End with statement 1 I mean the following group of lines that are known as With a statement: S selection.Interior . The XlSolid template. PatternColorIndex - xlAutomatic . Color number 255 . TintoEndSchede No 0. PatternTintAndShade - 0 End With The Excel macro that you created has already completed two of the four steps it has to perform: it's scored this is the best Excel tutorial in the active cell. He automatically set the width of the column so that the text he typed would fit properly. As you might expect, the next action Excel does is the color of the active red cell. One would expect that coloring an active cell is a simple step. However, it turns out that Excel needs to take many steps in order to conduct this action. This is the reason why with ... End with statements exist. The main goal with... The end to the statement is to simplify the syntax by following several instructions that all refer to the same object without having to refer to that object each time. In the case of the example used in this beginner manual, this object is an active cell. As you can see in the screenshot below, the main macro that you recorded has two with ... End with statements: C... The end with the statements has the following structure: In the beginning, they must take away with objectExpression. I explore in other tutorials what objectExpression means. At this point, it is enough to know that in the case of the example included in this guide, objectExpression is The Choice. Interior (for the first with ... End with statement) and Choice. Font (for the second c... 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 to which the link was made. After all, they have to say: The end. In the case of the first... End with a statement, each of these objects looks like this: Now that you have a basic understanding of what's up with... End with statement makes, let's look at the first line by line: 1. Line #1: From Selection.Interior. This line tells Excel that it should always relate to the interior of the active cell when making statements that are part of With... End with a statement. How is this achieved? C is the beginning with ... Finish with a statement and tells Excel that subsequent lines of code refer (work with) the object mentioned in that line. Selection.Interior is the object Express, which I mentioned above, explaining the structure with... Finish with the statements. The choice represents the current choice, which in this example is an active cell, while Interior denotes the interior of the object, in this case inside the active cell. 2. Line #2: . XlSolid template. This is the first line with ... End with a statement that makes a reference to the interior of the active cell. He tells Excel to set an internal template active cell to solid. It does it this way: Pattern sets the inner pattern. xlSolid indicates that the pattern should be solid in color. 3. Line #3: . PatternColorIndex - xlAutomatic. This line defines the automatic pattern for the inside of the active cell as follows: PatternColorIndex sets the color of the internal pattern. xlAutomatic indicates that the color should be automatic color. 4. Line #4: . Color number 25. This is a statement that actually announces To Excel what color it should use to fill the interior of the active cell. Color assigns the color of the cell, while the number (in this case 255) determines the color that Best_Excel_Tutorial the macro is red. 5. Line #5: . TintoNdchede No 0. This line orders Excel not to lighten or darken the color chosen to actively fill the cell. TintAndShade sets the color lightening or darkening. When TintAndShade is set to 0 (as in this case), the property is fixed neutrally and therefore there is no light or dime of the color chosen for the active cell. 6. Line #6: . PatternTintAndShade No 0. As you can imagine, this line conveys to Excel that it should not set neither the shade nor the shade of the template for the interior of the active cell. PatternTintAndShade sets the shade and shadow pattern for the inside of the object, in this case the selected cell. 7. Line #7: End with. This line signals Excel end with... End with a statement. Thus, subsequent lines of code refer to a different object than the one to which it is... The end with the statement made. In the case of the example used in this Excel Macro Tutorial for beginners, the end of the first... The End With statement states that subsequent operators do not refer to the inside of the active cell. Item #7: C... End with Statement 2 You learned above that with ... End with a statement and what its overall structure is. So I go straight to line after line explanation of the second with ... End with a macro statement that you probably expect performs the last of the instructions you gave when you created it: changing the color of the active cell font to blue. You will probably be happy to read that this second with... The end with the statement is shorter than the first. Let's start line by line explanation... 1. Line #1: with selection. Font. As I explain above, this discovery with ... End with the operator where With tells Excel that the following operators are working with the object that appears here. In this case, this object is Selection.Font. So what is Choice. Font.? Choice is a current choice that in Best_Excel_Tutorial macro is an active cell, while Font is (not surprisingly) a font. In other words, Selection.Font means the font of text in Cell. So, with Selection.Font basically informing Excel that all the lines of code that are part of with ... End with a statement to make a link to the font of the active cell. 2. Line #2: . Color No. -4165632. This line of code, as you can expect, given the very similar line in the first with ... The end with the statement above, says Excel, what color should be used for the font in the active cell. Color assigns color, while the number (in this case -4165632) is the actual color code that, in this case, is blue. 3. Line #3: . TintoNdchede No 0. This statement is just like one of the lines in With... End with statement above. It instructs Excel not to lighten or darken the color of the font. Because TintAndShade detects color lightening or darkening when it is 0 (as here), Excel does not lighten or darken the color of the active cell font. 4. Line #4: End of C. It's the end with ... End with a statement. Thus, any lines of code below do not make a reference to the font of the active cell. Item #8: End of Sub End application to stop performing something, in this case Sub procedure. This means that once Excel completes this line of code, the macro you created stops working. In other words, this is the end of the code of your first Excel macro. A few final tips on how to learn about Excel Macros If you want to go the extra mile for the purpose of speeding up the learning process about Excel macros, I'll find out some of the final tips below. You can try most of them in the example excel workbook that accompanies this Excel Macro beginner tutorial. You can get immediate free access to this approximate book by clicking on the button below. Change parts of the VBA code to try new things. For example, Change ActiveCell.FormulaR1C1 Is the best Excel tutorial for ActiveCell.FormulaR1C1 I love Microsoft Excel.. You can also change the numbers that indicate the feed box and the color of the font. For example, change. Color No. 255 for . Color No 10 and. Color No.4165632 for . Color 200. Go back to the main Excel window and run the macro again (for example, using the Ctrl and Shift B keyboard shortcut you've assigned) and check what's going on. The results have changed a lot, haven't they? Isn't it interesting how much difference a couple of small items in the VBA code can make? Remove certain operators from the code to see how they affect the macro. For example, what do you think will happen if you remove Selection.Columns.AutoFit? Give it a shot. Go back to Excel and run the macro again with this deletion. What happened? Was that what you expected? One of the best ways to learn the macrococ hundred of Excel code is to repeat the exercise contained in this guide, so I encourage you to do so. How?1. Record Excel macros from the example that appears in this Excel Macro Beginners tutorial. Try new things and see what happens.2 Open the VBE and follow the VBA line to see what the purpose of each statement is. Maybe even better if you have a big enough screen (or two monitors), it's to follow John Walkenbach's advice in excel 2013 Bible and... (...) set up the screen so that you can see the code that is generated in the windows of the VB editor. Read and learn. You can, for example, go through the archives of power tables in order to find all the excel tutorials I've written regarding VBA and macros. Conclusion Again, congratulations! Once you've been through this Excel Macro tutorial for beginners, you've created your first macro and figured out the VBA code behind it. As you could see, setting up a macro with an Excel recorder is relatively simple and can be done in seven simple steps. If your main goal is to just record and play Excel macros, you're ready to go! If your goal is to become a macro expert, I hope this basic guide has given you a good idea of how to record Excel macros and a basic introduction to VBA programming. Also, I hope this Excel Macro Tutorial for Beginners gives you some confidence in your Excel programming abilities. Put into practice the final tips on how to learn about macros that I've provided in the section above. Produce macros, study the VBA code behind them and try different things to see what's going on. If you continue to study and practice Visual Basic for applications, including topics I cover in other Excel VBA tutorials in Power Spreadsheets, you'll soon be able to, among other things, understand much better what you did when creating your first macro and how the macro Best_Excel_Tutorial works. Write and use much more complex and complex Excel macros. Books, links in this Excel Macro Tutorial for Beginner Books in the power table library. Darlington, Keith (2004), VBA for Excel Made Simple. Burlington, Mass.: Made simple books. Harvey, Greg (2013). Excel 2013 All-in-one for Dummies. Hoboken, N.J.: John Wylie and Sons Inc. Elena, Bill (2013). Excel 2013 in depth. United States: Ke Publishing. Helen, Bill and Sirstad, Tracy (2013). Excel 2013 VBA and Macros. United States: Pearson Education, Inc. Walkenbach, John (2013). Excel 2013 Bible. Indianapolis, IN: John Wylie and Sons Inc. Walkenbach, John (2013). Excel VBA Programming for Dummies. Hoboken, N.J.: John Wylie and Sons Inc. tutorial macro excel pdf. tutorial macro excel bahasa indonesia pdf. tutorial macro excel vba. tutorial macro excel 2016. tutorial macro excel untuk pemula. tutorial macro excel pemula. tutorial macro excel youtube. tutorial macro excel 2010