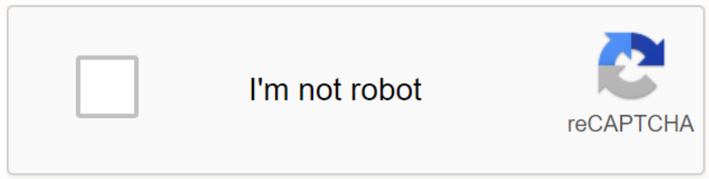
Excel vba for beginners pdf



Continue

Visual Basic for Applications (VBA) is the programming language in Excel and other Office applications. 1 Create a macro: With Excel VBA, you can automate tasks in Excel by typing so-called macros. In this chapter, you can learn how to create a simple macro. 2 MsgBox: MsgBox is a dialog box in Excel VBA that you can use to inform the users of your application. 3 Workbook and worksheet object: Learn more about the Workbook and Worksheet object in Excel VBA. 4 Area object: The range object, which is the representation of a cell (or cells) on the worksheet, is the most important object in Excel VBA. 5 Variables: In this chapter, you'll learn how to declare, initialize, and display a variable in Excel VBA. 6 If the phrase then: Use the If then phrase in Excel VBA to execute lines of code if a specific condition is met. 7 Loop: Looping is one of the most powerful programming techniques. A loop in Excel VBA allows you to go through a series of cells with just a few code lines. 8 Macro errors: In this chapter, you'll learn how to handle macro errors in Excel. 9 String manipulation: In this chapter you will find the most important features for manipulating strings in Excel VBA. 10 Date and time: Learn how to work with dates and times in Excel VBA. 11 Events: Events are actions performed by users that trigger Excel VBA to execute code. 12 Matrix: An array is a group of variables. In Excel VBA, you can refer to a specific variable (element) in an array by using the array name and index number. 13 Function and Sub: In Excel VBA, a function can return a value while a sub cannot. 14 Application object: The mother of all objects is Excel itself. We call it the application object. The application object provides access to many Excel-related settings. 15 ActiveX controls: Learn how to create ActiveX controls, such as <a0></a0> or <a1></a1>. This is a tutorial on writing code in Excel spreadsheets using Visual Basic for Applications (VBA). Excel is one of Microsoft's most popular products. In 2016, the CEO of Microsoft said think of a world without Excel. It's just impossible for me. Well, maybe the world can not think without Excel.In 1996, there were over 30 million users of Microsoft Excel (source). Today, there are an estimated 750 million users of Microsoft Excel. It is slightly more than the population of Europe and 25x more users than there were in 1996.We're a big happy family! In this tutorial, you'll learn about VBA and how to write code in an Excel spreadsheet using Visual Basic.PrerequisitesYou don't need a prior programming experience to understand this tutorial. However, you must:Basic for intermediate knowledge of Microsoft ExcelIf you want to keep up with the VBA examples in this article, you must have access to Microsoft Excel, preferably the latest version (2019), but Excel 2016 and Excel 2013 work fine. A willingness to try new GoalOver this article you learn: What VBA isWhy you would use VBAHow to get created in Excel to write VBAHow to solve some real world problems with VBAImportant ConceptsHer are some important concepts that you should be familiar with fully understanding this tutorial. Objects: Excel is object-oriented, which means that everything is an object - the Excel window, the workbook, a sheet, a chart, a cell. VBA allows users to manipulate and perform actions with objects in Excel. If you have no experience with object-oriented programming, and this is a brand new concept, take a second to let it sink in! Procedures: A procedure is part of the VBA code that is written in the Visual Basic Editor and which performs a task. Sometimes this is also called a macro (more about macros below). There are two types of procedures:Subroutines: a group of VBA statements that perform one or more actionsFunctions: a group of VBA statements that perform one or more actions and return one or more valuesNote: You may have functions that work within subroutines. You'll see later. Macros: If you've spent time learning more advanced Excel features, you've probably come across the concept of macro. Excel users can record macros that consist of user commands/keystrokes/clicks and run them at lightning speed to perform repetitive tasks. Recorded macros generate VBA code, which you can then examine. It's actually quite fun to record a simple macro and then look at the VBA code. Keep in mind that sometimes it can be easier and faster to record a macro instead of hand-code a VBA procedure. For example, maybe you work in project management. Once a week, turn a raw exported report from your project management system into a beautifully formatted, clean report for leadership. You must format the names of the overbudget projects in bold red text. You can record the formatting changes as a macro and run it when you need to make the change. What is VBA? Visual Basic for Applications is a programming language developed by Microsoft. Each program in the Microsoft Office suite is bundled with the VBA language at no additional cost. VBA allows Microsoft Office users to create small programs that work in Microsoft Office software programs. Think of VBA as a pizza oven in a restaurant. Excel is the restaurant. The kitchen comes with standard commercial appliances, like large fridges, ovens and regular ole' ovens - these are all Excel's standard features. But what if you want to make wood-fired pizza? Can't do it in a standard commercial baking oven. VBA is the pizza oven. Yum.Why use VBA in Excel? Because wood-fired pizza is the best! But seriously. Many people spend a lot of time in Excel as part of their job. The time in Excel also moves differently. Depending on the circumstances, 10 minutes in Excel can feel like eternity if you're not able to do what you need, 10 hours can pass by quickly if all goes well. Which is when you have to ask yourself why on earth am I spending 10 hours in Excel? Sometimes those days are inevitable. However, if you spend 8-10 hours every day in Excel doing repetitive tasks, repeating a lot of the same processes, trying to clean up after other users of the file, or even updating other files after changes are made to the Excel file, a VBA procedure just might be the solution for you. You should consider using VBA if you need to: Automate repetitive tasksCreate easy ways for users to interact with your spreadsheetSCryed large amounts of dataGetting Set Up to Write VBA in ExcelFor writing VBA, add the Developer tab to the ribbon so you can see the ribbon this way. To add the Developer tab to the ribbon:Go to Settings > Customize the ribbon under Customize the ribbon and under Main Tabs, select the Developer check box. After you view the tab, the Developer tab remains visible unless you do not clear the check box or reinstall Excel. For more information, see Microsoft Help documentation. VBA EditorNavigate to the Developer tab, and click the Visual Basic button. A new window appears - this is the Visual Basic Editor. In this tutorial, simply familiarize yourself with the Project Explorer pane and the Property Properties pane. Excel VBA ExamplesFirst, let's create a file for us to play in. Open a new Excel fileDelete it as a macro-enabled workbook (. xlsm)Select the Developer Weapons VBA Editorv's rock and roll tab with some easy examples to get you to write code in a worksheet using Visual Basic.Preview #1: Show a message, When users open the Excel workbookI VBA editor, choose Insert -> New ModuleWrite this code in the module (don't insert!):Sub Auto_Open() MsgBox (Welcome to the XYZ workbook.) Exit Sub Save, close the workbook, and reopen the workbook. This dialog box must be displayed. Ta da! How does it do it? Depending on your knowledge of programming, you may have some guesses. It's not very complicated, but there's quite a lot going on: Sub (short for Subroutine): remember from the beginning, a group of VBA statements that perform one or more actions. Auto_Open: This is the specific subroutine. It automatically runs your code when the Excel file is opened - this is the event that triggers the procedure. Auto_Open run only when the workbook is opened manually. It won't run if the workbook is opened using code from another workbook (Workbook_Open will do so, learn more about the difference between the two). By default, a subroutine access is public. This means that any other module can use this subroutine. All examples in this tutorial will be public subroutines. If necessary, you can declare subroutines as private. This may be necessary in some situations. Learn more about subroutine accessmodifiers.msgBox: This is a feature - a group of VBA statements that one or more actions and returns a value. The returned returned is the Welcome to XYZ workbook message. In short, this is a simple subroutine that contains a feature. When can I use this? Maybe you have a very important file that is accessed rarely (say, once in the quarter) but automatically updated daily by another VBA procedure. When it's access, it's by many people in multiple departments, the whole company. Problem: Most of the time, when users access the file, they are confused about the purpose of this file (why it exists), how it is updated so often, who maintains it and how to interact with it. New recruits always have tons of questions, and you have to field those questions over and over and over again. Solution: Create a user message that provides an accurate answer to each of these frequently answered questions. Examples of real worldsUse the MsgBox feature to display a message when there is any event: the user closes an Excel workbook, user prints, a new sheet is added to the workbook, and so on. Use the MsgBox function to display a message when a user must meet a condition before using the InputBox function to retrieve information from userExample #2: Allow the user to perform a different procedureI VBA editor, select Insert -> New ModuleFrag this code in module window (do not insert!): Sub UserReportQuery() Dim UserInput As long as dim responses like integer UserInput = vbYesNo Response = MsgBox(Process XYZ report?, UserInput) Whose Response = vbYes Then ProcessReport End Sub ProcessReport() MsgBox (Thank you for processing the XYZ report.) Exit Sub Save and go back to the Developer tab in Excel, and select the Button option. Click a cell and assign the UserReportQuery macro to the button. Now click the button. This message should show: Click yes or press Enter.Again, tada! Please note that the secondary subroutine, ProcessReport, could be anything. I will show several options in example #3. But first... How does it do it? This example builds on the previous example and has quite a few new elements. Let's review the new things:Dim UserInput For as long as: Dim is short for dimension and allows you to declare variable names. In this case, UserInput is the variable name, and Long is the data type. In plain English, this line means here is a variable called UserInput, and it is a long variable type. Dim Answer As integer: declares another variable called Reply, with a data type integer. Read more about data types here. UserInput = vbYesNo: assigns a value to the variable. In this case, vbYesNo, which displays yes and no buttons. There are many button types, learn more here. Answer = MsgBox(Process XYZ report?, UserInput): assigns the value of the Reply variable to be an MsgBox function and user input variable. Yes, a variable in a variable. If Answer = vbJa Then ProcessReport: this is an If statement, a conditional statement that allows us to say whether x is true, makes y. In this case, if the user has selected Yes, perform the SubroutineProcesReport. When can I use this? This could be used in many, many ways. The value and versatility of this functionality is more defined by what the secondary subroutine does. You might have a file that is used to generate 3 different weekly reports. These reports are formatted in dramatically different ways. Problem:

Each time one of these reports is generated, a user opens the file and changes formatting and charts. so on and so on. This file is extensively edited at least 3 times a week, and it takes at least 30 minutes each time it is edited. Solution: Create 1 button per report type that automatically reformats the necessary components of the reports and generates the necessary charts. Examples of real world examplesCreate a dialog box where the user can automatically fill in certain information across multiple sheetsUse the InputBox feature to get information from the user, which is then filled across multiple sheetsExamp #3: Add numbers to a range with a For-Next LoopFor loops are very useful if you need to perform repetitive tasks on a specific set of values - arrays or ranges of cells. In plain English, say a loop for each x, makes y. In the VBA editor, choose Insert -> New ModuleWrite this code in the module (do not insert!):Sub LoopExample() Dim X As Integer For X = 1 To 100 Range(A & X). Value = X Next X End Sub Save and navigate back to the Developer tab in Excel and select the Macros button. Run the LoopExample macro. This should be done:Etc, until 100. How does it do it? Dim X As integer: Declares variable X as a data type integer. For X = 1 To 100: This is the start of the loop. In short, it tells the loop to keep repeating until X = 100. X is the counter. The loop continues to run until X = 100, performed one last time, and then stops. Range(A & X). Value = X: This declares the range of the loop and what to put in this range. Since X = 1 initially, the first cell will be A1, at which point the loop will put X in that cell. Next X: this tells the loop to run againThen I could use this? For-Next loop is one of the most powerful features of VBA; there are many potential use cases. This is a more complex example that would require multiple layers of logic, but it communicates the world of possibilities in For-Next loops. Perhaps you have a list of all products sold at your bakery in column A, the product type in column B (cakes, doughnuts, or muffins), the cost of ingredients in column C, and the average market prices for each product type in a different sheet. You have to figure out what should be the retail price for each product. You think it should be the price of ingredients plus 20%, but also 1.2% below the market average if possible. A For-Next loop allows you to make this type of calculation. Examples of real worldsUse a loop with a nested if-phrase for only add specific values to a separate array if they meet certain conditionsPrequitary calculations each value in a range, eg calculate extra fees and add them to the valueLoop through each character in a string and extract all numbersRandomly choose a number of values from an arrayConclusionNow that we have talked about pizza and muffins and oh-yes, how to write VBA code in Excel spreadsheet, let's do a learning check. See if you can answer these questions. What is VBA? How do I get set up to start using VBA in Excel? Why and when would you use VBA? What are some issues I could solve with VBA? If you have a fair idea of how you can answer these questions, then it was a success. Whether you are an occasional user or a great user, I hope that this tutorial provided useful information about what can be achieved with just a bit of code in your Excel spreadsheets. Happy coding! Learning ResourcesA a bit about meI'm Chloe Tucker, an artist and developer in Portland, Oregon. As a former educator, I continuously search for the intersection of learning and teaching or technology and art. Reach me out on Twitter @_chloetucker and check out my website at chloe.dev. If you read that far, tweet to the author to show those you care about. Tweet a thank you Learn to code for free. freeCodeCamp's open source curriculum has helped more than 40,000 people get jobs as developers. Getting started