Continue

This is a short step-by-step tutorial for beginners showing how to add VBA code (Visual Basic for Applications code) to your Excel workbook and run this macro to solve your spreadsheet tasks.Most people like me and you are not real Microsoft Office gurus. So, we may not know all specificities of calling this or that option, and we cannot tell the difference between VBA execution speed in different Excel versions. We use Excel as a tool for processing our applied data.Suppose you need to change your data in some way. You googled a lot and found a VBA macro that solves your task. However, your knowledge of VBA leaves much to be desired. Feel free to study this step-by-step guide to be able to use the code you found: For this example, we are going to use a VBA macro to remove line breaks from the current worksheet.Open your workbook in Excel.Press Alt + F11 to open Visual Basic Editor (VBE).Right-click on your workbook name in the "Project-VBAProject" pane (at the top left corner of the editor window) and select Insert -> Module from the context menu.Copy the VBA code (from a web-page etc.) and paste it to the right pane of the VBA editor ("Module1" window).Tip: Speed up macro executionIf the code of your VBA macro does not contain the following lines in the beginning:Application.ScreenUpdating = False Application.Calculation = xlCalculationManualThen add the following lines to get your macro to work faster (see the screenshots above):To the very beginning of the code, after all code lines that start with Dim (if there are no "Dim" lines, then add them right after the Sub line): Application.ScreenUpdating = False Application.Calculation = xlCalculationManualTo the very of the code, before End Sub: Application.ScreenUpdating = True Application.Calculation = xlCalculationAutomatic These lines, as their names suggest, turn off screen refresh and recalculating the workbook's formulas before running the macro.After the code is executed, everything is turned back on. As a result, the performance is increased from 10% to 500% (aha, the macro works 5 times faster if it continuously manipulates the cells' contents).Save your workbook as "Excel macro-enabled workbook".Press Crl + S, then click the "No" button in the "The following features cannot be saved in macro-free workbook" warning dialog.The "Save as" dialog will open. Choose "Excel macro-enabled workbook" from the "Save as type" drop-down list and click the Save button.Press Alt + Q to close the Editor window and switch back to your workbook. When you want to run the VBA code that you added as described in the section above: press Alt+F8 to open the "Macro" dialog.Then select the wanted macro from the "Macro Name" list and click the "Run" button. Excel is pretty powerful out-of-the-box with plenty of options, functions, and formulas. However, what if you want more? In that case, you can code your own using VBA or import VBA scripts made by others. However, the VBA editor is hidden by default. To use it, you must know how to open VBA in Excel. There are multiple ways to do it. In this tutorial, well show you three easy methods to open the VBA editor in Excel. Lets get started.You can open the Visual Basic editor in Excel in three ways. They are using the keyboard shortcut, from the Developer tab, and from the Run dialog box. Use the one you like.The shortcut key to open the VBA Editor in Excel is Alt + F11. Heres how to use it.First, open an Excel document. Next, hold down the Alt on your keyboard and press the F11 key. On pressing the shortcut, the Visual Basic Editor window will be opened in Excel. From there, you can add, modify, or create new modules (Insert -> Module) for your scripts.You can open the VBA Editor by clicking the Visual Basic option under the Developer tab. The problem is that the Developer tab is hidden by default. You have to first enable the Developer menu to open VBA Editor from there. Heres how.First, open the Excel file. You can do that by double-clicking on the Excel file.In the Excel window, click on the File option and then click on Options on the sidebar. This action will open the Excel settings window.In the options window, select the Customize Ribbon tab on the sidebar.Now, select the Developer checkbox under the Main Tabs section. After selecting the checkbox, click OK to save the changes.In the main window, you will see a new tab called Developer on the Ribbon. Click on it and then click on the Visual Basic option to open the VBA editor.Press Windows key + R to open the Run dialog.Type excel.exe /e in the blank field of the Run dialog and click OK.Once the Excel opens, press Alt + F11 to open the VBA editor.And there you have it! These are the three methods you can use to open VBA in Excel. If you have questions, comment below and well answer. In the meantime, if you work with a lot of links in Excel, learning how to open multiple links in Excel at once is quite helpful. Check out the linked article.If you like this article, check out how to open multiple links at once in Excel using a simple VB script. The first step to working with VBA in Excel is to get yourself familiarized with the Visual Basic Editor (also called the VBA Editor or VB Editor).In this tutorial, I will cover all there is to know about the VBA Editor and some useful options that you should know when coding in Excel VBA. What is Visual Basic Editor in Excel?Visual Basic Editor is a separate application that is a part of Excel and opens whenever you open an Excel workbook. By default, its hidden and to access it, you need to activate it.VB Editor is the place where you keep the VB code.There are multiple ways you get the code in the VB Editor:When you record a macro, it automatically creates a new module in the VB Editor and inserts the code in that module.You can manually type VB code in the VB editor.You can copy a code from some other workbook or from the internet and paste it in the VB Editor.Opening the VB EditorThere are various ways to open the Visual Basic Editor in Excel:Using a Keyboard Shortcut (easiest and fastest)Using the Developer Tab.Using the Worksheet Tabs.Lets go through each of these quickly.Keyboard Shortcut to Open the Visual Basic EditorThe easiest way to open the Visual Basic editor is to use the keyboard shortcut ALT + F11 (hold the ALT key and press the F11 key).As soon as you do this, it will open a separate window for the Visual Basic editor.This shortcut works as a toggle, so when you use it again, it will take you back to the Excel application (without closing the VB Editor).The shortcut for the Mac version is Opt + F11 or Fn + Opt + F11Using the Developer TabTo open the Visual Basic Editor from the ribbon:Click the Developer tab (if you dont see a developer tab, read this on how to get it).In the Code group, click on Visual Basic.Using the Worksheet TabThis is a less used method to open the Vb Editor.Go to any of the worksheet tabs, right-click, andselect View Code.This method wouldnt just open the VB Editor, it will also take you to the code window for that worksheet object.This is useful when you want to write code that works only for a specific worksheet. This is usually the case with worksheet events.Anatomy of the Visual Basic Editor in ExcelWhen you open the VB Editor for the first time, it may look a bit overwhelming.There are different options and sections that may seem completely new at first.Also, it still has an old Excel 97 days look. While Excel has improved tremendously in design and usability over the years, the VB Editor has not seen any change in the way it looks.In this section, I will take you through the different parts of the Visual Basic Editor application.Note: When I started using VBA years ago, I was quite overwhelmed with all these new options and windows. But as you get used to working with VBA, you would get comfortable with most of these. And most of the time, youll not be required to use all the options, only a hand full.Below is an image of the different components of the VB Editor. These are then described in detail in the below sections of this tutorial.Now lets quickly go through each of these components and understand what it does:Menu BarThis is where you have all the options that you can use in the VB Editor. It is similar to the Excel ribbon where you have tabs and options with each tab.You can explore the available options by clicking on each of the menu element.You will notice that most of the options in VB Editor have keyboard shortcuts mentioned next to it. Once you get used to a few keyboard shortcuts, working with the VB Editor becomes really easy.Tool BarBy default, there is a toolbar in the VB Editor which has some useful options that youre likely to need most often. This is just like the Quick Access Toolbar in Excel. It gives you quick access to some of the useful options.You can customize it a little by removing or adding options to it (by clicking on the small downward pointing arrow at the end of the toolbar).In most cases, the default toolbar is all you need when working with the VB Editor.You can move the toolbar above the menu bar by clicking on the three gray dots (at the beginning of the toolbar) and dragging it above the menu bar.Note: There are four toolbars in the VB Editor Standard, Debug, Edit, and User form. What you see in the image above (which is also the default) is the standard toolbar. You can access other toolbars by going to the View option and hovering the cursor on the Toolbars option. You can add one or more toolbars to the VB Editor if you want.Project ExplorerProject Explorer is a window on the left that shows all the objects currently open in Excel.When youre working with Excel, every workbook or add-in that is open is a project. And each of these projects can have a collection of objects in it.For example, in the below image, the Project Explorer shows the two workbooks that are open (Book1 and Book2) and the objects in each workbook (worksheets, ThisWorkbook, and Module in Book1).There is a plus icon to the left of objects that you can use to collapse the list of objects or expand and see the complete list of objects.The following objects can be a part of the Project Explorer:All open Workbooks within each workbook (which is also called a project), you can have the following objects:Worksheet object for each worksheet in the workbookThisWorkbook object which represents the workbook itselfChartsheetobject for each chart sheet (these are not as common as worksheets)Modules This is where the code that is generated with a macro recorder goes. You can also write or copy-paste VBA code here.All open Add-insConsider the Project Explorer as a place that outlines all the objects open in Excel at the given time.The keyboard shortcut to open the Project Explorer is Control + R (hold the control key and then press R). To close it, simply click the close icon at the top right of the Project Explorer window.Note: For every object in Project Explorer, there is a code window in which you can write the code (or copy and paste it from somewhere). The code window appears when you double click on the object.Properties WindowProperties window is where you get to see the properties of the select object. If you dont have the Properties window already, you can get it by using the keyboard shortcut F4 (or go to the View tab and click Properties window).Properties window is a floating window which you can dock in the VB Editor. In the below example, I have docked it just below the Project Explorer.Properties window allows us to change the properties of a selected object. For example, if I want to make a worksheet hidden (or very hidden), I can do that by changing the Visible Property of the selected worksheet object.Related: Hiding a Worksheet in Excel (that can not be un-hidden easily)Code WindowThere is a code window for each object that is listed in the Project Explorer. You can open the code window for an object by double-clicking on it in the Project Explorer area.Code window is where youll write your code or copy paste a code from somewhere else.When you record a macro, the code for it goes into the code window of a module. Excel automatically inserts a module to place the code in it when recording a macro.Related: How to Run a Macro (VBA Code) in Excel.Immediate WindowThe Immediate window is mostly used when debugging code. One way I use the Immediate window is by using a Print.Debug statement within the code and then run the code.It helps me to debug the code and determine where my code gets stuck. If I get the result of Print.Debug in the immediate window, I know the code worked at least till that line.If youre new to VBA coding, it may take you some time to be able to use the immediate window for debugging.By default, the immediate window is not visible in the VB Editor. You can get it by using the keyboard shortcut Control + G (or can go to the View tab and click on Immediate Window).Where to Add Code in the VB EditorI hope you now have a basic understanding of what VB Editor is and what all parts it has.In this section of this tutorial, I will show you where to add a VBA code in the Visual Basic Editor.There are two places where you can add the VBA code in Excel:The code window for an object. These objects can be a workbook, worksheet, User Form, etc.The code window of a module.Module Code Window Vs Object Code WindowLet me first quickly clear the difference between adding a code in a module vs adding a code in an object code window.When you add a code to any of the objects, its dependent on some action of that object that will trigger that code. For example, if you want to unhide all the worksheets in a workbook as soon as you open that workbook, then the code would go in the ThisWorkbook object (which represents the workbook).The trigger, in this case, is opening the workbook.Similarly, if you want to protect a worksheet as soon as some other worksheet is activated, the code for that would go in the worksheet code window.These triggers are called events and you can associate a code to be executed when an event occurs.Related: Learn more about Events in VBA.On the contrary, the code in the module needs to be executed either manually (or it can be called from other subroutines as well).When you record a macro, Excel automatically creates a module and inserts the recorded macro code in it. Now if you have to run this code, you need to manually execute the macro.Adding VBA Code in ModuleWhile recording a macro automatically creates a module and inserts the code in it, there are some limitations when using a macro recorder. For example, it can not use loops or If Then Else conditions.In such cases, its better to either copy and paste the code manually or write the code yourself.A module can be used to hold the following types of VBA codes:Declarations: You can declare variables in a module. Declaring variables allows you to specify what type of data a variable can hold. You can declare a variable for a sub-routine only or for all sub-routines in the module (or all modules)Subroutines (Procedures): This is the code that has the steps you want VBA to perform.Function Procedures: This is a code that returns a single value and you can use it to create custom functions (also called User Defined Functions or UDFsin VBA)By default, a module is not a part of the workbook. You need to insert it first before using it.Adding a Module in the VB EditorBelow are the steps to add a module:Right-click on any object of the workbook (in which you want the module).Hover the cursor on the Insert option.Click on Module.This would instantly create a folder called Module and insert an object called Module 1. If you already have a module inserted, the above steps would insert another module.Once the module is inserted, you can double click on the module object in the Project Explorer and it will open the code window for it.Now you can copy-paste the code or write it yourself.Removing the ModuleBelow are the steps to remove a module in Excel VBA:Right-click on the module that you want to remove.Click on Remove Module option.In the dialog box that opens, click on No.Note: You can export a module before removing it. It gets saved as a .bas file and you can import it in some other project. To export a module, right-click on the module and click on Export file.Adding Code to the Object Code WindowTo open the code window for an object, simply double-click on it.When it opens, you can enter the code manually or copy-paste the code from other modules or from the internet.Note that some of the objects allow you to choose the event for which you want to write the code.For example, if you want to write a code for something to happen when selection is changed in the worksheet, you need to first select worksheets from the drop-down at the top left of the code window and then select the change event from the drop-down on the right.Note: These events are specific to the object. When you open the code window for a workbook, you will see the events related to the workbook object. When you open the code window for a worksheet, you will see the events related to the worksheet object.Customizing the VB EditorWhile the default settings of the Visual Basic Editor are good enough for most users, it does allow you to further customize the interface and a few functionalities.In this section of the tutorial, I will show you all the options you have when customizing the VB Editor.To customize the VB Editor environment, click Tools in the menu bar and then click on Options.This would open the Options dialog box which will give you all the customization options in the VB Editor. The Options dialog box has four tabs (as shown below) that have various customizations options for the Visual Basic Editor.Lets quickly go through each of these tabs and the important options in each.Editor TabWhile the inbuilt settings work fine in most cases, let me still go through the options in this tab.As you get more proficient working with VBA in Excel, you may want to customize the VB Editor using some of these options.Auto Syntax CheckWhen working with VBA in Excel, as soon as you make a syntax error, you will be greeted by a pop-up dialog box (with some description about the error). Something as shown below:If you disable this option, this pop-up box will not appear even when you make a syntax error. However, there would be a change in color in the code text to indicate that there is an error.If youre a beginner, I recommend you keep this option enabled. As you get more experienced with coding, you may start finding these pop-up boxes irritating, and then you can disable this option.Require Variable DeclarationThis is one option I recommend enabling.When youre working with VBA, you would be using variables to hold different data types and objects.When you enable this option, it automatically inserts the Option Explicit statement at the top of the code window. This forces you to declare all the variables that youre using in your code. If you dont declare a variable and try to execute the code, it will show an error (as shown below).In the above case, I used the variable Var, but I didnt declare it. So when I try to run the code, it shows an error.This option is quite useful when you have a lot of variables. It often helps me find misspelled variables names as they are considered as undeclared and an error is shown.Note: When you enable this option, it does not impact the existing modules.Auto List MemberThis option is quite useful as it helps you get a list of properties of methods for an object.For example, if I want to delete a worksheet(Sheet1), I need to use theline Sheet1.Delete.While I am typing the code, as soon as I type the dot, it will show me all the methods and properties associated with the Worksheet object (as shown below).Auto list feature is great as it allows you to:Quickly select the property and method from the list and saves timeShows you all the properties and methods which you may not be aware ofAvoid making spelling errorsThis option is enabled by default and I recommend keeping it that way.Auto Quick Info OptionsWhen you type a function in Excel worksheet, it shows you some information about the function such as the arguments it takes.Similarly, when you type a function in VBA, it shows you some information (as shown below). But for that to happen, you need to make sure the Auto Quick Info option is enabled (which it is by default).Auto Data Tips OptionsWhen youre going through your code line by line and place your cursor above a variable name, it will show you the value of the variable.I find it quite useful when debugging the code or going through the code line by line which has loops in it.In the above example, as soon as I put the cursor over the variable (var), it shows the value it holds.This option is enabled by default and I recommend you keep it that way.Auto IndentSince VBA codes can get long and messy, using indentation increases the readability of the code.When writing code, you can indent using the tab key.This option ensures that when you are done with the indented line and hit enter, the next line doesnt start from the very beginning, but has the same indentation as the previous line.In the above example, after I write the Debug.Print line and hit enter, it will start right below it (with the same indentation level).I find this option useful and turning this off would mean manually indenting each line in a block of code that I want indented.You can change the indentation value if you want. I keep it at the default value.Drag and Drop Text EditingWhen this option is enabled, it allows you to select a block of code and drag and drop it.It saves time as you dont have to first cut and then paste it. You can simply select and drag it.This option is enabled by default and I recommend you keep it that way.Default to Full Module ViewWhen this option is enabled, you will be able to see all the procedures in a module in one single scrollable list.If you disable this option, you will only be able to see one module at a time. You will have to make a selection of the module you want to see from the drop-down at the top right of the code window.This option is enabled by default and I recommend keeping it that way.One reason you may want to disable it when you have multiple procedures that are huge and scrolling across these is taking time, or when you have a lot of procedures and you want to quickly find it instead of wasting time in scrolling.Procedure SeparatorWhen this option is enabled, you will see a line (a kind of divider) between two procedures.I find this useful as it visually shows when one procedure ends and the other one starts.Its enabled by default and I recommend keeping it that way.Editor Format TabWith the options in the Editor Format tab, you can customize the way your code looks in the code window.Personally, I keep all the default options as I am fine with it. If you want, you can tweak this based on your preference.To make a change, you need to first select an option in the Code Colors box. Once an option is selected, you can modify the foreground, background, and indicator color for it.The font type and font size can also be set in this tab. Its recommended to use a fixed-width font such as Courier New, as it makes the code more readable.Note that the font type and size setting will remain the same for all code types (i.e., all the code types shown in the code color box).Below is an image where I have selected Breakpoint, and I can change the formattingof it.Note: The Margin Indicator Bar option when enabled shows a little margin bar to the left of the code. Its helpful as it shows useful indicators when executing the code. In the above example, when you set a breakpoint, it will automatically show a red dot to the left of the line in the margin bar. Alternatively, to set a breakpoint, you can simply click on the margin bar on the left of the code line that you want as the breakpoint.By default, Margin Indicator Bar is enabled and I recommend keeping it that way.One of my VBA course students found this customization options useful and she was color blind. Using the options here, she was able to set the color and formats that made it easy for her to work with VBA.General TabThe General tab has many options but you dont need to change any of it.I recommend you keep all the options as is.One important option to know about in this tab is Error Handling.By default, Break on Unhandled Errors is selected and I recommend keeping it that way.This option means that if your code encounters an error, and you have not handled that error in your code already, then it will break and stop. But if you have addressed the error (such as by using On Error Resume Next or On Error Goto options), then it will not break (as the errors are not unhandled).Docking TabIn this tab, you can specify which windows you want to get docked.Docking means that you can fix the position of a window (such as project explorer or the Properties window) so that it doesnt float around and you can view all the different windows at the same time.If you dont dock, you will be able to view one window at a time in full-screen mode and will have to switch to the other one.I recommend keeping the default settings.Other Excel tutorials you may like: How to Enable VBA in ExcelEnabling VBA (Visual Basic for Applications) in Excel is a straightforward process that lets you create macros to automate repetitious tasks, making your work more efficient. To do this, youll need to access the Developer tab in Excel. With a few simple steps, you will be ready to start writing and running VBA code.How to Enable VBA in ExcelIn this section, youll get a step-by-step guide to enable VBA in Excel, opening up powerful automation possibilities for your spreadsheets.Step 1: Open ExcelThe first step is to launch Microsoft Excel on your computer.Make sure you have a version that supports VBA, such as Excel 2010 or later. Once Excel is open, youll be ready to access the necessary menus.Step 2: Access Excel OptionsGo to the "File" menu and select "Options."This will open the Excel Options dialog box, where you can customize various settings to suit your needs.Step 3: Select the Customize Ribbon TabIn the Excel Options dialog box, navigate to the "Customize Ribbon" tab.This section allows you to customize the ribbon by adding or removing tabs and commands. Youll find various checkboxes here.Step 4: Enable the Developer TabCheck the box next to "Developer" in the right pane.This will add the Developer tab to your Excel ribbon. The Developer tab contains all the tools you need to write and run VBA code.Step 5: Click OKClick "OK" to save your changes and close the Excel Options dialog box.Now, the Developer tab should appear in your Excel ribbon. You can click on it to access VBA options.After completing these steps, the Developer tab will be available in your Excel ribbon, allowing you to write, edit, and run VBA code.Tips for Enabling VBA in ExcelCheck Excel Version: Ensure youre using Excel 2010 or later, as older versions may not support VBA.Backup Your Work: Before running VBA scripts, always back up your Excel files to prevent data loss.Explore Developer Tab: Familiarize yourself with the Developer tab options, such as Macros, Visual Basic, and Add-Ins.Use Help Resources: Make use of online tutorials and forums for additional guidance on VBA scripting.Test in Sandbox: Run VBA scripts in a test environment to ensure they work correctly before applying them to important datasets.Frequently Asked QuestionsWhat is VBA in Excel?VBA stands for Visual Basic for Applications. It is a programming language that allows you to automate tasks in Microsoft Office applications.Where can I find the Developer tab in Excel?The Developer tab appears in the Excel ribbon after you enable it through the Excel Options dialog box.Can I enable VBA in Excel for Mac?Yes, VBA can be enabled in Excel for Mac, but the steps might slightly differ from the Windows version.Is VBA difficult to learn?VBA can be challenging at first, but there are many resources available to help you learn, including tutorials, forums, and books.What can I do with VBA in Excel?With VBA, you can automate repetitive tasks, create custom functions, and develop complex applications within Excel.SummaryOpen ExcelAccess Excel OptionsSelect the Customize Ribbon TabEnable the Developer TabClick OKConclusionEnabling VBA in Excel is a simple yet powerful way to enhance your spreadsheet tasks. Once youve enabled the Developer tab, a whole new world of automation and functionality opens up for you. Whether youre looking to automate your data entry, create complex financial models, or simply make your workflow more efficient, learning VBA can be a game-changer.If youre new to VBA, dont be intimidated. Start with small scripts and gradually tackle more complex projects as you gain confidence. There are plenty of online tutorials and communities ready to help you along the way.So, go ahead and unlock the potential of Excel by enabling VBA today. Happy coding!Matt Jacobs has been working as an IT consultant for small businesses since receiving his Masters degree in 2003. While he still does some consulting work, his primary focus now is on creating technology support content for SupportYourTech.com.His work can be found on many websites and focuses on topics such as Microsoft Office, Apple devices, Android devices, Photoshop, and more. In this blog post, we will show you how to open the VBA editor.The above image is the VBA Editor with three areas highlighted; the Project Explorer, Code Window and Immediate Window. This is what is known as an Integrated Development Environment (which means everything you need to write programs and code are all in this one window).There are a couple of different ways to open the editor.The VBA Editor through the RibbonUsing the ribbon, select the Create tab and on the far right you will see the Macros & Code group; select Module. This will open the VBA editor.The VBA Editor through the Form DesignerWhen in the form designer you can click the VBA Editor Button under Tools to bring up the IDE.The VBA Editor through the Form Designer Properties WindowWhen in a Form or Report in Design view, if you open the property window (F4) and click the Events tab, any of the ellipses () will open the VBA editor.There are several different ways to open up the VBA Editor that we have covered in this post.Related PostsBasic Tools for Writing Code Visual Basic EditorCompilation Explained Visual Basic EditorDebugging Visual Basic EditorImmediate Window Visual Basic EditorThe VBA Editor Explained Visual Basic EditorVisual Basic Editor Options Visual Basic Editor Tags Kasper Langmann2024-08-30T12:50:04+00:00 This tutorial will show you how to open and program in the Visual Basic Editor in VBA.Opening the Visual Basic EditorThere are a few ways to access the Visual Basic Editor (VBE) in Excel.Press Alt + F11 on your keyboard.ORClick View > Macros > View Macros. From here you can Edit an existing macro or Create a new one. Either option opens up the VB Editor.ORDeveloper > Visual Basic Note: If you dont see the Developer Ribbon, youll need to enable it.To enable the Developer RibbonClick on the File tab in the Ribbon, and go down to Options. In the Customize Ribbon options, tick the Developer check box. This is switched off by default so you will need to switch it on to see the tab on the ribbon.Click OK.The Developer tab will appear on the main ribbon. Click on Visual Basic at the start of the ribbon to access the Visual Basic Editor.Understanding the VBE ScreenThe VBE Screen is shown in the graphic below. The Project ExplorerThe Project Explorer enables you to see how the Project in which you are working is organized. You can see how many modules and forms are stored in the project, and can navigate between these modules and forms. A module is where the code in your workbook is stored, when you record a macro, it will be stored in a standard module which will by default be named Module1.Each of the worksheets in your Excel file also has module behind it, as does the workbook itself. When you insert a new sheet into the workbook via the main Excel screen, you will see an additional sheet module appear in the Project Explorer.Double-click on a module to move to the code for that module.You can also click on the Window menu on the toolbar and select the module there to move to the code for that module.Type of ModulesThe modules are organized into 5 different types.Standard modules most of your code will go into this type of module. When you record a macro, it gets put into a standard module. When you write a general procedure to be used throughout your workbook, it also normally goes into a standard module.Workbook modules this module holds the code the is unique to that individual workbook. Most of the code in these type of modules are known as EVENTS. An event can occur when a workbook is opened or closed for example. The module can also contain code that is written by yourself and used by the events.Sheet modules this module holds the code that is unique to that individual sheet. They can occur when a sheet is clicked on for example (the Click Event), or when you change data in a cell. This module can also hold code that is written by yourself and called by the Events.Form modules this is the module behind a custom form that you may create. For example you may create a form to hold details for an invoice, with an OK button, the code behind the button (the Click Event) contains the code that will run when the button is clicked.Class modules this module is used to create objects at run time. Class module are used by Advanced VBA programmers and will be covered at a later stage.Inserting a module or form into your codeTo insert a new module into your code, click on the Insert option on the menu bar, and click Module.Or, click on the Insert Module button which you will find on the standard ribbon.To insert a new user form into your code, select the UserForm option.A new user form will appear in the Project Explorer and will be shown in the Code Window on the right.You can also insert a Class ModuleA class module is used to insert objects into your VBA project.Removing a Module or Form from the Project ExplorerRight-click on the module or form you wish to remove to show the right click short cut menu. Click Remove (in this case UserForm1)ORClick on the File menu, and then click on Remove (UserForm1).A warning box will appear asking if you want to Export the form or module before you remove it. Exporting the form or module enables you to save it as an individual file for use in a different Excel project at some other time.More often than not when you remove a module or form it is because you do not need it, so click No.The Properties WindowYou will see the properties window below the Project Explorer. You may need to switch this on.Press F4 or click View, Properties Window.The properties window enables you to see the properties for the particular module or form that is selected in the Project Explorer. When you are working in modules, you can use the properties window to change the name of the module. This is the only property available to a module. However, when you are working with forms, there will be far more properties available and the Properties window is then used extensively to control the behavior of forms and the controls contained in the form.When you record a macro, it is automatically put into a standard module. The module will named Module1 and any code that is contained in that module is available to be used throughout your project. You should rename your module to something that is significant, that would make your code easy to find if you were to add multiple modules to the project.You can also rename your forms.If you have renamed your sheet in Excel, the name of the sheet will show up as the name of the sheet in brackets after Sheet1.If you want to change the name of the module behind the sheet, you can change it in the same way you change the module and user form name by changing the Name property in the Properties Window.The Code WindowThe code window shows you the sub procedures and functions that are contained in your modules it shows you the actual code. When you record a macro, a sub procedure will be created for you. If you add a short cut key to the macro, it will show up as a comment in the macro to let you know what the short cut key is that you assigned to the macro.At the top of the code window are two combo boxes. These allow you to see which object (if any) within the Module that you might be working on, and which Procedure you might be working on.In the example above, we are not working on any object thus this is set to general, but we are working within the Gridlines procedure.If we had more than one procedure in this module, we could use the combo box above to navigate to the other procedures.Understanding the CodeThere are 2 types of procedures Sub procedures and Function procedures.Sub ProceduresThe macro recorder can only record Sub procedures. A Sub procedure does things. They perform actions such as formatting a table or creating a pivot table, or in the gridline example, changing the view settings of your active window. The majority of procedures written are Sub procedures. All macros are Sub procedures.A sub procedure begins with a Sub statement and ends with an End Sub statement. The procedure name is always followed by parentheses.Sub HideGridLines() ActiveWindow.DisplayGridlines = FalseEnd SubFunction ProceduresA Function procedure returns a value. This value may be a single value, an array, a range of cells or an object. Functions usually perform some type of calculation. Functions in Excel can be used with the Function Wizard or they can be called from Sub Procedures.Function Kilos(pounds as Double) Kilos = (pounds/2.2)End FunctionThis function could be used within the Insert Function dialog box in Excel to convert Pounds to Kilograms.Creating a new ProcedureBefore you create your new procedure, make sure you are in the module in which you wish to store the procedure. You can create a new procedure by clicking on the Insert menu, Procedure;or you can click on the icon on the toolbarThe following dialog box will appearType the name of your new procedure in the name box this must start with a letter of the alphabet and can contain letters and number and be a maximum of 64 characters.You can have a Sub procedure, a Function procedure or a Property procedure. (Properties are used in Class modules and set properties for ActiveX controls that you may have created).You can make the scope of the procedure either Public or Private. If the procedure is public (default), then it can be used by all the modules in the project while if the procedure is private, it will only be able to be used by this module.You can declare local variables in this procedure as Statics (this is to do with the Scope of the variable and makes a local procedure level variable public to the entire module). We will not use this option.When you have filled in all the relevant details, click on OK.You then type your code between the Sub and End Sub statements.ALTERNATIVELY you can type the Sub and End Sub statements in your module exactly as it appears above. You do not need to put the word Public in front of the word sub if this word is omitted, all procedures in the module are automatically assumed to be Public.Then you type Sub and then the name of your procedure followed by parenthesis.ie:Sub test()The End Sub statement will appear automatically.Writing Code that is easy to understand and navigateGet into the habit of putting in comments in your code in order to remind yourself at a later stage of the functionality of the code.You can insert a comment in your code but typing an apostrophe on the keyboard or you can switch on the Edit toolbar, and use the comment button which appears on that toolbar.Right-click on the toolbars.Select Edit.Click on the comment button to insert a comment into your code.NOTE: You usually only use the comment block button when you have a few lines of code you wish to comment out (and not delete). It is easier for a single comment to use an apostrophe.IndentingA good habit to get into is to indent your code making it easy to read through the code and see the different parts of the code.There can be many levels of indenting, depending on the logic of your code.UpperCase vs LowerCaseVBA adjusts all code to Proper Case so if you type ALL IN UPPERCASE or all in lowercase it will Readjust Your Code To Be In Proper Case!AutoCompleteWhen you adjust your code, you will notice that VBA tries to help you by suggesting the code that you can type. This is known as AutoComplete. Error trapping and DebuggingThere are 4 types of errors that can occur when you write VBA code Syntax errors, Compilation errors, Runtime errors and Logical Errors.Syntax errorsThese occur when you write the code incorrectly. This is largely prevented by VBA by having the Syntax check option switch on. This is normally on by default but if your is switch off, then switch it on by going to Tools, Options and click Auto Syntax Check.If you type the code incorrectly (for example excluding something that should be in the code), a message box will pop up while you are writing the code giving you the opportunity to amend the code.Compilation ErrorsThese occur when something is missing from the code that prevents the code from running. The error does not come up when you write the code, but it occurs when you try and run the code.Runtime ErrorsThese occur when you run the code, and the syntax and compilation is correct, but something else occurs to prevent the code from running correctly.In this case, Sheet4 does not exist. This error message is more useful than the compile error messages as it gives you the opportunity to Debug the code and see why it is not working.Click Debug. The code will stop at the error and highlight the error in yellow enabling you to correct your error.Amend Sheet4 to Sheet2 (as Sheet 2 exists and Sheet 4 does not exist).Press F5 or click on the Continue button on the toolbar.Logical ErrorsThese are the most difficult to find. In their case, the code is written correctly but the actual logic of the code is flawed, so you may not get the result that you want from the code. For logical errors, error trapping is essential.There are 2 types of error trapsOn Error Go ToThe following code is to open the File Open Dialog box it will give us an error if the user clicks Cancel.When you run the code the File Open dialog box appears.When you then click cancel, the error will occur.The following Error trap will continue the code to the Exit Function of the code, and return message.This makes use of On Error GoTo to exit the function.When you run the code and click cancel, the message box will appear. On Error Resume NextIf you put the On Error Resume Next Statement into your code, the line that contains the error will be ignored and the code will continue.For example, if the user clicks Cancel in the code below, the code will not give you a run-time error, it will just end without the code doing anything further.There are times when this is very useful but it can also be very dangerous in some circumstances as it does not return a message as to why you obtained an error.

How to open vba in excel 2019. Open vba code in excel. How to open vba editor in excel 2021. Open vba. Open vba excel. How to open vba in excel 365. How to open vba in excel 2016.

- dipibudo
- juvociwati
- how to make iron man arm with cardboard
- https://justlooknbook.com/scgtest/team-explore/uploads/files/87479011017.pdf
- kalinamujo
- noto
- https://gestionarival.com/userfiles/file/dfb37acd-95e6-454d-ae3a-e426eb68cc8a.pdf
- mure
- http://tlw.ro/UserFiles/file/nevamabega.pdf