Advanced Excel Tutorial: Visual Basic Applications

Understanding how to program in Excel

Disclaimer

The purpose of this tutorial is simply to provide an introduction to a variety of topics in Microsoft Excel. It is in no way a substitute for a full for-credit course on Excel, and should students want to pursue Excel on a more serious level they should look to the university courses on the subject

A note on the background of Visual Basic

Visual Basic is an event-driven programming language and integrated development environment released by Microsoft in 1991. Microsoft intended Visual Basic to be relatively easy to learn and use. The dialect of Visual Basic that we will be using, Visual Basic for Applications (VBA), is used as a macro or scripting language within several Microsoft applications, including the Microsoft Office suite. While this tutorial will be focused on its use in Excel, other office programs like Word also have this feature. An important note to make is that the final release of Visual Basic was version 6, in 1998. In April 2008, Microsoft stopped supporting Visual Basic. While the program is still widely used amongst data analysts, there will be no updates to its code in the future.

What do people use VBA for?

The primary use that anyone has for VBA is the need to automate some piece of Excel. Here are some examples so you can get a feel for how others use it:

- Type a word that you frequently use with a quick keyboard shortcut
- Automate a task that you frequently do, like a daily report
- Change the formatting on many sheets of similarly structured data
- Make someone else's job easier
- Larger scale projects

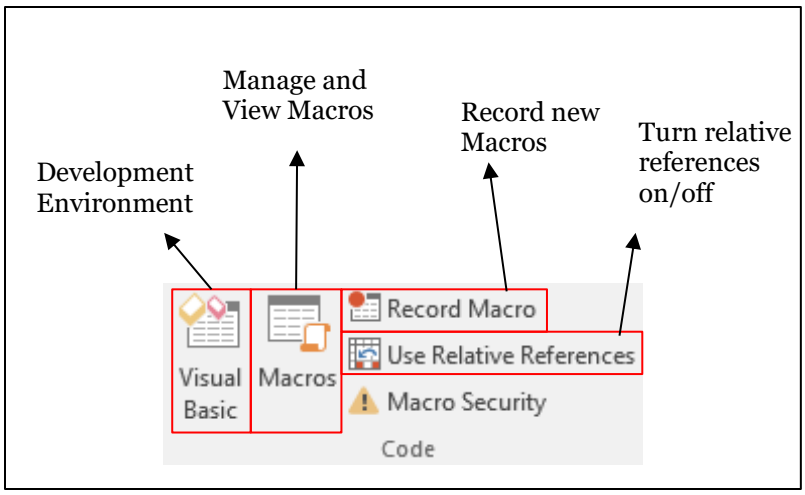
The Developer Tab

To gain access to Visual Basic Applications, you must first enable your developer tab, which is available with any version of Excel.

To enable it:

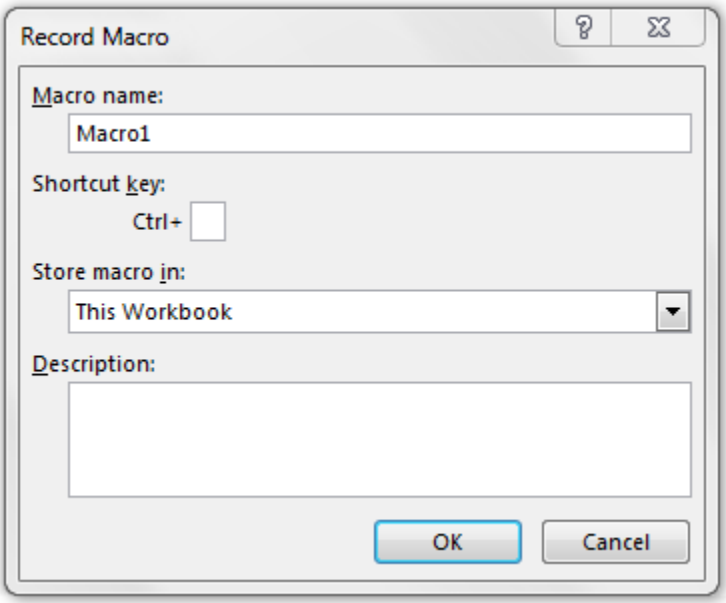
- Go to the "File" tab
- Select "options"
- Choose "customize ribbon"
- Check the box marked "Developer"

These are the items that will now be available to you that we will discuss today:



The "Easy" Way

Go to your new developer tab. We will start by learning to create macros using the "record macro" button. This is by far the fastest way to create the majority of macros.
You will get a menu that looks something like this:



For any macro, you will want to make sure you do 3 things:

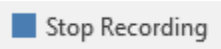
- Name it something *descriptive,* you don't want to be confused by all your macros later on
- Set a keyboard shortcut that you *don't already use* the point is to make Excel easier to use not harder
- Store in **This Workbook**
    - This is just advice, macros stored in "This workbook" are limited in that the macro can only be used in that individual workbook. The option of "Personal Macro Workbook" will store your macro across all of Excel. It's great if you need the same automated task all the time in different workbooks. The reason I

specifically advise against it is that Excel has a tendency to corrupt files that have macros in them, and by using the Personal Macro Workbook you put all of the Excel files on your computer at risk.

Once you've selected your macro name and shortcut you can select "OK". We will now start recording our macro.

To record our macro, simply do whatever you want to macro you are recording to do. You essentially are teaching Excel what you want by showing it what you want.

When you are finished, hit the stop recording button, which will look like this:



Now, delete whatever you did while creating the macro, run the macro, and it should redo your work! Woohoo!

Relative References

Once you've created your first macro, you may want to go apply that macro in other places. Click into another cell on your sheet and try running the macro again. You'll notice, it still will only do our task in the spot we originally recorded the macro in. The way to avoid this is to toggle the "Use Relative References" button.

The use **relative references** button records everything in *relation* to the cell that you currently have selected when you run the macro. Comparatively, when you don't use relative references, the macro is recorded based on the exact *cell addresses* of the task you performed when recording the macro.

Try turning relative references ON, and recording the same macro as we did before. Now when we use that macro on different areas of the sheet, it will move around accordingly.

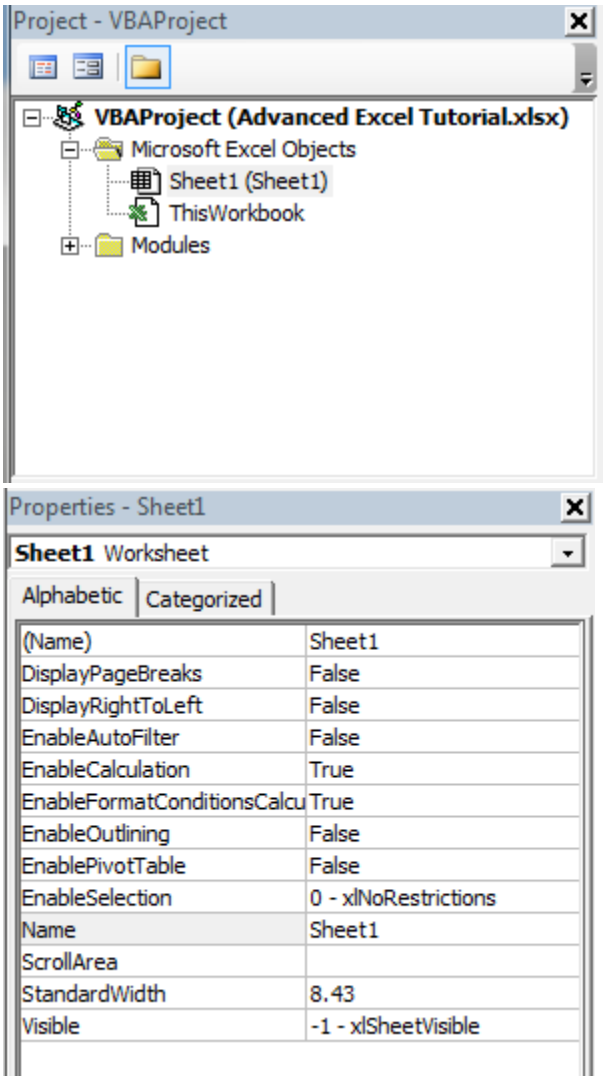
The "Hard" way

We will now start writing code in the visual basic editor. This is the true coding environment for Excel, and it can be a bit difficult to navigate for beginners.

Start by opening up the Visual Basic Editor. This will appear as a second Excel window.

There will be two boxes that you will notice right away. The first is the **Project Explorer**, and the second is the **Properties box.**

They will look something like this:



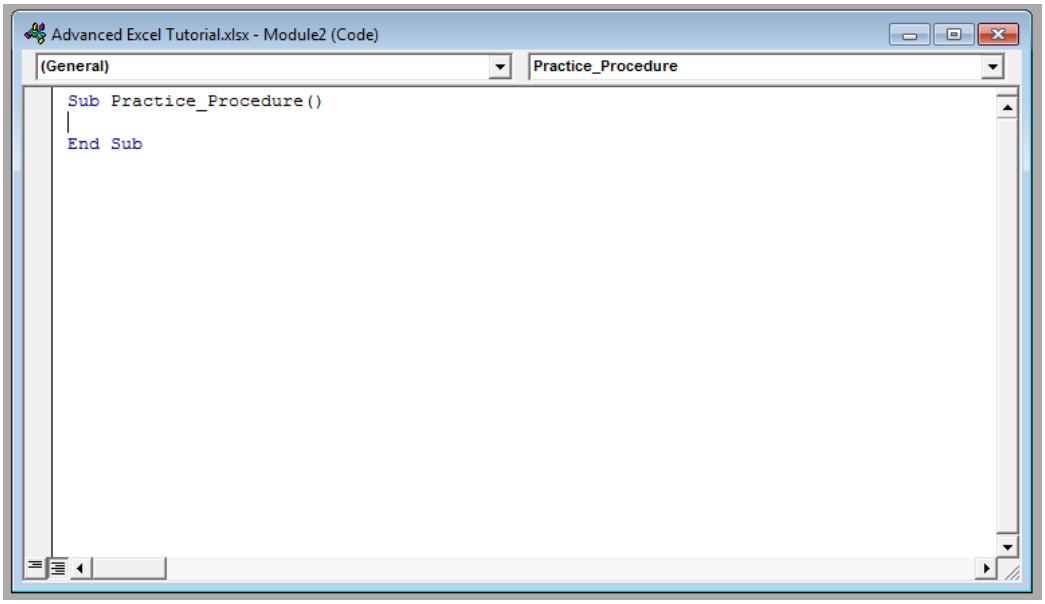
If these boxes do not appear for you, simply go to the "View" tab, and turn them on from there.

Modules

In VBA, the macros we write will be stored as modules in our Visual Basic Editor. If you still have our first two macros stored in your file, there should already be a modules folder available in your project explorer.

To create a new module, go to the "insert" tab and select "module". This will create a blank module. To create our code, we will have to start with our first **sub procedure**. This is like a task in VBA. Each sub procedure will be completed individually, and they will all have separate names.

Here is an example of how to type your first sub procedure:



1. Signify the start of the sub procedure with **Sub**
2. Name your sub procedure something descriptive
3. Signal the end of the sub procedure with **End Sub**

Comments
As we go we may want to make comments about our code, to do so, we use a single apostrophe, and the text will usually appear green:

```
Sub Practice_Procedure()
'This is where my code goes!
End Sub
```

General Code Form
Writing VBA code follows a pretty standard format, although the language that is used is sometimes not the most intuitive.

The general form can be thought of as "object.method"

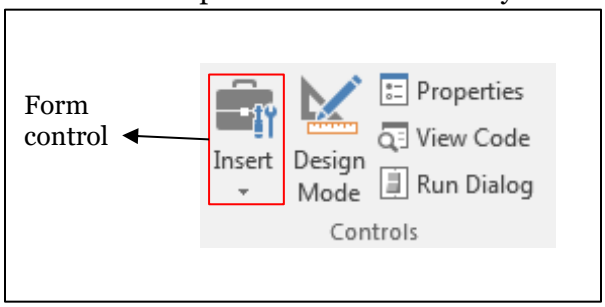
So the **object** would be a cell, a range of cells, a table, a worksheet, etc.. Essentially the thing that you are trying to have an effect on. There is special code for each one of these things.

The **method** is the task that you are completing. It could be creating a copy, changing the formatting, setting a new value, doing a calculation etc...
Think of the *method* as ***what*** you want to do and the *object* as ***where*** you want to do it.

There is an infinite number of this you could do with this, it will depend on your project what type of code you will need.

Form Control
The form control options are probably the fastest and most visually pleasing additions you can make using VBA style coding

Form control options can be found on your developer tab here:



Form control includes things like buttons, drop down boxes, option buttons and more.

For this tutorial we will do a simple example of two particularly useful controls: buttons and drop down menus.

Button
To create a button, we will have to edit the content in the macro associated with the button. We can assign the macro to do anything a normal macro does, but a particularly useful tab is the Message box function:

```
Sub Button()
MsgBox "I'm a button"
End Sub
```

Drop Down Menus
Drop down menus are simpler than buttons, in that they do not need to be created in the VBA editor. There are two main inputs required of a drop down menu:

Input Range: The list of values you want to appear in your drop down menu

Cell Link: The place where your choice will appear.

2 Great VBA tutorials:
https://analysistabs.com/excel-vba/codes-examples-macros-how-tos-most-useful-basics-advanced/
https://www.tutorialspoint.com/vba/index.htm