Creating Custom Functions in Excel

Publication Date: June 2016
Creating Custom Functions in Excel

Course Description: This course covers how to create a custom function (sometimes called a user-defined function) in Excel. A custom function is a calculation that the user creates and names. This custom function is then stored in Excel and can recalled when it is needed. In this course we will be using the Visual Basic Editor and using a couple of VBA keywords; however this is NOT a VBA or programming course. You will create a couple of different custom functions and then see how to create an ADD-IN file so that these functions can be used in all your workbooks.

Course Objectives

- Recognize how to find and access the Visual Basic Editor
- Identify the components of custom functions and how to use them
- Recognize how to create, save and remove custom functions
- Recognize the implications of creating custom functions and how they work

Category: Computer Science/Specialized Knowledge

Level: Intermediate to Advanced Excel user

Prerequisites: Excel 2007 through Excel 2016. Should have a basic understanding of an IF statement.
OVERVIEW
Before we jump into this topic, let’s make sure everyone is on the same page and that everyone is clear as to what a custom function is. Some people refer to it as a custom function while others call it a User Defined Function while others, with no life, affectionately refer to it as a UDF.

Whatever you want to call it – they all do the same thing. They allow you to create and save a formula that you have written. Now, given the fact that Microsoft offers over 300 pre-defined functions do we really need more? Unfortunately the answer can be “yes” for heavy spreadsheet users. So, the purpose of a custom function is to allow you to create functions that do not currently exist in the “pre-packaged” function categories that are included with Excel.

A custom function simplifies the time spent working on worksheet formulas. Custom functions are for people who frequently create the same complex formula over and over. In essence, you are creating and naming your own formula so, it would really be better named “Custom Formula” but, hey, we will use Microsoft’s naming convention of Custom Function. With a custom function, you create the formula or function once, name it, store it and then you select it whenever you need it. So, instead of selecting a pre-defined Microsoft function, such as COUNT or IF, you select one that you created from the same Insert Function dialog box.

Let me give you an example. I am always comparing YTD sales against the previous year on a percentage basis, and every time I have to think about the formula and try to remember if Prior Year or Current Year is in the denominator. Then I always have to “pre-test” it to make sure that I wrote the formula correctly. Obviously, I have a poor memory as I am sure some of you would just automatically know it but… anyway- it would be so much easier if I could just create the formula (Current Year Sales-Prior Years Sales)/Prior Years Sales, name it and then simply select it when I want to use it in a spreadsheet instead of recreating it every time. That is, in a nutshell, what a custom function is. It saves you time, frustration and reduces typing errors.

I always shied away from creating custom functions as I assumed it would be time-consuming and difficult. Guess what- I was wrong! Well - partially wrong as it really depends on the complexity of the calculation you are creating. If you are creating some really complex calculation then, yes, it is time consuming and difficult and will probably require some VBA (Visual Basic Application language); however, in many cases the formulas we use are not that complex.

In this course we are going to cover the basics of how to create a custom function, which is where everyone needs to start, and you can progress from there if need be. We will NOT be writing VBA procedures. However, we will be using the VBA Editor. For those who use VBA, I will cover a basic VBA keyword or two that you might want to play around with on your own. I have also added a link to ozgrid.com, a website containing a lot of custom functions that require at least a bit of VBA knowledge for those who want to expand their knowledge and/or do a little cutting and pasting. This is NOT a VBA course – just to be clear.

We are going to work though how to create 3 custom functions. The first one, which is totally simple, is a function that multiplies a single value by 2. Not terribly exciting but I wanted to start with something very basic so the steps are clear. Then, we will progress to the sales formula, a percentage year to year variance, which I discussed above. The third is a simple IF. We will also discuss how to save a custom function so that it can be used in any Excel workbook that you open. Finally, we will discuss the basics of how to later edit or remove a custom function and an ADD-IN.
Creating Custom Functions in Excel

DEVELOPER TAB
Okay, I did warn you that we would be using the VBA Editor which can be intimidating if you have never used it before. However, don’t worry as we are not going to be writing a lot of code, but custom functions do use the Visual Basic for Applications (VBA) programming language and we are going to need to put our custom function inside its procedures.

Let’s jump in and find the Editor. Now, the Editor is not part of the standard Excel installation so chances are that if you look at your formula bar you will not see a Developer tab.

If you do have it that’s great – you can ignore this little section and jump to the next. For those of you who do not have the Developer tab, let’s add it to the Ribbon.

Excel 2010/Excel 2013/2016 users:
1. Click the File Command and then look down the list.
2. Click Options.
3. Select **Customize Ribbon**.

4. In the **Main** tab on the right side of the dialog box select **Developer**.

5. Click **OK**.

**Excel 2007 users:**

1. Click the **Office Button**

2. Click the Excel **Option** button at the bottom, then you will enter the Excel **Option** window.

3. Click the **Popular** button at the left.
Creating Custom Functions in Excel

4. Under Top Option for Working with Excel, check the **Show Developer** tab in the Ribbon option.

You should now see the Developer tab on the Ribbon. If you click on the Developer tab, it should look similar to the one below.

**BASIC RULES**

Let’s discuss some basic rules before we really get into this. Don’t worry, I will mention most of these again as we go through creating some custom functions.

- First, each custom function needs to begin with a **Function** statement and end with an **End Function** statement.
- Typically a Function statement specifies one or more arguments. For example, =SUM(A1:A15) is telling Excel to sum cells A1 through A15. Only a few functions such as =NOW() do not contain arguments and they are generally volatile functions meaning that they continually change.
- Custom functions are considered safer than macros as they are very limited and basically just return a value to a formula or expression. In other words, they cannot:
  - Insert, delete or format cells in a workbook
  - Change the value of another cell
  - Change the structure of a workbook
- All custom functions should be documented. There are couple of different ways to document, but the key thing to remember is that they need to be documented. If you create 3 or 4 functions you may not remember what one of them does in a year or so, particularly, if you don’t use them frequently. Another consideration is, when you get promoted, the new person handling the workbook would probably like to know what the heck the function is or does.
- Custom Functions trigger a “macro security flag”, so if you have a corporate policy which does not allow worksheets with enabled macros then this may not work for you. Now, a custom function is not a macro. However, whenever Excel opens up a file containing a custom function, it treats it like a macro.

What this means is that when you open a file, you will get a security notice asking if macros can be enabled. This is a generic message that pops up. Generally, if you know where the file came from then most people are comfortable opening it. However, the warning does intimidate some people. **If you do not know the source, it is NOT recommended that you open the file.**
VISUAL BASIC EDITOR
Now, let’s get started
First we need to open the Visual Basic Editor

1. Click on the Developer tab.

2. Press Alt+F11 which will open the Visual Basic Editor or click the first icon which is called Visual Basic.
   The Project Window opens in the top left side of the window and the Properties are shown down in the bottom left window.

3. Click Insert on the menu and then select Module.
   Make sure that you selected Module and NOT Class Module.
   We are placing the code in a Module because a module is not associated with any specific workbook. There are 4 different types of modules. Chip Pearson does a good job explaining the differences in case you are interested. The explanations can be found at http://www.cpearson.com/excel/codemods.htm
   If you have Solver or a General Ledger add-in, then Module 1 or more may be already taken so if you start off with a different Module number instead of Module1 don’t worry about it.
If you make an error when writing the code, Excel is not very helpful and but will usually tell you it has a compile error and display a yellow arrow to the left of the line it has an issue with. If you make changes after you have tried to run the function you may see a dialog box telling you that your action will reset the project.

Click OK and it will accept whatever changes you have made and then determine again whether the syntax is correct. In other words, it accepts your changes and tries to rerun the function. Personally, I found my biggest challenge was spelling everything correctly. In that case you often don’t get an error message – you just get a 0 or an error message in the spreadsheet. Also, the VBA Editor does not like spaces or special characters.
CREATING A CUSTOM FUNCTION - DOUBLE VALUE

Doubling a Value - Formula

Let’s start with a very simple formula. We are going to have Excel return a value that is twice as large as the cell we select. In other words, we are going to double the value of the selected cell.

**Every Custom Formula needs to start with the word “Function”!**

We will name this custom function Doubleval. We are also going to tell Excel that the cell we select as the value should be multiplied by 2 or doubled – hence the name of the function – clever eh?

Enter the following information as shown below:

1. Type `Function Doubleval (value)` on the first line of the window.
   This is telling Excel that the name of the function is Doubleval and that the function works on a single value.
2. Press Enter.
   As soon as you press Enter, notice that Excel adds `End Function`. Everything else you type must be placed before the `End Function` variable.
3. Type `Doubleval=value*2` and make sure that your cursor is in between the `Function` and the `End Function` variable.
   This statement is telling Excel that the value of the cell selected should be multiplied by 2. We are telling Excel that the formula Doubleval, when called into action, will double the selected value or cell that has been selected in the worksheet.
4. Click on the Excel icon to return to the Excel worksheet or press **Alt + F11**
   Alt+F11 acts as a toggle and will take you back and forth between the Excel worksheet and the VBA Editor.
5. Type 500 in cell A1 and then click in cell B1.
6. Type =Doubleval and you should see the formula already displaying off of your cursor. Either double-click to select it or type it out =Doubleval(

7. Click on cell A1 and press Enter.

The resulting answer should be 1000. Easier than you thought – right? I did not need to select the ending parentheses as Excel understood it needed to be included as it was part of the syntax: =Doubleval(value).

If something is not working, make sure that everything you typed is between the Function and the End Function keywords. End Function should be the last line of text.

Let’s move on to something a little more complex.

**CREATING A CUSTOM FUNCTION - Sales Variance PY to CY**
This is the formula that started my quest for creating Custom Functions. We want to compare current years and prior year sales as a percentage variance.

Now the formula we want to create is (Current Year Sales-Prior Year Sales)/Prior Year Sales and while it would be nice to just type that into the module, it is unfortunately not that easy, as we need to define all the different components for Excel. First, we have to define Current Year for Excel. We are going to shorten it to CY as I am a bit lazy and I try to reduce typing as much as possible. Also CY is easy to remember and it is easier and faster to type.

1. Switch to the Visual Basic Editor. If it is not visible, then press Alt+F11.
2. Click Insert and then Module so that we can create a second module. You could immediately start typing under the first custom function code and Excel would treat it as a second function, however, that way the code for two different functions is in the same module window.

   It is generally recommended that you keep them in separate modules particularly if you are doing anything complex as it is easier to view and to audit.

   Notice that we now have Module 1 and Module 2.

3. In the Module window type Function CY(value1) and press the Enter key.

As soon as you press the Enter key, Excel displays the typed entry and also adds its function procedure key words End Function. Excel did this because we entered the keyword Function on the first line. Since we defined it as a function, Excel will as well.

Everything that you type must be between the first Function line and the End Function line.
4. Enter the following information **above** the End Function line:

```
CY=Value1
Function PY(Value2)
    PY=Value2
Function Svar(CY,PY)
    Svar=(CY-PY)/PY
End Function
```

When you are finished your screen should look similar to the one below:

It may look slightly different as Excel spaces some of the lines differently.

A lot of people recommend that you indent using the Tab and the Shift+Tab to make your entries more readable, however, this was pretty simple so I didn’t bother. However, you can indent here if you wish. Let’s take a look at what this is going to do.

- Line 1 reads as **Function CY(Value1)**. This tells Excel that we are creating a function called CY (CY represents Current Year) and that is comprised of a single value.
- The Second line tells Excel that the first cell we select will be assigned or considered to be the CY value. A placeholder if you will.
- The Third and Fourth lines tell Excel that we are creating a new function called PY (which we want to represent Prior Year) and that the second cell we select will be considered as assigned or belonging to PY. Again, a placeholder.

Notice that CY is defined as Value1 and PY is defined as Value 2. You need to do this so that Excel knows which value to put in which placeholder. This is telling Excel that the first value selected is the CY while the second is PY.
Creating Custom Functions in Excel

• Line 5 is where we begin to provide Excel with some information. First, we are telling Excel that we are creating a function named Svar (short for Sales Variance) which is also a new function just as CY and PY are. However, we are also telling Excel that Svar uses both the CY value and the PY value that we just created in its formula.

• Line 6 tells Excel the syntax of Svar and how it will calculate. Excel understands that it needs to take the CY value and the PY value and use them to create a resulting solution for the Svar function. In English we would say that Svar = (Current Year minus Prior Year) divided by Prior Year. The CY and PY serve as placeholders for the values that you are going to select.

• Line 7 End Function needs to be the last line and it tells Excel that we are finished.

Tip: As you may have already noticed, this VBA Editor is very picky and will let you know immediately if it doesn’t like something you typed. If you press the Enter key too many times, it will add an End Function as it thinks you are done. You only want one End Function at the end of each module so if it adds more of them, make sure to delete them.

Okay, let’s try it out.

When you start typing in the new function, Excel looks for the name Svar and determines that it is not a Microsoft function and it then recognizes it as a procedure in a module.

5. Click on the Worksheet icon or press Alt+F11 as this will toggle you back and forth.
   a) In the Excel worksheet, type what is shown in the screenshot below:

   ![Excel Worksheet](image1)

   b) After you enter the information format cells C2:C4 as a percentage.

6. Click in cell C2, start typing =sv and you should see the new function that we just created display. To have Excel type the function, double-click on the Svar and the function will display. Alternatively, you can type =Svar( in yourself.
Creating Custom Functions in Excel

7. Click on A2 and then type in a comma and then select B2.

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>B</td>
<td>C</td>
<td>D</td>
</tr>
<tr>
<td>1</td>
<td>2016 YTD Sales</td>
<td>2015 YTD Sales</td>
<td>Variance</td>
</tr>
<tr>
<td>2</td>
<td>1200</td>
<td>600</td>
<td>=Svar(A2,B2)</td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

8. Press Enter (Excel will put in the closing parentheses) and voila! – a percentage should display.

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>B</td>
<td>C</td>
<td>D</td>
</tr>
<tr>
<td>1</td>
<td>2016 YTD Sales</td>
<td>2015 YTD Sales</td>
<td>Variance</td>
</tr>
<tr>
<td>2</td>
<td>1200</td>
<td>600</td>
<td>100%</td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Let’s try an alternative way so that you can see how this custom formula can really shine.

1. Click in cell C4.

2. Click the Fx button on the formula bar.

3. In the dialog box click on the drop down arrow beside Or select a category.

4. Select User Defined category and then select the Svar function.
5. Click **OK**.
6. In the **Cy** section of the dialog box click on **A4**.
7. In the **Py** section of the dialog box click on **B4**.
8. Click **OK**.

You should see an answer of 20% in C4.
DOCUMENTATION
I promised you I would come back to this. Documentation really is extremely important and it is easy to do. First, use descriptive names because, let’s face it, you are going to forget what this function does. Svar makes sense to me but may not to someone else. In addition, or as an alternative, you can add comments right into the module so that you can refer to them when you forget what the heck this function does. Comments are easy- all you have to do is type an apostrophe and Excel ignores everything after that and displays it in green. Comments are also helpful if you need to make changes to the formula, particularly if it is a complex procedure.

```
' This function calculates sales variance as a percentage
' The calculation is (Current Year Sales minus Prior Year Sales)/Prior Year Sales
' Created by Patricia McCarthy
Function CY(Value1)
CY = Value1
Function PY(Value2)
PY = Value2
Function Svar(CY, PY)
Svar = (CY - PY) / PY
End Function
```

The general convention is that comments go at the beginning of the module. So click above the first line and enter an apostrophe and type ‘This function calculates sales variance and I created it’ – then press the Enter key. You don’t even need an end parentheses. As soon as you press the Enter key, you should see that the comment is in green. If it is not in green then Excel does not recognize it as a comment and will try to incorporate the lines into the procedure when it compiles it.

Let’s talk about how to save these functions that we have created.

SAVING CUSTOM FUNCTIONS
If you open a new workbook and try to use the custom function we just created, you will get an error message of #Name? unless the workbook where we created the custom function is also open. In this case, every time you want to use your custom function you have to remember to open the workbook containing the custom function. Obviously this is not ideal. The bigger problem with this, though, is that even after you retrieve the original file, you still need to refer to it when using the formula. For example, if we had named this workbook Exercise1.xlsx, then to use the function we would have to refer to it as =exercise1.xlsx!Svar() instead of =Svar and we would have had to make sure that Exercise1.xlsx was open as well. If this doesn’t bother you then you are set. This is also a work around if you are uncomfortable with macro-enabled files or your company’s software security does not allow macro-enabled files. There are some other options though.

Some people opt to save it to their personal macro workbook as that automatically opens whenever you open Excel. Depending on your version you may not have a personal macro worksheet. If you want to check, go to View>Unhide. If Unhide is dimmed, then you do not have a personal macro workbook. From the looks of it, Excel stopped creating a personal macro workbook back in Excel 2007. If you do not have a personal macro workbook, then you actually have to create a macro and then from the “Store Macro in” option select Personal Macro Workbook. If you are interested in creating your own personal macro workbook to store your custom formulas please refer to the Appendix. However, please remember that you still need to incorporate the name of the personal macro worksheet into the formula, so the only thing this gets you is you don’t have to manually open the file when you are going to use the custom function.
Creating Custom Functions in Excel

The most efficient way to use the custom functions is to create all your custom functions in a separate workbook and then save that workbook as an add-in. Now, just be aware that the ADD-IN adds these custom formulas to every workbook and Excel treats the file as a macro enabled file, even though we have not created a macro. If your corporate software security is locked down this might be a deal breaker and you may want to stick with the first method discussed. (Just wanted to mention the possibility of this being a problem.)

Also, please be careful about using custom functions in workbooks if you tend to share files with others. The custom functions will not work in their file if they do not have the add-in.

First let’s give it a little bit of a description because we love documentation.

1. Click **ALT+F11** to toggle back to the VBA Editor.
2. Click **Tools>VBAPr** - **ject - Project Properties** dialog box or **Tools>VBAPr** - **jectProperties** depending upon your version.
3. Click on the **General** tab enter a project name and description. I entitled mine Myfunctions. If you are going to put all your custom functions in one workbook, which is what I recommend, then you should give it a name that means something to you such as MyFunctions or CustomFunctions.

The VBA Editor does not like spaces or special characters.

4. If you wish to pass word protect you can click on the **Protection** tab and check the **Lock project for viewing** option. However, if you forget the password, you are out of luck, so be careful. The password protects someone from viewing and changing your code.
5. Click OK.

Now, let’s save the file as an Excel Add-in.

6. Toggle back to the Excel worksheet using Alt+F11.
7. Click File>Save As.
8. I named the file as Myfunctions.
9. Click Save as Type.
10. Select Add-in (.xlam file extension) and accept the default directory.
11. Click Save.

Now, let’s add the Add-in in.

1. Click File (or the Office Button for Excel 2007).
2. Click Options.
3. Click Add-ins.
   If you look at the list of Add-Ins you should see yours. In my case you can see Myfunctions.
4. Click the **Go…** button.
When you click the **Go…** button the Add-Ins dialog box displays. This dialog box contains all the Add-Ins currently available. To have Excel access them you need to place a check mark in front of the Add-in you want to use.
5. Put a **checkmark** in front of **Myfunctions**.
6. Click **OK**.

Now, your custom functions will be available every time you open and run Excel. If you want to add additional functions or make a change just press **Alt+F11** or click the **Visual Basic Editor** icon on the **Developer** tab to return to the **Visual Basic Editor**. The beauty of this is you only need to create the Add-In once. As you add additional modules, they will automatically be incorporated.
Double-click on that module in the Project Explorer window and your function code will display. For example, I double-clicked on Module 2 and the following displayed.

When I double-clicked on Module 1, the first function that we created displayed.

If you wanted to add another function all you have to do is put your cursor after the End Function that is under the last function in the Code window and start typing. You could start creating a new function code in either Module 1 or 2. However, my recommendation would be to insert a new module. Again, this just makes it easier to view and edit. Any additional functions that you create will automatically display in the User Defined category in the Insert Function dialog box.

Before getting into a more complex example, let’s try something a bit simpler.

**LET’S TRY-IT- ON YOUR OWN REVIEW**

Before moving onto something more complex I want to give you a chance to practice what we covered so far. Let’s say you want to annualize the monthly return on your investments. If you know the monthly rate, then all you need to do is use the following formula: \((1+\text{Rate})^{12}\)-1. So, if your monthly rate is 2% for all months, then your formulas would be \((1+2\%)^{1}\), which is 26.28%. Nice rate of return if you can get it!

Write it out and then try it. If you are looking for the ^ symbol press Shift + 6.

If it works you are set. If it doesn’t work, compare it to my solution below but remember there is not necessarily one way to do write this.

**Solution**

Here is my solution. Again- if you answer is not identical that does not mean that yours is incorrect.

Function ARR(Value)
Creating Custom Functions in Excel

ARR = (Value + 1) ^ 12 – 1

End Function

Now, if you looked at this module, you would probably have no idea what ARR was so how can we write it better? Documentation of course!

‘ARY is Annualized Rate of Return
‘Assumes all months are the same monthly rate. Formula is ARR=(1+Rate)^12-1
“Created by Patricia McCarthy – May 2015

Switch to Excel and enter 2% in cell B6 and then try your formula. The answer is .268 or 26.8%

CREATING A CUSTOM FUNCTION – IF STATEMENT

Let’s create our last custom function- an IF statement. We are going to do this the long way and then discuss an alternative method.

1. Switch over to Excel by pressing the Excel icon or using the shortcut keys Alt+F11.
2. On a new worksheet, type the following:
## Creating Custom Functions in Excel

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Customer</td>
<td>Product ID</td>
<td>Quantity</td>
<td>Price</td>
</tr>
<tr>
<td>2</td>
<td>Smokin’ Susie</td>
<td>CB-Tanzania</td>
<td>100</td>
<td>5</td>
</tr>
<tr>
<td>3</td>
<td>Kelsey’s Coffee House</td>
<td>CB-Hawaii</td>
<td>30</td>
<td>10</td>
</tr>
</tbody>
</table>

We want to create a custom IF function that will test the quantity and see if it equals or exceeds 50. If quantity is equal to or greater than 50, we want to give a 10% discount. If we were doing this in Excel itself, the calculation would be `$=IF(C2>=50, (C2*D2)*.1, 0)`.

Surprise, it is a bit more work to create this in a custom formula.

1. Let’s toggle back to the VBA Editor and create the new custom formula.
2. Click **Insert>Module** to create a function in a new module.

> Excel will not accept 10% which is why I wrote it as .1.

3. Enter the information as shown below:

   ```vba
   Function Quantity(Value1)
   Quantity = Value1
   End Function
   
   Function Price(Value2)
   Price = Value2
   End Function
   
   Function Discount(Quantity, Price)
   If Quantity >= 50 Then
   Discount = (Quantity * Price) * 0.1
   Else
   Promotion = 0
   End If
   End Function
   ```

4. Excel should add the last line which is End Function.
5. Press **Enter**.

If you went all out and added some documentation, then it should similar to below:

```vba
' This function tests to see if a discount should be applied.
' If the Quantity is greater than or equal to 50, a 10% discount is applied.
Function Quantity(Value1)
    Quantity = Value1
End Function

Function Price(Value2)
    Price = Value2
End Function

Function Discount(Quantity, Price)
    If Quantity >= 50 Then
        Discount = (Quantity * Price) * 0.1
    Else
        Promotion = 0
    End If
End Function
```

Notice that I used the Tab and the Shift+Tab to indent some of the code so that it is easier to read.

6. Press **Alt+F11** to get back to the worksheet you just created.
Creating Custom Functions in Excel

9. Go to cell E2 and then click the Fx button and select Discount from the User Defined Category.

10. Enter C2 for Quantity.
11. Enter D2 for Price.

12. Press OK.
13. Copy it down to E3 and you can see that a discount was not given as the quantity was not equal to or greater than 50.
COMPARING SYNTAX

Notice a difference in the way we wrote this? Besides the fact that it is in IF syntax? You should notice two things:

1. First, since this was an IF, Excel automatically added an End IF function immediately before the End Function.
2. Second, we defined Quantity and Price as we did in the Svar example; however this time we really did not need to do that. Why not you ask?

Here is the alternative code we could have written. The only difference between this function and the one we just created is that instead of calling it Discount, I called it Promotion as you cannot create two functions with the same name.

This is clearly a lot cleaner as it has fewer lines and is easier to understand and gets us to the same answer.

In this case of the Discount formula, we did not really need to define quantity and price as Excel assumed the first variable was quantity and discount was the second variable. So, why didn’t we let Excel assume that when we created the Svar function since we only used two variables there as well?
Creating Custom Functions in Excel

We did use 2 variables; however, we used one of the variables twice if you recall. If we had wanted to select the PY cell a second time Excel will not allow us to use the same named variable twice. When I tried to use PY twice as shown below, I got a compile error.

```vba
Function Variance (cy,py,py)
    Variance=(cy-py)/py
End Function
```

A work-around to avoid defining the variables would result in coding something like this with PYAgain representing PY Sales Again:

```vba
Function Variance (cy,py,pyagain)
    Variance=(cy-py)/pyagain
End Function
```

We have 3 variables and it will work but someone would have to know the formula as they would need to know that they need to select the PY sales a second time. That would pretty much defeat the purpose of creating the function in the first place. Hopefully, that makes sense.

**EDITING AND DELETING A CUSTOM FUNCTION**

**EDIT**

If you need to edit a custom function that has already been created you just need to go to the VBA Editor, find the module that contains the code and make your changes. In the screenshot below, I
Creating Custom Functions in Excel

double-clicked on *Module1* (in the Project Window) and our first function displays on the right side of the screen so I can easily edit it if I wish.

DELETE

If you right-click on Module1, you are offered a variety of choices including viewing the code, seeing the module’s properties, **printing**, importing and removing the Module itself.

If you do delete a custom function, then any existing files using that function will display a Name error message.

*To delete a custom function that you no longer need, simply right-click on the Module you want to remove, and select Remove Module.* Excel will ask you if you want to export it first. Click No and Excel will remove the module. When you delete the module, the function no longer displays in the User-Defined Category.

ADD-INS

If you wish to deactivate the Excel Add-In

1. Click **File>Options**.
2. Select the **Add-Ins** category.
3. Go the **Manage** box and select **Add-ins**.
Creating Custom Functions in Excel

4. Click Go.
5. Remove the checkbox next to the Add-to be inactivated.
6. Click OK.

Removing the checkmark simply inactivates that Excel Add-in. It does not remove it. To remove the add-in from your computer you need to uninstall or delete it depending on how it was created.

In this case, I clicked the Browse button on the Add-Ins dialog box and the Browse dialog box showed me the Add-Ins that I had created. To remove it, simply right-click on it and select Delete. If you installed the Excel Add-in from a network file server or a filed folder you may have to do some additional steps and go to the Control Panel and look in the ADD/Remove Program window and see if the Add-In is listed there. If so, remove it.

CONCLUSION
Well, we are finished. We created a couple of different custom functions and we created an Add-In file so that the custom functions could be used with any workbook. As you go forward and create new custom functions, they will automatically be added to the Add-IN and be available. We also saw how to delete and remove ADD-Ins.

Ozgrid.com offers a variety of useful custom functions that the Hawley team have created. All of these custom functions are VBA Based. If you use VBA or have an interest in gaining more knowledge in that area than I recommend that you check it out. The website is:
http://www.ozgrid.com/VBA/Functions.htm

Another good reference is Ron de Bruin if you are interested in creating a personal macro workbook.
http://www.rondebruin.nl/win/personal.htm

In the Appendix, I have an explanation from Microsoft on how to create a personal macro workbook in case you are interested in that. If you currently use macros then that may be the way to go; however, I think that using the ADD-IN feature is more efficient – but that is just my opinion.
REVIEW QUESTIONS
Review Questions are designed to assist in the learning process and solutions can be found at the end of the EBook. They are a requirement of NASBA. Answers immediately follow.

1. To add in the Developer tab, you need to go to ________________.
   A. File >Options>Customize Ribbon
   B. File>Customize Ribbon
   C. File>Advanced>Custom Ribbon
   D. File>Add-Ins

2. VBA Stands for
   A. Visual Basic for Applications
   B. Visual Basic Applications
   C. Virtual Basic Applications
   D. Virtual Basic Apparitions

3. When you open the VB Editor, you are in the ___Window which opens on the top left side of the screen.
   A. Module
   B. Project
   C. Macro
   D. Book

4. To create a module in which to type your code, you need to click the ___menu.
   A. View
   B. Run
   C. Insert
   D. Add-Ins

5. When creating a custom function, the last line of code should be the word___.
   A. Function
   B. End Function
   C. Sub Procedure
   D. End Procedure

6. To have a custom function work in every workbook you need to include the file as a/an____.
   A. Macro
   B. UDF
   C. ADD-IN
   D. Module

7. To deactivate an Excel Add-in
   A. Remove the checkmark beside it
   B. Right-click on the Add-in and press Delete
   C. Right-click on the Add-in and select Inactive
   D. Right-click on the Add-in and select Remove
8. Which statement is TRUE?
   A. A custom function is considered safer than a macro
   B. You cannot create custom functions in Excel 2007
   C. Custom functions allow you to change the structure of the workbook.
   D. Custom functions do not need to be documented.
SOLUTIONS TO REVIEW QUESTIONS

1. To add in the Developer tab, you need to go to _________________.
   A. File > Options > Customize Ribbon
   B. File > Customize Ribbon
   C. File > Advanced > Customize Ribbon
   D. File > Add-Ins

   **A is correct.** You need to select File and then Options to locate the Customize Ribbon which is where you can add the Developer tab to the Ribbon.
   **B is incorrect.** You are missing a step between File and Customize Ribbon.
   **C is incorrect.** To find the Advanced or Custom Ribbon you need to first select Options. Advanced takes you to editing options.
   **D is incorrect.** You are missing a step between File and Add-Ins. Add-ins does not allow you to add the Developer tab. It shows you programs that can be added in and used in Excel.

2. VBA Stands for
   A. Visual Basic for Applications
   B. Visual Basic Applications
   C. Virtual Basic Applications
   D. Virtual Basic Apparitions

   **A is correct.** VBA stands for Visual Basic for Applications. It is a programming language used in Excel.
   **B is incorrect.** It is missing the word “for”
   **C is incorrect.** There is no such thing as Virtual Basic Applications- I made up the term.
   **D is incorrect.** Apparitions are typically ghosts and have nothing to do with programming.

3. When you open the VB Editor, you are in the ___Window which opens on the top left side of the screen.
   A. Module
   B. Project
   C. Macro
   D. Book

   **A is incorrect.** The module is where you actually write the code to create your custom function.
   **B is correct.** The Project Window opens up on the left side of the VB Editor and provides an index of all the VB files, linked to all open Excel workbooks including modules.
   **C is incorrect.** There is no such thing as a Macro window.
   **D is incorrect.** A work refers to an Excel workbook and there is no such thing as a Book Window.

4. To create a module in which to type your code, you need to click the ___ menu.
   A. View
   B. Run
   C. Insert
   D. Add-Ins
Creating Custom Functions in Excel

A is incorrect. You want to create a module – not view or look at it.
B is incorrect. If you run a programming module you are executing it.
C is correct. To create a module you click Insert in the Visual Basic Editor menu.
D is incorrect. Add-Ins will show you programs that can be run inside of Excel. After you create a module you can turn it into an Add-In but that is not how you create a module.

5. When creating a custom function, the last line of code should be the word ____.
   A. Function
   B. End Function
   C. Sub Procedure
   D. End Procedure

A is incorrect. Function should be the first line of code when creating a custom function – not the last line.
B is correct. End Function is the last line of code that should display when creating a custom function. The Visual Basic Editor will automatically add this line of code if it recognizes your code as a custom function.
C is incorrect. Sub Procedure is a keyword to identify a programming procedure and does not belong in the syntax of a custom function.
D is incorrect. End Procedure would be the last line of a programming procedure and does not belong in the syntax of a custom function.

6. To have a custom function work in every workbook you need to include the file as a/an____.
   A. Macro
   B. UDF
   C. ADD-IN
   D. Module

A is incorrect. A macro is an independent self–executable file. You can have a macro run a custom function however it would not work in every workbook just by creating the macro. You would have to associate the macro and the custom function and then save it in a personal macro workbook.
B is incorrect. UDF is simply another older term for custom functions. It stands for User Defined Function.
C is correct. If you create a custom function and then make it into an ADD-IN file or program then it will work in every Excel workbook that you open. It will not work if you share your workbook with someone else unless they also have the ADD-IN.
D is incorrect. You write the custom function in a module however if you only do that then it will only work in that workbook or others that you open while that workbook containing the module is open as well. In the latter case, you need to include the workbook name when calling the custom function.

7. To deactivate an Excel Add-in
   A. Remove the checkmark beside it
   B. Right-click on the Add-in and press Delete
   C. Right-click on the Add-in and select Inactive
   D. Right-click on the Add-in and select Remove
Creating Custom Functions in Excel

**A is correct.** Simply remove the checkmark beside the name of the Add-in to be deactivated. This does not delete the add-in – it just makes it inactive.

**B is incorrect.** If you right-click on the Add-in nothing will happen.

**C is incorrect.** If you right-click on the Add-in nothing will display.

**D is incorrect.** If you right-click on the Add-in nothing will display.

8. Which statement is TRUE?
   A. A custom function is considered safer than a macro
   B. You cannot create custom functions in Excel 2007
   C. Custom functions allow you to change the structure of the workbook.
   D. Custom functions do not need to be documented.

**A is correct.** Custom functions are considered safer than macros because they cannot change a cell’s value or the structure of the workbook.

**B is incorrect.** You can create custom functions in Excel 2007. In fact, you could create them in earlier versions of Excel as well.

**C is incorrect.** A macro can change the structure of a workbook however a custom function cannot.

**D is incorrect.** Custom function do not absolutely require documentation; however they do in fact need to be documented. Be kind to other users!
APPENDIX A

Create and use Personal Workbook - Microsoft
https://support.office.com/en-nz/article/Copy-your-macros-to-a-Personal-Macro-Workbook-aa439b90-f836-4381-97f0-6e4c3f5ee566
APPENDIX B

CHANGE MACRO SETTINGS

Change macro security settings in Excel

In Microsoft Excel, you can change the macro security settings to control which macros run and under what circumstances when you open a workbook. For example, you might allow macros to run based on whether they are digitally signed by a trusted developer (a person who writes programming code).

The following list summarizes the various macro security settings. Under all settings, if antivirus software that works with Microsoft Office is installed and the workbook contains macros, the workbook is scanned for known viruses before it is opened.

- **Disable all macros without notification**  Click this option if you don't trust macros. All macros in documents and security alerts about macros are disabled. If there are documents that contain unsigned macros that you do trust, you can put those documents into a trusted location. Documents in trusted locations are allowed to run without being checked by the Trust Center security system.

- **Disable all macros with notification**  This is the default setting. Click this option if you want macros to be disabled, but you want to get security alerts if there are macros present. This way, you can choose when to enable those macros on a case by case basis.

- **Disable all macros except digitally signed macros**  This setting is the same as the Disable all macros with notification option, except that if the macro is digitally signed by a trusted publisher, the macro can run if you have already trusted the publisher. If you have not trusted the publisher, you are notified. That way, you can choose to enable those signed macros or trust the publisher. All unsigned macros are disabled without notification.

- **Enable all macros (not recommended, potentially dangerous code can run)**  Click this option to allow all macros to run. Using this setting makes your computer vulnerable to potentially malicious code and is not recommended.

- **Trust access to the VBA project object model**  This setting is for developers and is used to deliberately lock out or allow programmatic access to the VBA object model from any Automation client. In other words, it provides a security option for code that is written to automate an Office program and programatically manipulate the Microsoft Visual Basic for Applications (VBA) environment and object model. This is a per user and per application setting, and denies access by default. This security option makes it more difficult for unauthorized programs to build "self-replicating" code that can harm end-user systems. For any Automation client to be able to access the VBA object model programatically, the user running the code must explicitly grant access. To turn on access, select the check box.

1 Excel Help- Enable and Disable Macros
CHANGE MACRO SECURITY SETTINGS

You can change macro security settings in the Trust Center, unless a system administrator in your organization has changed the default settings to prevent you from changing the settings.

On the Developer tab, in the Code group, click Macro Security.

In the Macro Settings category, under Macro Settings, click the option that you want. For detailed information about these settings, see the section Macro security settings and their effects, earlier in this article.

**Note** Any changes that you make in the Macro Settings category in Excel apply only to Excel and do not affect any other Microsoft Office program.

**Tip** You can also access the Trust Center in the Options dialog box.

- Click the File tab, click Options, and then click the Trust Center category.
- Click Trust Center Settings, and then click the Macro Settings category.
USING DIGITAL SIGNATURES

A digital signature is an electronic, encrypted, stamp of authentication on digital information such as email messages, macros, or electronic documents. A signature confirms that the information originated from the signer and has not been altered.

The following is an example of a signature line.

![Signature Example]


TROUBLE SHOOTING

I can’t change my macro security settings

Some users may not be able to change Trust Center settings due to group security policies in their organizations. In such cases, you need to contact the IT administrator for your organization.
GLOSSARY

Custom Function
A function that has been written or defined by the user

Macro
A self-executing program that is comprised of a series of steps.

Module
An object or file that holds VBA Code and/or user-defined code

Personal Workbook
A personal workbook is hidden by default and can contain macros that have been written by or for the user. The personal macro if it exists opens whenever the Excel program is started.

Project Window
It opens up on the left of the VBA Editor and provides you with an index of all Visual Basic files, linked to all open Excel workbooks. This consists of a file linked to each worksheet of your Excel workbook and modules, which contain user-defined code and are not linked to any specific worksheet.

UDF
User Defined Function – See Custom Function

VBA
Visual Basic for Applications is the name of the Excel programming language and is generally used to automate tasks within the program.