How To Maximize Excel by Using Custom Function

Whether you play a technical role or are a financial analyst, Excel is likely a major asset in your toolbox. Whether it is the SUM function, the VLOOKUP function, or one of the many others, we have all used Excel functions for a plethora of reasons.

There is a lot of potential hidden in Excel that you may not be aware of. Excel offers the ability to create your own user defined functions, and it's not hard to create them. With a little ingenuity and strategic thinking, custom Excel functions can be a huge asset.

Below are two examples. Neither is difficult, but they will provide you with a taste of what you can do with custom functions. The first example calculates a better/worse value based on three inputs (prior period, current period, and expense vs. revenue). The second concatenates columns together with a user specified delimiter and the option to use quotes around the values.

Background on Custom Functions

Custom functions are Visual Basic for Applications (VBA) code snippets that are stored in modules in a workbook. This is the same place macros are stored, so it may be familiar. To open the VBA window, use ALT F11. Once the window opens, right-click the workbook you want to add the function to in the VBAProject window and select Insert->Module. A new window will open named Module1. Custom functions have to be in a module to be accessed in a workbook.

Each function has a function name, input arguments that pass data to the function, and return a value.

A very simple example shows these pieces. “Test” is the function name. “Input” is one argument passed to the function. The function returns a numeric value, which is the input value multiplied by ten.

```
Function Test(input as double) as Double
    Test=input * 10
End Function
```

To use this function, return to your worksheet and enter “=Test(5)” in a cell. This function can also be found in the Insert Function option by selecting User Defined in the Select A Category dropdown box. The input parameter doesn’t have to be a value. A cell reference can be used, just like any other Excel function. The result should return 50.

Example: Better(Worse) Calculation

For you finance folks, you will almost always have a better/worse calculation in a spreadsheet that compares two periods. For revenue, the current period is subtracted from the prior period. For expense, it is the inverse.

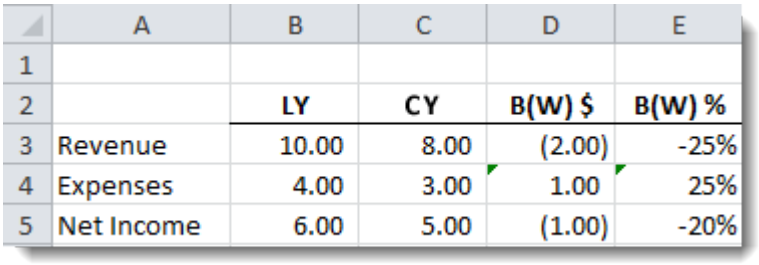
To accomplish this, we will have a function with 4 parameters.

1. Prior Period
2. Current Period
3. Whether the numbers being evaluated should be calculated as an expense or revenue
4. Whether the result returned is in the form of a dollar value or percentage change

```
Function BetterWorse(Prior_Period As Double, Current_Period As Double, Expense As Boolean, Return_Dollar As Boolean) As Double
    If Expense = True Then 'Calculate as an expense
        If Return_Dollar = True Then  'Return a dollar value
            BetterWorse = Prior_Period - Current_Period
        Else  'Return a percentage
```

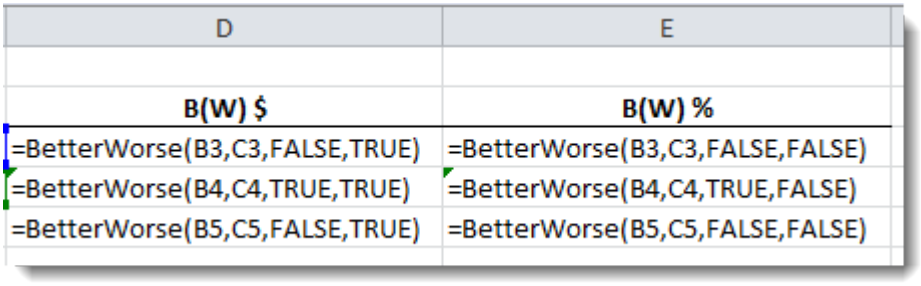
```
            BetterWorse = (Prior_Period - Current_Period) /
Prior_Period
      End If
    Else  'Calculate as a revenue
        If Return_Dollar = True Then  'Return a percentage
            BetterWorse = Current_Period - Prior_Period
        Else  'Return a percentage
            BetterWorse = (Current_Period - Prior_Period) /
Current_Period
      End If
    End If
End Function
```

Below is an example of this function being used. The result of the custom function resides in column D and E. Revenue is lower in the current year, resulting in a negative variance. Expenses are also lower, but result in a positive variance.



	A	B	C	D	E
1					
2		LY	CY	B(W) $	B(W) %
3	Revenue	10.00	8.00	(2.00)	-25%
4	Expenses	4.00	3.00	1.00	25%
5	Net Income	6.00	5.00	(1.00)	-20%

The formulas that exist in columns D and E are as follows.



D	E
B(W) $	B(W) %
=BetterWorse(B3,C3,FALSE,TRUE)	=BetterWorse(B3,C3,FALSE,FALSE)
=BetterWorse(B4,C4,TRUE,TRUE)	=BetterWorse(B4,C4,TRUE,FALSE)
=BetterWorse(B5,C5,FALSE,TRUE)	=BetterWorse(B5,C5,FALSE,FALSE)

Example: Concatenation

The need to create a delimited file from Excel is very common. The problem with doing this is that the entire worksheet is extracted. If the worksheet had data in rows or columns that are now blank, Excel still exports those blank cells. One way to overcome this is to create a function that

concatenates a range into one cell. Then, the concatenated values can be copied and pasted to a text file. Many times this is very handy. This can obviously be done with a cell formula, but gets time consuming to create when many cells are required. It is further complicated when quotes around the fields are necessary.

```
Function ConcatForExport(InRange As Range, Delimiter As String, UseQuotes As Boolean) As String
    Dim TheCount As Integer
    TheCount = 0
    For Each cell In InRange
        If TheCount = 0 Then
           If UseQuotes = True Then
              strString = Chr(34) & cell.Value & Chr(34)
           Else
              strString = cell.Value
           End If
        Else
           If UseQuotes = True Then
                strString = strString & Delimiter & Chr(34) & cell.Value & Chr(34)
           Else
               strString = strString & Delimiter & cell.Value
           End If
        End If
        TheCount = TheCount   1
    Next cell
    ConcatForExport = strString
End Function
```

To expand on the variance example above, an additional column has been added to show the use of this function. Each row passes different parameters. Columns B through E are concatenated together into one cell. The delimiter is altered in row 5, and no quotes are around the value in row 4.

	A	B	C	D	E	F	G
1							
2		**LY**	**CY**	**B(W) $**	**B(W) %**		**Export Ready**
3	Revenue	10.00	8.00	(2.00)	-25%		"10"|"8"|"-2"|"-0.25"
4	Expenses	4.00	3.00	1.00	25%		4|3|1|0.25
5	Net Income	6.00	5.00	(1.00)	-20%		"6","5","-1","-0.2"

The corresponding formulas are below.

G
Export Ready
=concatforexport(B3:E3,"|",TRUE)
=concatforexport(B4:E4,"|",FALSE)
=concatforexport(B5:E5,",",TRUE)

There are a wealth of opportunities that open up using custom functions. Adding functionality and automating tasks like the examples above are just the start of what can be done.