Introduction to VBA for Excel-Tutorial 8

In this tutorial, we will revisit VBA functions. Recall the difference between the Sub and Function is that: the Sub is set of commands that the user or another procedure can execute. On contrast, the function returns a single value (or array) based on input arguments passed in by the user and is executed from Excel interface. For example, if you would like to use the built-in square-root function in Excel to calculate $\sqrt{4}$, then you go a cell and type:

=sqrt(4)

Then press Enter, and in the same cell you selected the results will show. In this example, ***sqrt*** is the function name and ***4*** is the argument of the. Recall that the function arguments are not limited to certain number. For example, **Round** function:

=Round(number,num_digits)

Again, ***round*** is the function name and ***number*** and ***num_digits*** are required arguments. Remember that the difference between a function and Sub is the method of passing inputs and receiving outputs. The logic remains the same regardless of using a Sub or a function.

Example:

Let's say, you would like to create your own custom function to calculate the square-root, where:

$$\sqrt{A} = \frac{1}{2}(x_i + \frac{A}{x_i}) \quad i = 1, 2, 3.....$$

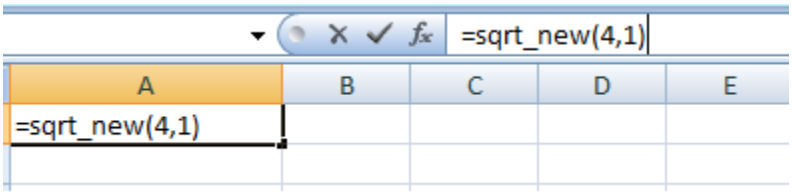
where, x_o is the initial guess and A is the number of which is the square root to be calculated. For this example, take the convergence criteria to be 1. In the effort of planning this code, you should consider three aspects; (1) Need a loop to iterate till convergence achieved, (2) Need to calculate and update the convergence after each step and (3) Need to keep track of the results from the previous iteration. The code would be:

```
Option Explicit
Function sqrt_new(A As Double, xo As Double)
Dim delta As Double
Dim old As Double
Dim ans As Double
delta = 100
ans = xo
Do While delta > 1
ans = 0.5 * (ans + A / ans)
delta = Abs(ans - old)
old = ans
Loop
sqrt_new = ans
End Function
```

Notes:

1- The inputs or the arguments of the function are included in the bracket and each variable is declared as its appropriate datatype. The declaration is optional.
2- Implemented Do-While-Loop; because we don't know the number of iterations required to achieve convergence. Please note that at the end of each loop we cast the new results (i.e. ans) into a transfer variable (i.e. old) to allow the calculation on convergence on the next iteration.
3- The most important difference between Function and Sub is the output. Note that the last line is setting the function name (i.e. sqrt_new) to the calculated results. Without this line, the answer would be always equal to zero,

Once the function is written and debugged, you can use this function through the Excel interface such as:



where, A= 4 and the initial guess x_o = 1. Once you type in this formula and press Enter, Excel will output 2.05 as the results. To achieve better accuracy, simply set the convergence to a smaller number.

Not only you can use the Function from Excel, but you can call the Function in sub procedure.

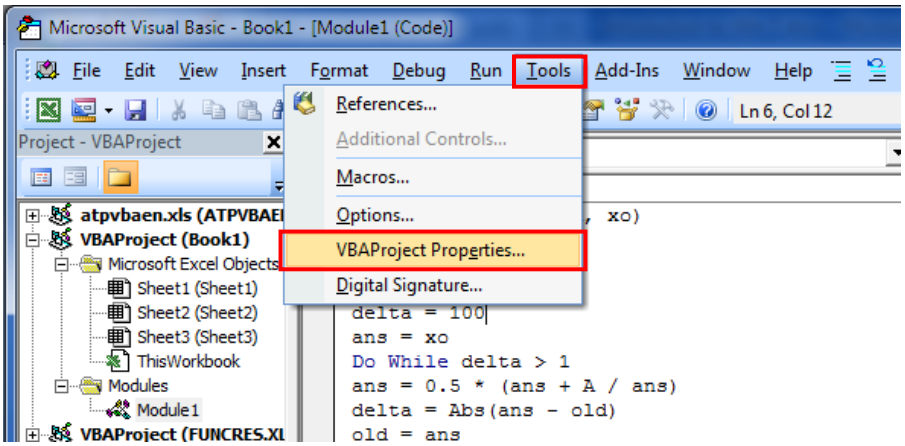
```
Sub testing_func()
Dim A As Double, xo As Double, root As Double
A = InputBox("Enter Positive Number")
xo = InputBox("Enter Guess")

root = sqrt_new(A, xo)

MsgBox root
End Sub
```

In this case, the Sub is placed in the same Module as the Function. However, the variables are defined using the same variable names as those used in the function, this not required. You can name the variables any name as far as you place the right variable in the right location (i.e. first argument vs second argument).

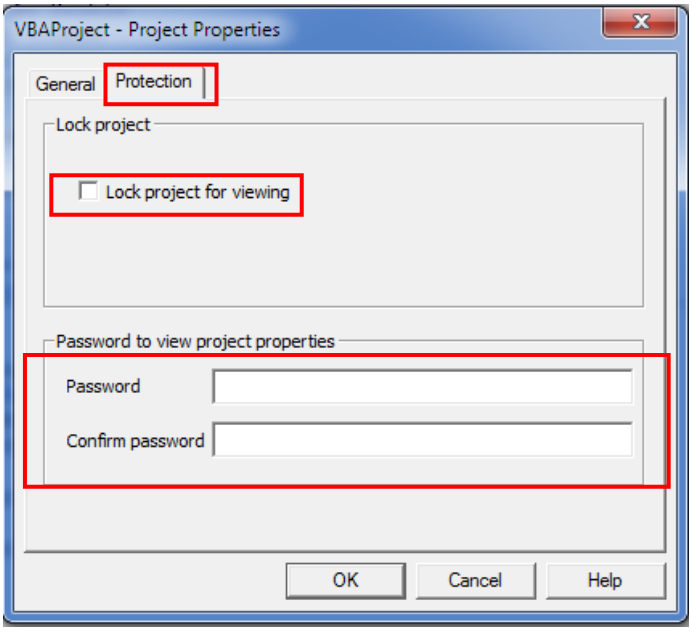
Deploying Functions (Creating Add-in)

After you checked your function, you may want to distribute this function to everyone in the office to display your superior skills. Follow the steps to fame:

1- Open the worksheet where the future add-in function is in.
2- Activate VBE and select the module where the function is in.
3- Click Debug/Compile. This is a final check to make sure the code would compile correctly and save you the embarrassment, especially with your boss.
4- If you would like to set a password for your project, so no one can tamper with your code, Click on Tools/VBA Project Properties (note: that if your project has been previously named the label would be *Project_name* Properties):



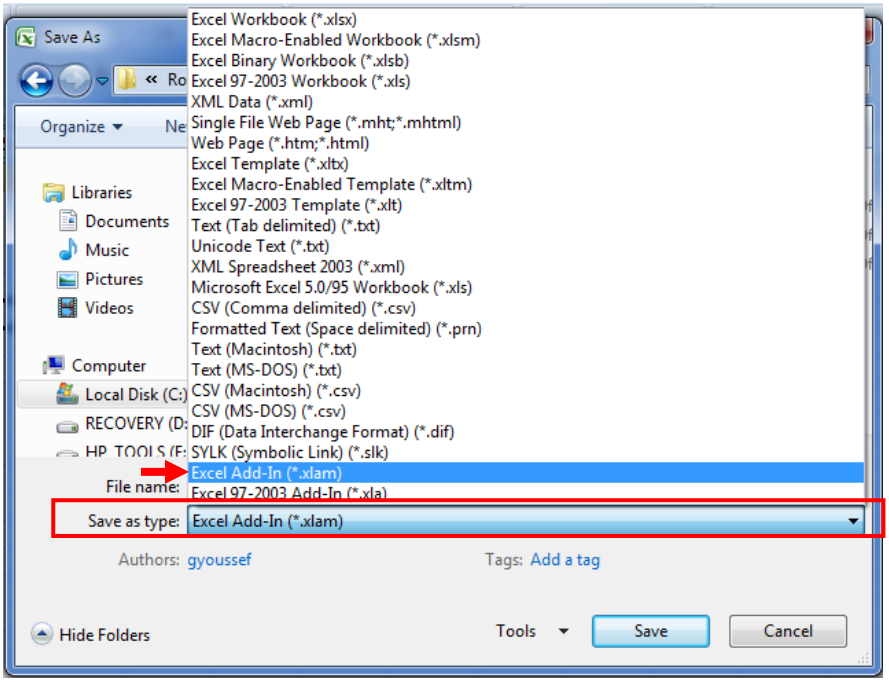
VBE will display Project Properties dialog box, Select the Protection tab, check the Lock project for viewing box, and enter credentials in the password textbox.



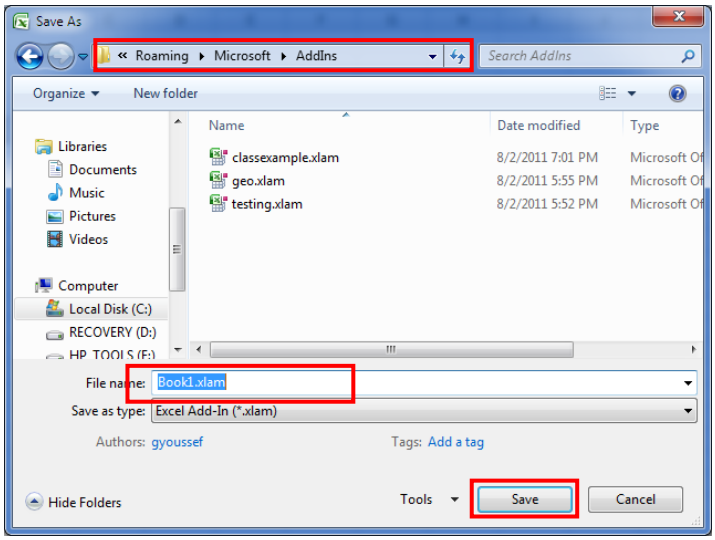
Word of cautions: this password is not easy to recover, so don't make it to complicated that you can forget it easily. Remember, you won't work on the same function sometimes for years.

5- In Excel, Click File and then click Save As

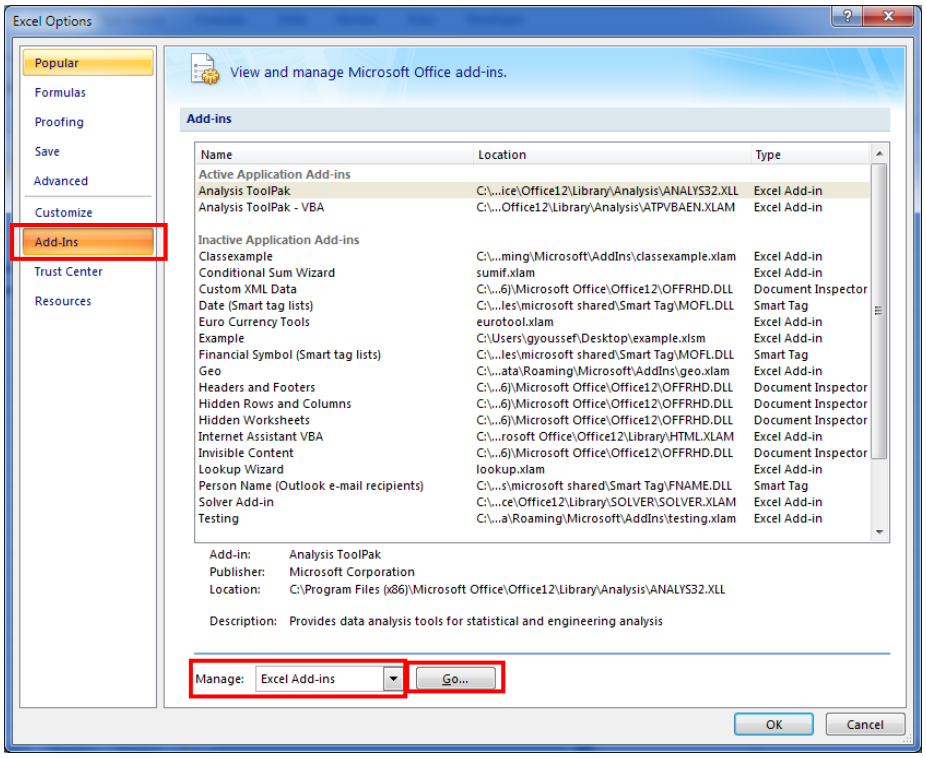
6- Select file type .xlam (Excel Add-in) from Save as Type dropdown menu, then Insert the File Name then click Save



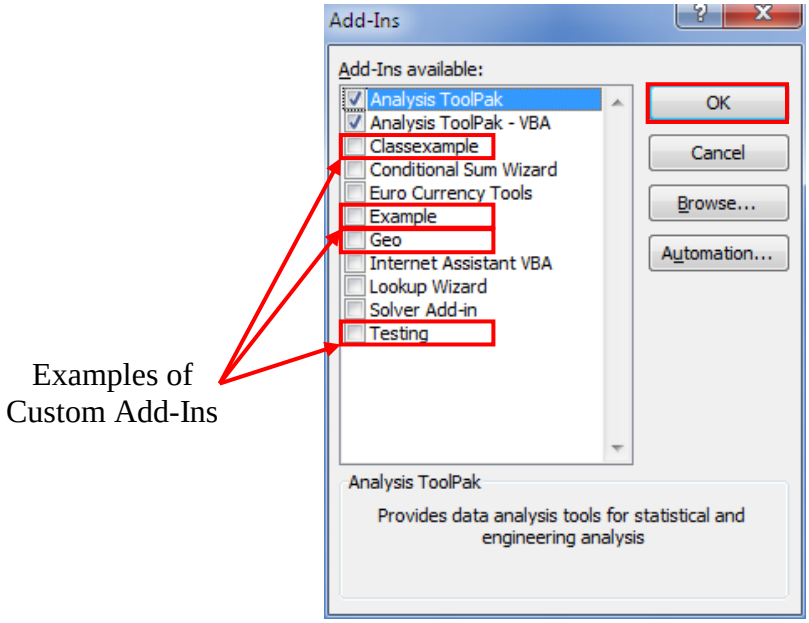
Note: Excel will automatically select the correct default directory for the Add-Ins which is (*C:\Documents and Settings\<username>\Application Data\Microsoft\AddIns*)



7- Close all open workbooks and Open a new workbook
8- Click File/ Options/Add-Ins/Excel Add-Ins from Manage dropdown menu then Click Go



9- Excel will prompt you with Add-Ins dialog box, Select the add-in you want to add and click OK.



Enjoy!