Excel VBA Programming: Macros

(Dr. Tom Co, 9/17/2008)

Introductory items:

1. A **macro** (or subroutine) is a group of statements (or code) that performs a set of tasks without yielding a value back.
2. One could usually use the Record option. Often one may need to enhance the macros directly by programming statements.

Guidelines for when to create a macro:

1. When the tasks are used often enough in which a template may be not be sufficient.
2. When the benefits (e.g. minimizing human errors in repeated tasks) outweigh the programming effort.

Standard Format of Function

Sub *function_name* (*argument_1* ,*argument_2*, ...)

⋮

statements and calculations

⋮

End Sub

Example 1. Recording a macro to access text data

Preliminary step: download and extract the file: antoine_data1.txt from the URL:

http://www.chem.mtu.edu/~tbco/cm3450/antoine_data1.zip

Step 1. Open a new Excel spreadsheet and start macro recording by selecting **[View]→[Macro]→[Record Macro...]** and select **[CTRL-a]** as the hotkey.

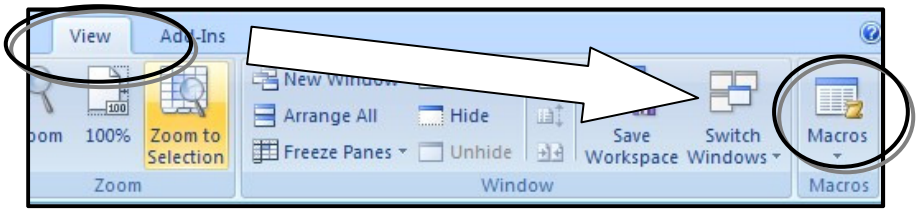

Figure 1. Macro Recording.

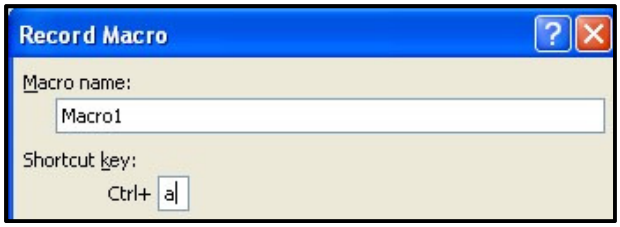

Figure 2. Macro Edit window.

Step 2. Select from menu: **[Data]→[Get External Data]→[From Text]**, and choose the file **antoine_data1.txt** from the location chosen in the preliminary step.

Step 3. After you have selected the cell to place the data, stop the recording by clicking on the box in the lower left corner as shown in Figure 3.

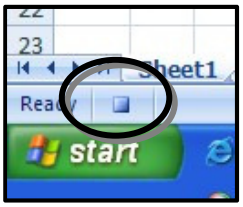

Figure 3. Stop-Macro-recording button.

Step 4. Press **[Alt-F11]** and you should see that a macro called **Macro1** has been programmed.

Step 5. Delete the columns containing the data and then test the macro by using the hotkey **[CTRL-a]**.

Question: When you run the macro, does it still asks you to input the file location? Why?

Example 2. Enhancing the data loading macro with additional statements.

Step 1. Modify Macro1 by changing the name to get_TP_data. Then add the lines shown in Figure 5 while modifying the destination to **A1** and name to "**RawData**".

```
Sub get_TP_data()
'
    Filename = Application.GetOpenFilename( _
        FileFilter:= _
        "Text Files (*.txt), *.txt, All Files (*.*), *.*", _
        Title:="T and P Data File")
    FileConnection = "TEXT;" & Filename
    With ActiveSheet.QueryTables.Add( _
            Connection:=FileConnection, _
            Destination:=Range("$A$1"))
        .Name = "RawData"
        .FieldNames = True

            ...

        .Refresh BackgroundQuery:=False
    End With
End Sub
```

Figure 5. Modified macro.

Step 2. In the spreadsheet, delete all columns once more and test the new macro.

Remarks:

1. VBA is object-oriented programming where each object can have properties or procedures (methods) connected with each object. These are separated by periods ('.').
2. Variables can also be set to cell-ranges. Note the entry in Figure 5 of **Range("A1")**.

Example 3. Automating the calculation of Antoine Coefficients.

Step 1. While inside the module where the other macro resides include the following macro:

```
Sub get_Antoine_Coefs()
    Cells(2, 1).Select
    Range(Selection, Selection.End(xlDown)).Select
    Set T = Selection
    Cells(2, 2).Select
    Range(Selection, Selection.End(xlDown)).Select
    Set P = Selection
    T.Name = "T"
    P.Name = "P"
    n = P.Cells.Count
    Cells(1, 3) = "T log(P)"
    Range(Cells(2, 3), Cells(1 + n, 3)).FormulaArray = "=T*log(P)"
    Cells(1, 4) = "log(P)"
    Range(Cells(2, 4), Cells(1 + n, 4)).FormulaArray = "=log(P)"
    Cells(1, 5) = "T"
    Range(Cells(2, 5), Cells(1 + n, 5)).FormulaArray = "=T"
    Set y = Range(Cells(2, 3), Cells(1 + n, 3))
    y.Name = "y"
    Set x = Range(Cells(2, 4), Cells(1 + n, 5))
    x.Name = "x"
    Cells(1, 6) = "a2"
    Cells(1, 7) = "a1"
    Cells(1, 8) = "a0"
    Range(Cells(2, 6), Cells(2, 8)).FormulaArray = _
            "=linest(y,x,true,false)"
    Cells(3, 10) = "A"
    Cells(4, 10) = "B"
    Cells(5, 10) = "C"
    Cells(3, 11) = Cells(2, 6).Value
    Cells(4, 11) = -Cells(2, 6).Value * _
            Cells(2, 7).Value - Cells(2, 8).Value
    Cells(5, 11) = -Cells(2, 7).Value
End Sub
```

Step 2. In the spreadsheet, select **[View]→[Macro]→[View Macros]** menu item. Select **get_Antoine_Coefs** macro then click **[Options...]** button and assign the macro with a hotkey, e.g. **[CTRL-b]**.

Step 3. Run the macro.

Remarks:

1. The first three lines selects the range of cells and assigns it to variable T.

```
Cells(2, 1).Select
Range(Selection, Selection.End(xlDown)).Select
Set T = Selection
```

   a) **Cells(2,1)** is the same as cell **A2**.
   b) The second line with **xlDown** is the same as the shortcut **[CTRL-SHIFT-Down]** to select a contiguous range of cells.
2. The line with **T.Name = "T"** just gives the cell range an array name.
3. The keyword **.FormulaArray** is needed to set array computation formulas (recall that manually we had to use **[CTRL-SHIFT-ENTER]**).
4. In the last few lines, we used the format such as : **Cells(2,8).Value** , to access the number contained in that cell, i.e. in **H2**.