RMS 4001 Tutorial 3

INTRODUCTION TO VBA PROGRAMMING

1. Starting and Quitting VBA Programming:
   - To start Visual Basic Application (VBA) in Excel from PC machine, choose <Program> from the <start menu> and click Microsoft Excel.
   - When the program is running, it can be interrupted by <Ctrl> and <Break>

2. Recording and Running Macros in Excel:
   - Excel has the capacity of recording very simple macros; it takes note of our keystrokes and converts them into a VBA subroutine (with no parameters).
   - We want to repeat the same procedures in several different Excel files with the same data format. For example, we want to compute the Return, Expected Return and Variance of Return for the stock Cheung Kong. So we record the procedures that performed in the file 0001.xls by the following steps:
     
     Choose <Tools> → <Macro> → <Record New Macro> → ……
     Perform the standard procedures you wanted ……→ <■>

     We can view the code which describes the procedures we did:
     
     Choose <Tools> → <Macro> → <▶ > → Click <Edit>

     To run the same procedures for other stocks (0002.xls, 0066.xls, 1038.xls), we simply do the followings:
     
     Keep the excel file with recorded Macros opened → Open the files you want to perform the same procedures e.g., 0002.xls → Choose <Tools> → <Macro> → <▶ > → Click <Run>

     To see more different functions or commands in the VBA, you can choose <View> → <Object Browser>. Further, if you want to see the help for any function, just click <?> on the right top hand corner. In the tutorial, we will demonstrate some simple use of functions like “Range”, “Activate Cell”, “.Select”, “Selection.ClearContents” and “Cells”.

1
3. **Self Defined Function in Excel:**

   **Example 1:**

   Given a function \( f(x) = x^3 - 2x^2 + 6x - 4 \)

   To write a program to calculate \( f(x) = x^3 - 2x^2 + 6x - 4 \), we can create a module. In the VBA Session Window, we choose `<Insert> → <Module>` → then type the followings:

   ```vba
   Function fun(x As Double) As Double
       fun = x ^ 3 - 2 * x ^ 2 - 6 * x - 4
   End Function
   ```

   The `As` command can help us to declare the VBA data type:

<table>
<thead>
<tr>
<th>Data Type</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>Byte</td>
<td>0 to 255</td>
</tr>
<tr>
<td>Boolean</td>
<td>True or False</td>
</tr>
<tr>
<td>Integer</td>
<td>-32768 to 32767</td>
</tr>
<tr>
<td>Long (Integer)</td>
<td>-2147483648 to 2147483647</td>
</tr>
<tr>
<td>Single (floating point)</td>
<td>-3.403E38 to –1.404E-45 for negative values</td>
</tr>
<tr>
<td></td>
<td>1.401E-45 to 3.403E38 for positive values</td>
</tr>
<tr>
<td>Double (floating point)</td>
<td>-1.798E308 to –4.941E-324 for negative values</td>
</tr>
<tr>
<td></td>
<td>4.941E-324 to 1.798E308 for positive values</td>
</tr>
<tr>
<td>Currency (scaled integer)</td>
<td>-922337203685477.5808 to 922337203685477.5807</td>
</tr>
<tr>
<td>Decimal</td>
<td>±79228162514264337593543950335 with no decimal point; ±7.9228162514264337593543950 with 28 places to the right of the decimal</td>
</tr>
<tr>
<td>Date</td>
<td>January 1, 100 to December 31, 9999</td>
</tr>
<tr>
<td>String (variable length)</td>
<td>0 to approximately 2 billion</td>
</tr>
<tr>
<td>String (fixed length)</td>
<td>1 to approximately 65400</td>
</tr>
<tr>
<td>Variant (with numbers)</td>
<td>Any numeric value up to the range of a double</td>
</tr>
<tr>
<td>Variant (with characters)</td>
<td>Same range as for variable-length String</td>
</tr>
</tbody>
</table>
Example 2:

The Black Scholes Formula for standard European Call option:

\[ c = S_0 N(d_1) - Ke^{-rT} N(d_2) \]

\[ d_1 = \frac{\ln(S_0 / K) + (r + \sigma^2 / 2)T}{\sigma \sqrt{T}} \]

\[ d_2 = d_1 - \sigma \sqrt{T} \]

Again we choose <Insert> → <Module> → then type the followings:

```vba
Function BSF_Call(S0 As Double, K As Double, r As Double, sigma As Double, T As Double)
    Dim d1 As Double
    Dim d2 As Double

    d1 = (Log(S0 / K) + (r + (sigma ^ 2) / 2) * T) / (sigma * Sqr(T))
    d2 = d1 - sigma * Sqr(T)

    BSF_Call = S0 * Application.WorksheetFunction.NormSDist(d1) - K * Exp(-r * T) * Application.WorksheetFunction.NormSDist(d2)
End Function
```

The Black Scholes Formula for standard European Put option:

\[ c = Ke^{-rT} N(-d_2) - S_0 N(-d_1) \]

\[ d_1 = \frac{\ln(S_0 / K) + (r + \sigma^2 / 2)T}{\sigma \sqrt{T}} \]

\[ d_2 = d_1 - \sigma \sqrt{T} \]

To add more flexibility such that the function can fit both the call and put option, we can use one more input parameter in the function:
The comparison operator:

<table>
<thead>
<tr>
<th>Operator</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;</td>
<td>Less than</td>
</tr>
<tr>
<td>&lt;=</td>
<td>Less than or equal to</td>
</tr>
<tr>
<td>&gt;</td>
<td>Greater than</td>
</tr>
<tr>
<td>&gt;=</td>
<td>Greater than or equal to</td>
</tr>
<tr>
<td>=</td>
<td>Equal to</td>
</tr>
<tr>
<td>&lt;&gt;</td>
<td>Not equal to</td>
</tr>
</tbody>
</table>

Other types of conditional execution commands or Loops structure will be introduced in the future tutorial.
4. Writing Subroutine in Excel:

The difference between subroutine and function is that subroutine allows no parameter, it is executable by pressing the <Run> or designed button. Let's use the Black Scholes Formula again as example:

```vba
Sub BScalculator()
    Dim S0 As Double
    Dim K As Double
    Dim r As Double
    Dim sigma As Double
    Dim T As Double
    Dim porc As Integer

    ' Read in the parameters from the spreadsheet
    S0 = Cells(1, 2).Value          ' current asset price
    K = Cells(2, 2).Value           ' strike price
    r = Cells(1, 4).Value           ' continuously compounding risk-free interest rate
    sigma = Cells(2, 4).Value       ' annualized asset volatility
    T = Cells(1, 6).Value           ' Time to expiration
    porc = Cells(1, 8).Value        ' put or call option

    Dim d1 As Double
    Dim d2 As Double
    Dim Temp As Integer

    d1 = (Log(S0 / K) + (r + (sigma ^ 2) / 2) * T) / (sigma * Sqr(T))
    d2 = d1 - sigma * Sqr(T)
    If porc = 1 Then
        Cells(4, 1) = "European Call Option"
        Cells(4, 3) = S0 * Application.WorksheetFunction.NormSDist(d1) _
        - K * Exp(-r * T) * Application.WorksheetFunction.NormSDist(d2)
    ElseIf porc = 0 Then
        Cells(4, 1) = "European Put Option"
        Cells(4, 3) = K * Exp(-r * T) * _
        Application.WorksheetFunction.NormSDist(-d2) - S0 * _
        Application.WorksheetFunction.NormSDist(-d1)
   ElseIf (porc <> 0 Or porc <> 1) Then
        Temp = MsgBox("Wrong Input of Call or Put Option Choice", vbOKOnly)
    End If
End Sub
```