Introduction to VBA for Excel-Tutorial 6

In this tutorial, you will learn how to structure loops within your code. In other words, a block of code, placed within the loop boundaries, will be executed till the user decide to stop or a pre-prescribed condition is met.

Loops:

Suppose you want to perfrom the same calcuation or calculations many times. One way, of course, to rewrite the statements as many times as needed. Not so effeicent! Another way is to use loop structures. In what follow we will learn multiple ways of looping.

1- ***Do-Loop***: it performs statements enclosed between ***Do*** and ***Loop*** commands. It includes a condition, if satisfied the procedure will exit the loop. The syntax is as follow:

```
Do

    [statements]

    If condition Then Exit Do

    [statements]

Loop
```

Here an example of conditional ***Do-Loop***, where the procedure will exit the loop when x <0. If the condition is not included, the loop would repreat infinitelly.

```
Do
x = x - 1
If x < 0 Then Exit Do
Loop
```

Here an example of counted loop, in this case the loop will perform till the count reach 10 then exit. Again, if the condition is not included, the loop would repreat infinitelly.

```
Do
If i > 10 Then Exit Do
i = 1 + 1
Loop
```

2- ***Do-While-Loop***: instead of using If/Then statement within the Do-Loop structure, we can use ***Do-While-Loop*** where the condition is stated at the beginning of the loop as part of syntax. The syntax is a follow:

```
Do While condition

    [statements]

    [statements]

Loop
```

The the previous examples can be rewriten.

```
Do While x < 0
x = x - 1
Loop
```

And

```
Do While i <= 10
i = i + 1
Loop
```

You will use the Do-While-Loop frequenctly in ME309, where the condition would be the convergence criteria or tolerance. Here a preperatory example since I know some of you can't wait.

Example to calculate the sin(x) using Maclaurin series such that:

$$\sin(x) = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + ...$$

Use tolerance is defined to be the value of the last termto be 1.0e-5. Note that x has be in radians.

```
Sub maclaurin()
Dim x As Single
Dim pi As Single
Dim fact As Single
Dim term As Double
Dim sign As Integer
Dim n As Integer
Dim p As Integer
Dim sum As Double
Dim maxterm As Integer
Dim tol As Single
'inputs
'define the max number of terms
maxterm = 30
tol = 0.00001
x = Val(InputBox("enter angle in radians"))
'logic
'first term
p = 1 'power of the first term and thereafter the power of each term
n = 1 'number of terms, used to restrict the number of terms refer to condition below
fact = 1 '1!=1
term = x 'assign the term to the input
sign = 1 'positive as shown in the Macluarin Series
sum = x
'looping
Do While n < maxterm And Abs(term) > tol
p = p + 2 'we need the odd terms only
n = n + 1 'counting the number of terms
fact = fact * p * (p - 1) 'calculating the factorial
term = x ^ p / fact
sign = -sign 'this will alternate the sign between terms
sum = sum + sign * term
Loop
'output
MsgBox "sin(" & x & ") = " & sum
End Sub
```

Well since you really can't wait to get to ME309, calculate the cos(x) using the general Maclaurin cosine series,

$$\cos(x) = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + ...$$

Limit the series to 30 terms and the tolerance of 0.5 e-6. Good Coding !

I hope your code compares well to mine. I tested mine but if you find any bugs, please let me know.

```
Sub maclaurin()
Dim x As Single
Dim pi As Single
Dim fact As Single
Dim term As Double
Dim sign As Integer
Dim n As Integer
Dim p As Integer
Dim sum As Double
Dim maxterm As Integer
Dim tol As Single
'inputs
'define the max number of terms
maxterm = 30
tol = 0.0000005
x = Val(InputBox("enter angle in radians"))
'logic
'first term
p = 0 'power of the first term and thereafter the power of each term
n = 1 'number of terms, used to restrict the number of terms refer to condition below
fact = 1 '1!=1
term = 1 'assign the term to the input
sign = 1 'positive as shown in the Macluarin Series
sum = 1
'looping
Do While n < maxterm And Abs(term) > tol
p = p + 2 'we need the odd terms only
n = n + 1 'counting the number of terms
fact = fact * p * (p - 1) 'calculating the factorial
term = x ^ p / fact
sign = -sign 'this will alternate the sign between terms
sum = sum + sign * term
Loop
'output
MsgBox "cos(" & x & ") = " & sum
End Sub
```

Before moving to the nexrt loop stucture, try writing a new sine and cosine fucntions using Maclaurin Serie. Allow the user to enter the number of terms.

3- ***For-Next***: if we know the number of iteration, number of times we need to repeat the operations, we can use then definite looping. ***For-Next*** is the perfect fit for such calculations. The syntax is

```
For index = first_value To last_value Step increment
    [statements]
    [statements]
Next index
```

Let say want to calcualte the factorial of a number, you can use the For-Next loop to code the calculation, such that:

```
Sub fact()
'variables declaration
Dim num As Integer
Dim j As Integer
Dim factorial As Single
'input
num = Val(InputBox("Enter a number"))
'logic
factorial = 1
For j = 1 To num Step 1
factorial = factorial * j
Next j
'output
MsgBox num & "! = " & factorial
End Sub
```

Note the Step is set to 1 in the above example. You can use any step size for example, to include the odd numbers from 1 to 11, the syntax would look like:

```
For i = 1 To 11 Step 2
    [statements]
    [statements]
Next i
```

We can also increment in decreasing order. This is very useful espicially for ME309. The syntax would look like:

```
For i = 11 To 1 Step -1
    [statements]
    [statements]
Next i
```

Finally, **Nested Loops** are presented. As the name implies they are multiple loops within each other. In other words, loops placed inside loops.

Suppose, you want to calculate the factorial of 30 consecutive number (i.e. 1 to 30). Moreover, let's assume you want to display the results in the worksheets. The code would be as follow.

```
Sub fact()
'variables declaration
Dim factorial As Double
Dim n As Integer, i As Integer

Cells(1, 1).Value = "n"
Cells(1, 2).Value = "factorial"
factorial = 1
For n = 1 To 30
  factorial = 1
  For i = 1 To n
  factorial = factorial * i
  Next i
Cells(1 + n, 1).Value = n
Cells(1 + n, 2).Value = factorial
Next n
End Sub
```

Before closing, I know most of you would ask, can't we just use an Excel worksheet function to calculate the previous example. Yes! The code would be as follow.

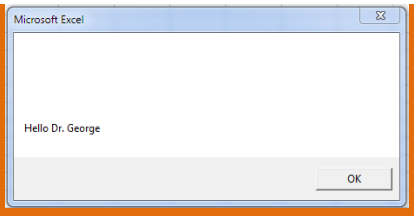
```
Sub fact()
'variables declaration
Dim factorial As Double
Dim n As Integer, i As Integer

Cells(1, 1).Value = "n"
Cells(1, 2).Value = "factorial"
factorial = 1
For n = 1 To 30
factorial = Application.WorksheetFunction.fact(n)
Cells(1 + n, 1).Value = n
Cells(1 + n, 2).Value = factorial
Next n
End Sub
```

Tidbit: If you want to change the size of MsgBox because it does not show the complete title, here the way to do it:

```
MsgBox String$(40, vbTab) & "Hello"
```

Adjust the tab size (i.e. 40) to change the size of the MsgBox accordingly.



Without the String$() command, the box would look like: