INSTITUTE OF ACTUARIAL & QUANTITATIVE STUDIES

Class: BSc

Subject : Application of IT- Basics and Advance Excel

Chapter: Unit 3 Chapter 2

Chapter Name: Loops

Loops

- *In VBA, loops allow you to go through a set of objects/values and analyze it one by one.*

- *Suppose you have a dataset and you want to highlight all the cells in even rows. You can use a VBA loop to go through the range and analyze each cell row number. If it turns out to be even, you give it a color, else you leave it as is.*

- *Here are some more practical examples where VBA loops can be useful:*

1. *Looping through a range of cells and analyzing each cell (highlight cells with a specific text in it).*
2. *Looping through all the worksheets and do something with each (such as protect/unprotect it).*
3. *Loop through all the open workbooks (and save each workbook or close all except the active workbook).*
4. *Loop through all the characters in a cell (and extract the numeric part from a string).*
5. *Loop through all the values an array.*
6. *Loop through all the charts/objects (and give a border or change the background color).*

For Next Loop

- *The 'For Next' loop allows you to go through a block of code for the specified number of times.*
- *For example, if I ask you to add the integers from 1 to 10 manually, you would add the first two numbers, then add the third number to the result, then add the fourth number to the result, as so on..*
- *The same logic is used in the For Next loop in VBA.*
- *Below is the syntax of the For Next loop:*

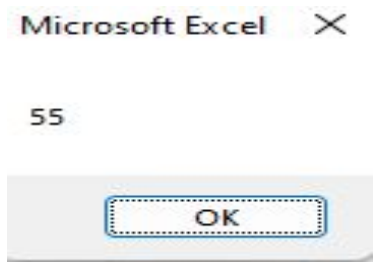
```
For Counter = Start To End [Step Value]
[Code Block to Execute]
Next [counter]
```

- *In the For Next loop, you can use a Counter (or any variable) that will be used to run the loop. This counter allows you to run this loop for a required number of times.*

Example 1 – Adding the first 10 positive integers

- *Below is the code that will add the first 10 positive integers using a For Next loop.*
- *It will then display a message box showing the sum of these numbers.*
- *Once it gets into the loop, it holds the total value after every loop. So after the first loop, when Counter is 1, 'Total' value becomes 1, and after the second loop it becomes 3 (1+2), and so on.*

```
Sub AddNumbers()
Dim Total As Integer
Dim Count As Integer
Total = 0
For Count = 1 To 10
Total = Total + Count
Next Count
MsgBox Total
End Sub
```



Example 2 – Entering Serial Number in the Selected Cells

- *You can also use the For Next loop to go through a collection of objects (such as cells or worksheets or workbooks)*
- *Here is an example that quickly enters serial numbers in all the selected cells.*
- *The above code first counts the number of selected rows and then assigns this value to the variable RowCount. We then run the loop from '1 to RowCount'.*

```
Sub EnterSerialNumber()
Dim Rng As Range
Dim Counter As Integer
Dim RowCount As Integer
Set Rng = Selection
RowCount = Rng.Rows.Count
For Counter = 1 To RowCount
ActiveCell.Offset(Counter - 1, 0).Value = Counter
Next Counter
End Sub
```

‘EXIT For’ Statements in For Next Loops

- *‘Exit For’ statement allows you to exit the ‘For Next’ loop completely.*
- *You can use it in cases where you want the For Next loop to end when a certain condition is met.*
- *Let’s take an example where you have a set of numbers in Column A and you want to highlight all the negative numbers in red font. In this case, we need to analyze each cell for its value and then change the font color accordingly.*
- *But to make the code more efficient, we can first check if there are any negative values in the list or not. If there are no negative values, we can use the Exit For the statement to simply come out of the code..*

```
Sub HghlightNegative()
Dim Rng As Range
Set Rng = Range("A1", Range("A1").End(xlDown))
Counter = Rng.Count
For i = 1 To Counter
If WorksheetFunction.Min(Rng) >= 0 Then Exit For
If Rng(i).Value < 0 Then Rng(i).Font.Color = vbRed
Next i
End Sub
```

Do While Loop

- *A 'Do While' loop allows you to check for a condition and run the loop while that condition is met (or is TRUE).*
- *There are two types of syntax in the Do While Loop.*

```
Do [While condition]
[Code block to Execute]
Loop
```

or

```
Do
[Code block to Execute]
Loop [While condition]
```

- *The difference between these two is that in the first, the While condition is checked first before any code block is executed, and in the second case, the code block is executed first and then the While condition is checked.*
- *This means that if the While condition is False is both the cases, the code will still run at least once in the second case (as the 'While' condition is checked after the code has been executed once).*

Example 1 – Add First 10 Positive Integers using VBA

- *Suppose you want to add the first ten positive integers using the Do While loop in VBA.*
- *To do this, you can use the Do While loop until the next number is less than or equal to 10. As soon as the number is greater than 10, your loop would stop.*
- *Here is the VBA code that will run this Do While loop and the show the result in a message box.*
- *The above loop continues to work until the value of 'i' becomes 11. As soon as it becomes 11, the loop ends (as the While condition becomes False).*
- *Within the loop, we have used a Result variable that holds the final value Once the loop is completed, a message box shows the value of the 'Result' variable.*

```
Sub AddFirst10PositiveIntegers()
Dim i As Integer
i = 1
Do While i <= 10
Result = Result + i
i = i + 1
Loop
MsgBox Result
End Sub
```

Example 2 – Enter Dates For the Current Month

- *Let's say you want to enter all the dates of the current month into a worksheet column.*
- *You can do that by using the following Do While loop code:*
- *The above code would enter all the dates in the D column of the worksheet. The loops continue till the Month value of the variable 'CMDate' matches that of the current month.*

```
Sub EnterCurrentMonthDates()
Dim CMDate As Date
Dim i As Integer
i = 0
CMDate = DateSerial(Year(Date), Month(Date), 1)
Do While Month(CMDate) = Month(Date)
Range("D1").Offset(i, 0) = CMDate
i = i + 1
CMDate = CMDate + 1
Loop
End Sub
```