VBA in Excel

What is VBA?

VBA stands for Visual Basic for Applications. It is a programming language developed by Microsoft that is primarily used for automating tasks in Microsoft Office applications like Excel, Word, and Access. With VBA, you can write scripts to perform repetitive tasks, create custom functions, and develop complex applications within the Office environment.

Basics of Writing VBA

Accessing the VBA Editor:

Open Excel.

Press Alt + F11 to open the VBA Editor.

Alternatively, you can go to the Developer tab (if not visible, enable it from Excel Options), and click on Visual Basic.

Understanding the VBA Editor:

Project Explorer: Shows the hierarchy of your open projects and their components.

Code Window: Where you write your VBA code.

Immediate Window: Useful for debugging and executing code snippets.

Writing a Simple VBA Script

Let's write a simple VBA script that greets the user with a message box.

Create a New Module:

In the VBA Editor, go to Insert > Module. This will create a new module where you can write your code.

Write the Code:

```
Sub GreetUser()
    MsgBox "Hello, welcome to VBA programming!"
End Sub
```

Run the Code:

Close the VBA Editor.

In Excel, go to the Developer tab, click on Macros.

Select GreetUser and click Run.

Example: Automating a Task in Excel

Let's create a VBA script to automate the following task: Copy data from one sheet to another and apply some formatting.

Open the VBA Editor and Insert a New Module.

Write the Code:

```
Sub CopyAndFormatData()
    Dim wsSource As Worksheet
    Dim wsTarget As Worksheet

    ' Define source and target sheets
    Set wsSource = ThisWorkbook.Sheets("Sheet1")
    Set wsTarget = ThisWorkbook.Sheets("Sheet2")

    ' Clear the target sheet
    wsTarget.Cells.Clear

    ' Copy data from source to target
    wsSource.Range("A1:D10").Copy Destination:=wsTarget.Range("A1")
```

```
    ' Apply formatting to target sheet
    With wsTarget.Range("A1:D10")
        .Font.Bold = True
        .Interior.Color = RGB(200, 200, 255)
        .Borders.LineStyle = xlContinuous
    End With

    ' Notify user
    MsgBox "Data copied and formatted successfully!"
End Sub
```

Run the Code:

In Excel, go to the Developer tab, click on Macros.

Select CopyAndFormatData and click Run.

Example 1: Simple Message Box

This example shows how to display a simple message box.

```
Sub ShowMessage()
    MsgBox "Hello, welcome to VBA programming!"
End Sub
```

Example 2: Loop Through Cells

This example shows how to loop through a range of cells and perform an action on each cell.

```
Sub LoopThroughCells()
    Dim cell As Range
    For Each cell In Range("A1:A10")
```

```
    cell.Value = cell.Value * 2
  Next cell
End Sub
```

Example 3: Copy and Paste Data

This example copies data from one sheet to another.

```
Sub CopyData()
  Dim wsSource As Worksheet
  Dim wsTarget As Worksheet

  ' Define source and target sheets
  Set wsSource = ThisWorkbook.Sheets("Sheet1")
  Set wsTarget = ThisWorkbook.Sheets("Sheet2")

  ' Clear the target sheet
  wsTarget.Cells.Clear

  ' Copy data from source to target
  wsSource.Range("A1:D10").Copy Destination:=wsTarget.Range("A1")
End Sub
```

Example 4: Create and Format a Chart

This example creates a chart from data in a worksheet and applies some formatting.

```
Sub CreateChart()
  Dim ws As Worksheet
  Dim chartObj As ChartObject

  ' Define the worksheet
  Set ws = ThisWorkbook.Sheets("Sheet1")
```

```
    ' Create a new chart
    Set chartObj = ws.ChartObjects.Add(Left:=100, Width:=375, Top:=50, Height:=225)

    With chartObj.Chart
        ' Set the chart type
        .ChartType = xlColumnClustered

        ' Set the data source
        .SetSourceData Source:=ws.Range("A1:B10")

        ' Add chart title
        .HasTitle = True
        .ChartTitle.Text = "Sample Chart"

        ' Format the chart
        .ChartStyle = 2
        .ApplyLayout (4)
    End With
End Sub
```

Example 5: User-Defined Function

This example creates a custom function that can be used in Excel formulas.

```
Function MultiplyByTwo(num As Double) As Double
    MultiplyByTwo = num * 2
End Function
```

You can use this function in Excel like any other built-in function. For example, in a cell, you can write =MultiplyByTwo(A1).

Example 6: Sending an Email

This example shows how to send an email using Outlook.

```
Sub SendEmail()
    Dim OutlookApp As Object
    Dim OutlookMail As Object

    ' Create a new instance of Outlook
    Set OutlookApp = CreateObject("Outlook.Application")
    Set OutlookMail = OutlookApp.CreateItem(0)

    With OutlookMail
        .To = "recipient@example.com"
        .Subject = "Test Email"
        .Body = "This is a test email sent from Excel VBA."
        .Send
    End With

    ' Clean up
    Set OutlookMail = Nothing
    Set OutlookApp = Nothing

    MsgBox "Email sent successfully!"
End Sub
```

Macros in Excel

What are Macros in Excel?

Macros in Excel are sequences of instructions that automate tasks. They are written in VBA (Visual Basic for Applications) and can perform repetitive tasks, complex calculations, data manipulation, and more. Macros can save time and reduce errors by automating manual processes.

How to Record a Macro in Excel

Recording a macro is a simple way to create a macro without writing VBA code manually. Excel's Macro Recorder captures your actions and converts them into VBA code.

Steps to Record a Macro

Enable the Developer Tab:

Go to File > Options.

Select Customize Ribbon.

Check the Developer box and click OK.

Start Recording a Macro:

Go to the Developer tab.

Click on Record Macro.

Record Macro Dialog Box:

Macro Name: Enter a name for your macro (e.g., FormatData).

Shortcut Key: Optionally, assign a shortcut key (e.g., Ctrl + Shift + F).

Store Macro In: Choose where to store the macro (This Workbook, New Workbook, or Personal Macro Workbook).

Description: Optionally, add a description.

Click OK to start recording.

Perform Actions:

Perform the actions you want to automate (e.g., formatting cells, entering data, etc.).

Stop Recording:

Go back to the Developer tab.

Click Stop Recording.

Example:

Recording a Macro

Suppose you want to format the header row (A1:D1) with bold text, a background color, and borders.

Start Recording:

Go to the Developer tab.

Click Record Macro.

Name the macro FormatHeader.

Click OK.

Perform Actions:

Select the range A1:D1.

Apply bold formatting.

Change the background color (e.g., yellow).

Add borders.

Stop Recording:

Go back to the Developer tab.

Click Stop Recording.

Running a Recorded Macro

Using the Developer Tab:

Go to the Developer tab.

Click Macros.

Select the macro you recorded (e.g., FormatHeader).

Click Run.

Using a Shortcut Key:

If you assigned a shortcut key, press the key combination (e.g., Ctrl + Shift + F).

Viewing and Editing the Macro Code

You can view and edit the VBA code generated by the Macro Recorder.

Open the VBA Editor:

Press Alt + F11 to open the VBA Editor.

Locate the Macro:

In the Project Explorer, find the module where your macro is stored (usually in Modules).

Double-click the module to view the code.

Here's an example of what the recorded macro code might look like:

```
Sub FormatHeader()
    Range("A1:D1").Select
    With Selection
        .Font.Bold = True
        .Interior.Color = RGB(255, 255, 0)
        .Borders.LineStyle = xlContinuous
    End With
End Sub
```

Prepared By,

Ahamed Basith