**VBA Macros in Excel (Basics to Intermediate)**

**1. Introduction to VBA and Macros**

**What is VBA?**

VBA (Visual Basic for Applications) is a programming language built into Microsoft Excel that allows users to automate repetitive tasks, create custom functions, and interact with Excel beyond its standard functionalities.

**What are Macros?**

A macro is a recorded or written sequence of commands and functions that automate repetitive actions in Excel. Macros are powered by VBA.

**Why Use VBA in Excel?**

* **Automate repetitive tasks** (e.g., formatting, data entry, report generation).
* **Create custom functions** beyond Excel’s built-in formulas.
* **Enhance interactivity** by creating buttons, forms, and user inputs.
* **Integrate with other applications** (e.g., Outlook, Word, PowerPoint).

In **VBA (Visual Basic for Applications)**, Sub stands for **"Subroutine"**.

A **Subroutine** is a block of code that performs a specific task but does not return a value. It is used to execute a sequence of VBA statements when called.

**Syntax of a Subroutine in VBA**

Sub SubroutineName()

' Code to execute

End Sub

Example:

Sub GreetUser()

MsgBox "Hello, welcome to VBA!"

End Sub

🔹 *This subroutine displays a message box with a greeting when executed.*

**Types of Procedures in VBA**

1. **Sub Procedures (Sub)** → Do not return values.
2. **Function Procedures (Function)** → Return values.

Example of a Function:

Function AddNumbers(a As Integer, b As Integer) As Integer

AddNumbers = a + b

End Function

🔹 *This function returns the sum of two numbers, unlike a Sub which does not return a value.*

**2. Enabling Developer Tab & Recording Macros**

**How to Enable Developer Tab in Excel?**

1. Open Excel, go to **File > Options**.
2. Click on **Customize Ribbon**.
3. Check **Developer** and click **OK**.

**How to Record a Macro?**

1. Go to the **Developer Tab**.
2. Click on **Record Macro**.
3. Name the macro, choose a shortcut key (optional), and select where to store it (This Workbook, New Workbook, or Personal Macro Workbook).
4. Perform actions (e.g., formatting, typing data, etc.).
5. Click **Stop Recording**.

*Note:* Recorded macros generate VBA code automatically, which can be viewed by pressing ALT + F11 to open the VBA Editor.

**3. Basics of VBA Programming**

**Opening the VBA Editor**

* Press ALT + F11 to open the **VBA Editor**.
* Insert > Module to create a new VBA module.

**Writing a Simple Macro in VBA**

Sub HelloWorld()

MsgBox "Hello, welcome to VBA macros!"

End Sub

*This macro displays a message box when run.*

**Understanding VBA Syntax**

* Sub MacroName() → Defines a macro (subroutine).
* End Sub → Ends the macro.
* MsgBox "Message" → Displays a message box.
* Dim → Declares a variable.

**Variables in VBA**

Variables store values for calculations and data manipulation.

Sub DeclareVariables()

Dim userName As String

userName = "Akshat"

MsgBox "Hello, " & userName

End Sub

*Stores a name in a variable and displays it in a message box.*

**4. Automating Tasks Using VBA**

**Example 1: Auto-Fill Data in a Sheet**

This macro enters data automatically into a worksheet.

Sub AutoFillData()

Range("A1").Value = "Name"

Range("A2").Value = "John"

Range("A3").Value = "Sarah"

Range("A4").Value = "David"

End Sub

*Fills column A with names.*

**Example 2: Formatting a Table Automatically**

Sub FormatTable()

With Range("A1:C10")

.Font.Bold = True

.Interior.Color = RGB(200, 200, 255)

End With

End Sub

*Applies bold formatting and a blue background to cells A1:C10.*

**5. Loops and Conditional Statements in VBA**

Loops and conditions help automate repetitive actions dynamically.

**For Loop (Repeat Action Multiple Times)**

Sub LoopExample()

Dim i As Integer

For i = 1 To 10

Cells(i, 1).Value = i

Next i

End Sub

*Fills column A with numbers from 1 to 10.*

**If-Else Condition (Decision Making)**

Sub CheckValue()

Dim value As Integer

value = Range("A1").Value

If value > 50 Then

MsgBox "Value is greater than 50"

Else

MsgBox "Value is 50 or less"

End If

End Sub

*Checks the value in A1 and shows an appropriate message.*

**6. User Interaction – Input & Buttons**

**Taking Input from Users**

Sub GetUserInput()

Dim userName As String

userName = InputBox("Enter your name:")

MsgBox "Hello, " & userName

End Sub

*Prompts the user to enter their name and displays a greeting.*

**Creating a Button to Run a Macro**

1. Go to **Developer Tab > Insert > Button**.
2. Click and draw a button on the sheet.
3. Assign the macro to the button.

**7. Automating Reports Using Macros**

**Example: Generating a Summary Report**

Sub CreateReport()

Dim ws As Worksheet

Set ws = ActiveSheet

ws.Range("A1").Value = "Employee"

ws.Range("B1").Value = "Salary"

ws.Range("A2").Value = "John"

ws.Range("B2").Value = 50000

ws.Range("A3").Value = "Emma"

ws.Range("B3").Value = 60000

MsgBox "Report Generated Successfully!"

End Sub

*Creates a simple employee salary report.*

**8**. Sending Automated Emails from Excel

Example: Sending an Email Using Outlook

Sub SendEmail()

Dim OutlookApp As Object

Dim Mail As Object

Set OutlookApp = CreateObject("Outlook.Application")

Set Mail = OutlookApp.CreateItem(0)

With Mail

.To = "example@email.com"

.Subject = "Automated Report"

.Body = "Hello, this is an automated email from VBA."

.Send

End With

MsgBox "Email Sent!"

End Sub

*Sends an email automatically using Outlook.*

**9. Error Handling in VBA**

**Example: Handling Errors Using On Error**

Sub ErrorHandlingExample()

On Error Resume Next

Dim x As Integer

x = 10 / 0 ' This will cause an error

If Err.Number <> 0 Then

MsgBox "An error occurred: " & Err.Description

End If

End Sub

*Prevents code from stopping due to errors and displays a message instead.*

**10. Real-World Use Cases of VBA Macros**

**Data Cleaning:** Remove duplicates, format columns, and standardize text.  
 **Report Automation:** Generate monthly reports and summaries.  
 **Stock Management:** Update inventory based on sales.  
 **Financial Analysis:** Automate calculations for budgets and forecasts.  
 **Email Automation:** Send notifications and reminders automatically.

**Conclusion**

* VBA Macros save time by automating repetitive tasks.
* They enhance Excel's functionality by enabling custom automation.
* Even basic VBA knowledge can significantly improve productivity.