



Macro Programming

Vignesh V

Department of Computer Applications

vigneshv@pes.edu

Macro Programming

**Experiential Learning — Building &
Testing UserForms + Error Handling
(Breakpoints, Watches, Debugging)**

Vignesh V

Department of Computer Applications



Why Test UserForms?

- UserForms involve user input — more prone to runtime errors.
- Testing ensures correct event triggers, data validation, and logical flow.
- Prevents crashes or wrong data entries.



Common UserForm Errors

Error Type	Cause	Example
Runtime Error	Invalid operation	Divide by zero
Type Mismatch	Wrong data type	Text entered for numeric input
Object Required	Missing reference	Control not named correctly
Out of Range	Invalid sheet or cell	Refers to deleted worksheet



Macro Programming

Setting Breakpoints

- A **breakpoint** pauses code execution at a chosen line.
- Helps inspect variable values and control flow.

Steps:

1. Click the **left margin** beside a line or press **F9**.
2. Run the form — execution pauses at the breakpoint.
3. Press **F8** to step through.



Macro Programming

Setting Breakpoints

Microsoft Visual Basic for Applications - techonthenet_2016.xlsm - [Module1 (Code)]

File Edit View Insert Format Debug Run Tools Add-Ins Window Help

Ln 28, Col 22

Project - VBAProject

(General) AlphaNumeric

```
' Example provided by techonthenet.com

Function AlphaNumeric(pValue) As Boolean

    Dim LPos As Integer
    Dim LChar As String
    Dim LValid_Values As String

    'Start at first character in pValue
    LPos = 1

    'Set up values that are considered to be alphanum
    LValid_Values = " abcdefghijklmnopqrstuvwxyzABCDE

    'Test each character in pValue
    While LPos <= Len(pValue)

        'Single character in pValue
        If LChar = " " Then
            AlphaNumeric = False
            Exit Function
        End If

        If LChar <= "Z" And LChar >= "A" Then
            AlphaNumeric = True
        Else
            AlphaNumeric = False
            Exit Function
        End If

        LPos = LPos + 1
    End While

    AlphaNumeric = True
End Function
```

Properties - Module1

Module1 Module

Alphabetic Categorized

(Name) Module1

Immediate Watches

Expression Value Type Context

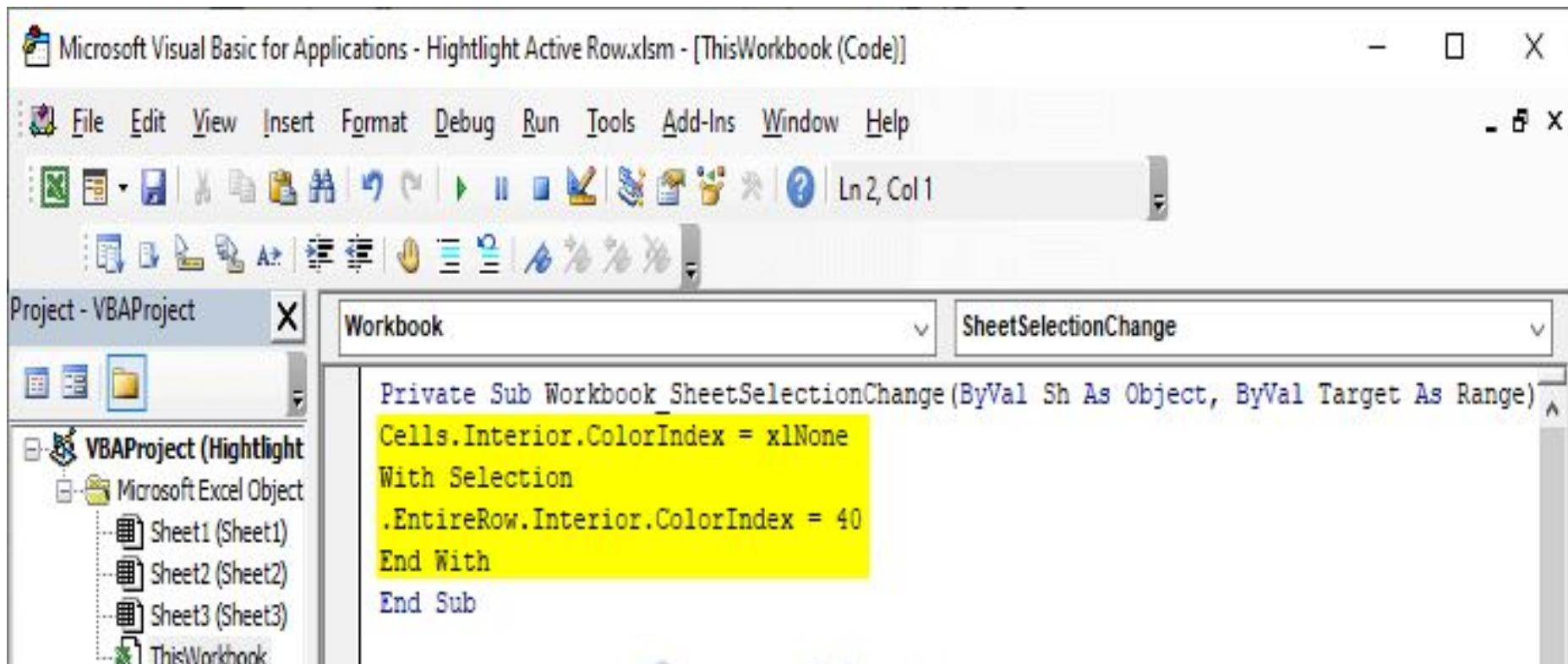


Using Step-Through Debugging

- Press **F8** to execute code **one line at a time**.
- Watch values change dynamically in the Locals or Immediate window.
- Great for isolating logic or loop errors.



Using Step-Through Debugging



Microsoft Visual Basic for Applications - Hightlight Active Row.xlsxm - [ThisWorkbook (Code)]

File Edit View Insert Format Debug Run Tools Add-Ins Window Help

Ln 2, Col 1

Project - VBAPrj

Workbook SheetSelectionChange

```
Private Sub Workbook_SheetSelectionChange(ByVal Sh As Object, ByVal Target As Range)
    Cells.Interior.ColorIndex = xlNone
    With Selection
        .EntireRow.Interior.ColorIndex = 40
    End With
End Sub
```



The Immediate Window

Test snippets or print variable values in real time.

- **Example:**

```
? Me.txtName.Value  
Debug.Print "Next Row: " & nextRow
```

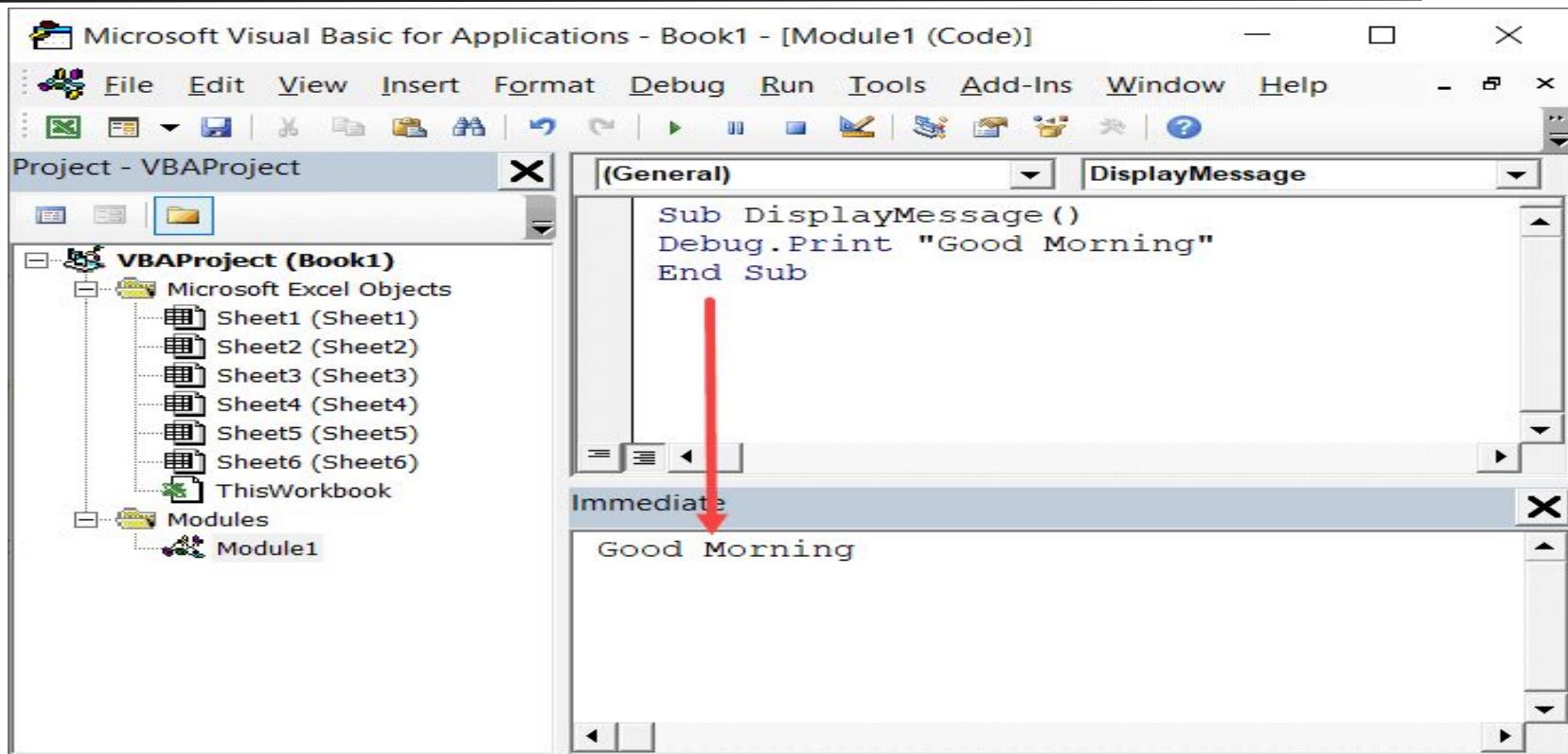
Use Cases:

- Check form control values.
- Verify calculated results.
- Troubleshoot without stopping code.



Macro Programming

The Immediate Window



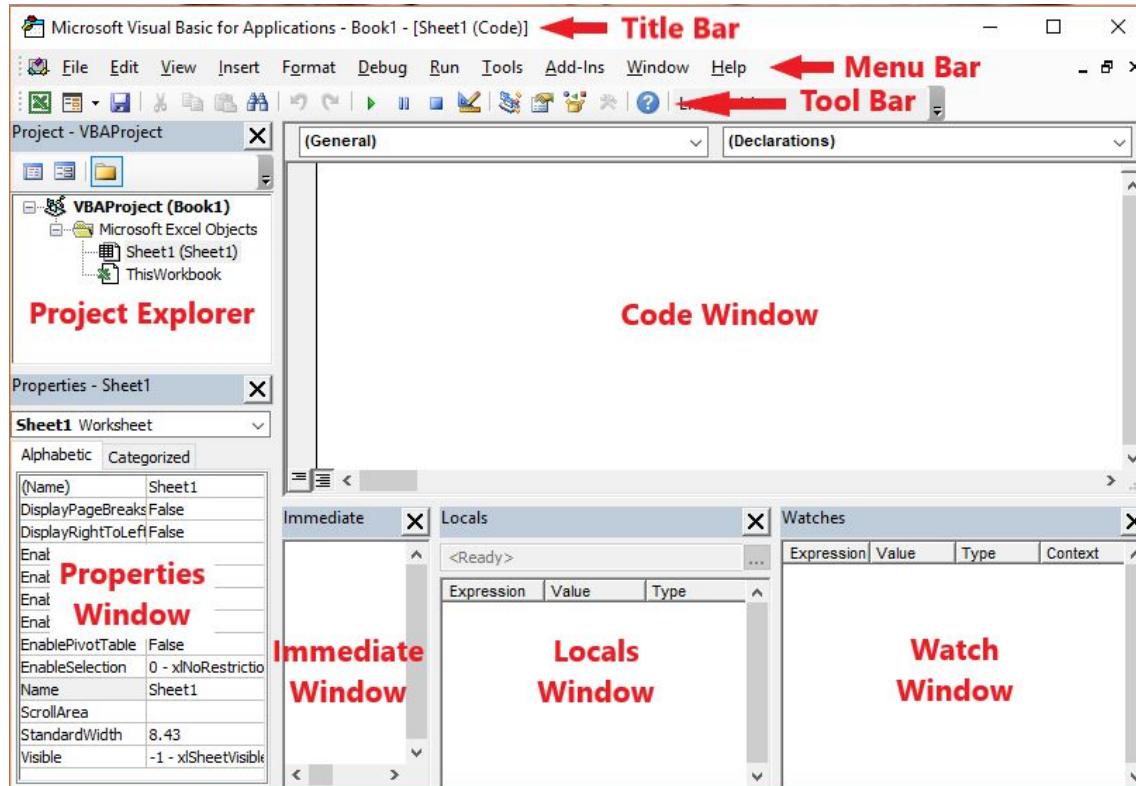
The screenshot shows the Microsoft Visual Basic for Applications (VBA) interface. The title bar reads "Microsoft Visual Basic for Applications - Book1 - [Module1 (Code)]". The menu bar includes File, Edit, View, Insert, Format, Debug, Run, Tools, Add-Ins, Window, and Help. The toolbar below the menu bar contains various icons for file operations, such as Open, Save, Print, and Find. The Project Explorer on the left shows the "VBAProject (Book1)" with "Microsoft Excel Objects" (Sheet1 to Sheet6, ThisWorkbook) and a "Modules" section containing "Module1". The code editor window displays a subroutine "DisplayMessage" with the following code:

```
Sub DisplayMessage()
    Debug.Print "Good Morning"
End Sub
```

Below the code editor is the Immediate window, which displays the output "Good Morning". A red arrow points from the text "Good Morning" in the Immediate window back up to the line "Debug.Print" in the code editor, highlighting the connection between the code and its execution.



Watch and Locals Windows





Watch and Locals Windows

```
public int PriceAfterDiscount(Customer customer, int initialPrice)
{
    if (customer.IsVIPCustomer)
    {
        var discount = initialPrice * (_vipDiscountPercent / 100.0);
        return initialPrice - (int)discount;
    }
    else
    {
        return initialPrice;
    }
}
```

Watch 1			
Search (Ctrl+E)		Search Depth: 3	▼
Name	Value	Type	▼
customer	{DebuggingFundamentalsPart2.Customer}	DebuggingFundamentalsPart2.Customer	▼
customer.IsVIPCustomer	true	bool	▼
customer.Age	35	int	▼
customer.Age + 12 - 6	41	int	▼
initialPrice	250	int	▼
initialPrice * (_vipDiscountPercent / 100.0)	25	double	▼



Error Handling

- Use structured error handling to manage unexpected errors.

```
On Error GoTo ErrHandler
```

```
' Code block
```

```
Exit Sub
```

```
ErrHandler:
```

```
    MsgBox "Error: " & Err.Description, vbCritical
```



Error Handling Example

```
Private Sub cmdSubmit_Click()
    On Error GoTo ErrHandler

    Dim age As Integer
    age = CInt(Me.txtAge.Value)
    MsgBox "Age: " & age
    Exit Sub

ErrHandler:
    MsgBox "Please enter a valid number!", vbCritical
End Sub
```



On Error Resume Next

Use only for non-critical errors.

On Error Resume Next

```
ws.Cells(1, 1).Value = Me.txtName.Value
```

On Error GoTo 0

- Skips over the error but should be used carefully — never for debugging suppression.



Experiential Task 1: Debug the Form

- Add intentional error (e.g., convert text to number).
- Insert breakpoints and step through code.
- Observe `Err.Description` output.



Experiential Task 2: Add Error Handling

Task:

1. Add a general error handler to your Submit button.
2. Display a meaningful error message if data is missing.
3. Log errors to a worksheet named “ErrorLog”.



Experiential Task 3: Watch Window Practice

Task:

1. Add a Watch for variable `nextRow`.
2. Run form and observe its value as data is entered.
3. Note how variable updates during execution.



Macro Programming

Vignesh V

Department of Computer Applications

vigneshv@pes.edu