```
====================================================
VBA - VARIABLES - LOCAL VARIABLES
====================================================
```

VARIABLE SCOPE (VERY IMPORTANT)

```
----------------------------------------------------
```

Procedure-level (Local)
• Declared inside a Sub or Function
• Available only there
• Deleted when procedure ends

Module-level
• Declared at top of a Module using Dim or Private
• Available to all Subs in that module
• Stays in memory while workbook open

Public (Global)
• Declared with Public at the top of a Module
• Available everywhere in the project
• Stays in memory while workbook open

---

1. LOCAL VARIABLES

```
----------------------------------------------------
```

Local variables are declared inside a procedure (Sub or Function)
and are only available while that procedure runs.

```
Sub LocalVariableExample()
    Dim message As String   ' Local variable
    message = "Hello! I am a Local Variable"
    MsgBox message
End Sub
```

How It Works:
• Declared with Dim inside Sub
• Only exists while Sub runs
• Cannot access from other Subs
• Memory freed when Sub ends

Example: Each Sub has its own local variable

```
Sub Sub1()
    Dim x As Integer
    x = 10
    MsgBox x
End Sub

Sub Sub2()
    Dim x As Integer      ' Different x than Sub1
    x = 20
    MsgBox x
End Sub
```

Both have variable x, but they are different

WHY USE LOCAL VARIABLES?

✓ Don't interfere with other Subs
✓ Memory efficient (freed when done)
✓ Easier to manage

✓ Less confusion
✓ Best practice

SCOPE COMPARISON TABLE

Scope	Where Declared	Available To	Memory
Local	Inside Sub/Function	Only that Sub	Freed when Sub ends
Module	Top of Module (Dim)	All Subs in Module	Until workbook close
Public	Top of Module (Public)	Entire Project	Until workbook close

EXAMPLES - ALL THREE SCOPES

```
' At top of Module (outside all Subs)
Public globalName As String         ' Public/Global
Dim moduleName As String            ' Module-level

Sub SubA()
    Dim localVar As String          ' Local to SubA

    localVar = "I am local"
    moduleName = "Module level"
    globalName = "Global"

    MsgBox localVar     ' Works
    MsgBox moduleName   ' Works
    MsgBox globalName   ' Works
End Sub

Sub SubB()
    Dim localVar As String          ' Different local var

    ' MsgBox localVar   ' ERROR - not set in SubB
    MsgBox moduleName   ' Works
    MsgBox globalName   ' Works
End Sub
```

LOCAL VARIABLE LIFECYCLE

```
Sub LifecycleExample()
    ' Variable created here
    Dim counter As Integer

    counter = 0
    ' Use variable
    counter = counter + 1
    MsgBox counter

    ' Sub ends, variable destroyed
End Sub
```

```
' After Sub ends, counter no longer exists
```

KEY DIFFERENCES

LOCAL VARIABLES
→ Inside Sub/Function
→ Declared with Dim
→ Only that Sub can use
→ Deleted when Sub ends
→ Memory efficient
→ Best practice

MODULE-LEVEL VARIABLES
→ Top of Module (before Subs)
→ Declared with Dim or Private
→ All Subs in module can use
→ Stays in memory
→ Use when multiple Subs need it

PUBLIC/GLOBAL VARIABLES
→ Top of Module (before Subs)
→ Declared with Public
→ Any Sub in any Module can use
→ Stays in memory
→ Use rarely (can cause confusion)

PRACTICAL EXAMPLE - LOCAL VARIABLES

```
Sub CalculateTotal()
    ' Local variables
    Dim price As Double
    Dim quantity As Integer
    Dim total As Double

    price = 100
    quantity = 5
    total = price * quantity

    MsgBox "Total: " & total
End Sub

' Variables only available in this Sub
' Deleted when Sub ends
```

QUICK REFERENCE

Local Variable:
```
Sub MySub()
    Dim x As Integer
    x = 10
End Sub
```

Module-Level Variable:
```
Dim y As Integer    ' At top of module
```

```
Sub MySub()
    y = 20          ' Can use here
End Sub
```

Public Variable:

```
Public z As Integer ' At top of module

Sub MySub()
    z = 30          ' Can use anywhere
End Sub
```

IMPORTANT POINTS

---

1. Local = most common and best practice
2. Always declare variables with Dim inside Sub
3. Different Subs can have same variable name
4. Local variables don't interfere with others
5. Memory is freed when Sub ends
6. Use Option Explicit to force declaration
7. Module-level for shared data only
8. Public for rare situations only
9. Local is always preferred
10. Less scope = better programming

BEST PRACTICES

✓ Use Local variables whenever possible
✓ Declare inside Sub, not at module top
✓ Use meaningful names
✓ Always use Option Explicit
✓ Initialize variables before use
✓ Keep scope as small as possible
✓ Avoid Public variables
✓ Use Module-level only when necessary
✓ Document why Module/Public variables exist

==
FILE: VBA_Local_Variables_Complete.txt
==