**Module 3: VBA Programming Basics**

* **Syntax, keywords, variables, constants**

1. **Syntax**

It means the rules for writing code so that VBA can understand and execute it correctly.

VBA (Visual Basic for Applications) follows a specific structure (syntax).

* **Each statement** generally goes on a new line.
* **Keywords** (predefined words) must be spelled correctly.
* Code is written inside **Sub procedures** or **Functions**.

**Example:**

Sub HelloWorld()

MsgBox Range("B1").Value

End Sub

**2.Comments**

Used to explain code. They start with a single apostrophe (').

' This is a comment

**3. Keywords in VBA**

Keywords are reserved words in VBA with predefined meaning.  
Examples:

* Sub, Function, End, If, Then, Else, For, Next, Dim, Set, Loop.

👉 You cannot use keywords as variable names.

**4. Variables and Constants**

**Variables**

* Variables are used to store values temporarily.
* Declared with Dim.

**Example:**

Sub VariableExample()

Dim studentName As String

studentName = "Amit"

MsgBox "Student Name: " & studentName

End Sub

**Constants**

* Constants are fixed values that cannot be changed during program execution.
* Declared with Const.

**Example:**

Sub ConstantExample()

Const PI As Double = 3.14159

Dim radius As Double

radius = Range(“B9”).value

MsgBox "Area of Circle: " & PI \* radius ^ 2

End Sub

* **Data types in VBA**

In **VBA (Visual Basic for Applications)**, data types define what kind of data a variable can store (numbers, text, dates, etc.). Using the correct data type improves performance and reduces memory usage.

**1.Integer**

* Stores whole numbers from **-32,768 to 32,767**

Sub Example\_Integer()

Dim count As Integer

count = 120

MsgBox "Count: " & count

End Sub

1. **Long**

* Stores large whole numbers from **-2,147,483,648 to 2,147,483,647**

Sub Example\_Long()

Dim population As Long

population = 1456789

MsgBox "Population: " & population

End Sub

1. **Single**

* Stores **decimal numbers** (approx. 7 digits precision)

Sub Example\_Single()

Dim temperature As Single

temperature = 36.55568675

MsgBox "Temperature: " & temperature

End Sub

1. **Double**

* Stores **decimal numbers** with more precision (approx. 15 digits)

Sub Example\_Double()

Dim distance As Double

distance = 12345.6789458958685695

MsgBox "Distance: " & distance

End Sub

1. **Currency**

* Stores monetary values (up to 4 decimal places, very accurate for money)

Sub Example\_Currency()

Dim price As Currency

price = 199.99

MsgBox "Price: " & price

End Sub

1. **String**

* Stores text (up to 2 billion characters)

Sub Example\_String()

Dim studentName As String

student Name = "Amit"

MsgBox "Student Name: " & studentName

End Sub

1. **Boolean**

* Stores **True** or **False**

Example-1

Sub Example\_Boolean()

Dim isPassed As Boolean

isPassed = True

MsgBox "Result: " & isPassed

End Sub

Example-2

Sub Button16\_Click()

Dim marks As Integer

Dim isPassed As Boolean

'Read marks from cell B1

marks = Range("J12").Value

'Check pass/fail

isPassed = (marks >= 40)

If isPassed Then

MsgBox "Congratulations! You passed with " & marks & " marks.", vbInformation

Else

MsgBox "Sorry! You failed with " & marks & " marks.", vbExclamation

End If

End Sub

1. **10. Variant**

* Can hold **any type of data** (string, number, date, etc.)
* Uses more memory, but flexible

Sub Example\_Variant()

Dim myData As Variant

myData = "Hello"

MsgBox myData

myData = 123

MsgBox myData

End Sub

* **Operators (Arithmetic, Comparison, Logical)**

**a) Arithmetic Operators**

| **Operator** | **Meaning** | **Example (a=10, b=3)** |
| --- | --- | --- |
| + | Addition | a + b = 13 |
| - | Subtraction | a - b = 7 |
| \* | Multiplication | a \* b = 30 |
| / | Division | a / b = 3.33 |
| \ | Integer Division | a \ b = 3 |
| Mod | Remainder | a Mod b = 1 |
| ^ | Exponent | a ^ b = 1000 |

Sub ArithmeticExample()

Dim a As Integer, b As Integer

a = Range("A1").Value

b = Range("A2").Value

MsgBox "Addition: " & (a + b) & vbCrLf & \_

"Division: " & (a / b) & vbCrLf & \_

"Remainder: " & (a Mod b)

End Sub

**b) Comparison Operators**

Used to compare values (Result = True / False).

| **Operator** | **Meaning** | **Example (a=10, b=3)** |
| --- | --- | --- |
| = | Equal | a = b → False |
| <> | Not Equal | a <> b → True |
| > | Greater than | a > b → True |
| < | Less than | a < b → False |
| >= | Greater or Equal | a >= b → True |
| <= | Less or Equal | a <= b → False |

Sub ComparisonExample()

Dim x As Integer, y As Integer

x = Range("B6").Value

y = Range("B7").Value

If x < y Then

MsgBox "x is smaller than y"

Else

MsgBox "y is Smaller than or equal to x"

End If

End Sub

**c) Logical Operators**

Used to combine conditions.

| **Operator** | **Meaning** | **Example (a=10, b=5)** |
| --- | --- | --- |
| And | True if both are True | (a > b) And (b > 0) → True |
| Or | True if any one is True | (a < b) Or (b > 0) → True |
| Not | Reverses condition | Not (a > b) → False |

AND

Sub Example\_AND()

Dim x As Long, y As Long

' Read values from Excel

x = CLng(Range("B10").Value)

y = CLng(Range("B11").Value)

' Apply logical condition

If x > 30 And y > 50000 Then

MsgBox "Both conditions are True"

Else

MsgBox "At least one condition is False"

End If

End Sub

OR

Sub Example\_AND()

Dim x As Long, y As Long

' Read values from Excel

x = CLng(Range("B10").Value)

y = CLng(Range("B11").Value)

' Apply OR logical condition

If x > 30 Or y > 50000 Then

MsgBox "At least one condition is True"

Else

MsgBox "Both conditions are False"

End IfEnd Sub

Not

Sub Example\_NOT()

Dim x As Long, y As Long

' Read values from Excel

x = CLng(Range("B10").Value)

y = CLng(Range("B11").Value)

' Apply NOT logical condition

If x > 30 Not y > 50000 Then

MsgBox "Both conditions are False"

Else

MsgBox "At least one condition is True"

End If

End Sub