Nikhil Analytics
Performance Driven By Values

Visual Basic for Applications Class Notes - 1

What is Visual Basic for Applications (VBA)?

Visual Basic for application - It is a high-level programming environment developed by Microsoft Corporation. Visual Basic provides the computer power-user with a set of robust tools that are almost as versatile as the developer's imagination. Visual Basic for Applications (VBA)

1. It is introduced in 1996 by Microsoft.
2. It is a subset of VB.
3. It is the common programming language used to manipulate other Microsoft Office applications like Word, Excel and Power Point.
4. It contains full-featured development environment in Excel
5. It was presented as the "glue" to integrate the Microsoft Office applications

Advantages and Disadvantages of VBA:

Advantages:

1. Programming can be started using MS-Office without installing Additional software
2. Easy to Learn
3. Excel application and database platform from Excel can be easily upgraded to other high-end databases like Access, SQL Server, Oracle, and MySQL.
4. Excel is capable of connecting directly to OLAP databases and can be integrated in Pivot Tables.
5. Excel is rich in financial functions like Fixed Assets Depreciation, Amortization, etc. No need to create your own financial modules.
6. Excel is portable. You can send it to someone through email.

Disadvantages:

1. Slow program execution
2. Excel files with macros are not robust. It may be lost or corrupted very easily.
3. Excel worksheet can hold only limited number of Records (limitation of excel).
4. Excel file with macro sent via email can be accessed by unauthorized persons (Does not provide any security, But you can protect your VBA code by giving password)

VBA Development Environment

1. Visual Basic Editor
2. Project Explorer

3. Code Window
4. Properties Window
5. Object Browser
6. IntelliSense

Visual Basic Editor

It is accessible by two ways-

1. Selecting Menu item Tools -> Macro -> Visual Basic Editor OR
2. Press Alt + F11 on your keyboard.

Writing first VBA code:

```
Sub showMessage()
‘ This is the First VBA Program
        MsgBox  "Hello World!"
End Sub
```

To execute code:

1. Click the Run button
2. press [F5], or
3. go to Run..Run Sub/UserForm to run the program

Comments in VBA:

A comment is a piece of text in code that would not be considered when reading your code.

1. **‘** (Single Quote) - is used to as a comment symbol

   Syntax:

   ```
   ' This line will not be considered as part of the code
   ```

2. **Rem** keyword is used to add the comments

   Syntax:

   ```
   Rem I can write anything I want on this line
   ```

Fundamentals of Objects:

Objects are the fundamental building blocks of Visual Basic. An object is a special type of variable that contains both data and codes. Each Object will have their own properties and will perform some actions.
Methods and Properties A Property represents a built-in or user-defined characteristic of the object. A method is an action that you perform with an object.
A **collection** is a group of similar type of objects or belongs to the same class. Group Members will share common properties and methods.

Ex-

Object – CAR

Properties – MODEL, COLOR

Methods – MOVE, START, STOP

Collection – group of cars

Excel Objects used in the VBA Programming:

1. Workbook
    a. It is a collection of all Workbook objects
    b. It represents all the currently opened workbooks
    c. In VBA it is referenced by
    Workbooks("Book1")
    Workbooks("Book2")
2. Worksheet
    a. It is a collection of Worksheet objects
    b. It represents a worksheet
    c. A Worksheet can be referenced in the following way
    Worksheets("Sheet1") Or
    Worksheets(1) - It is referred using the index number
3. Sheet
    a. Represents a worksheet or chart sheet
    Sheets(1) - It refers to the worksheet
4. Range
    a. It represents a cell, a row, a column, a selection of cells containing one or more contiguous blocks of cells, or a 3-D range.

Examples –

Worksheets("Sheet2").Range("A1:B5") = "AB"

Workbook Properties and Methods:

Close Method – It close the active workbook Syntax: Workbooks. Close

Count property – It returns the number of workbooks that are currently opened

Syntax: Workbooks. Count

Range Object Properties:

1. **Cells** - The Cells property takes one or two indexes as its parameters.
    Syntax Cells (index) or Cells (row, column)
    Examples –
        ActiveSheet.Range.Cells(1,1)

Range. Cells (1,1) Cells(1,1)

Range("A1") = 123 and Cells(1,1) = 123

- Refers to the cell A1

2. **Offset Property** – It is used to Move the active cell in the worksheet

Examples –

ActiveCell.Offset (1, 0) = 1 -> place a '1' below one row from active cell (Current selected cell)

ActiveCell.Offset(0,1) = 1 -> place a '1' one column right to the active cell

ActiveCell.Offset(0,-3) = 1 -> place a '1' three column left to the active cell

3. **Value Property** – It returns the value in the specified cell

Examples –

Range ("A1") = 1 and Range ("A1").Value = 1

Building procedures

1. What is Procedure
2. Introduction to Sub Procedure
3. Calling procedures
4. Passing arguments to procedures
5. Introduction to the function procedure

What is a procedure? A procedure describes a unit of VBA code that automates a task. It is a stand-alone segment of code that holds a series of VBA commands.

Types of Procedure

1. Subroutine procedure
2. Function procedures

Subroutine Procedure:

It is an individual block, or unit, of VBA code that performs a specific task **but does not return a value.** Subroutines always begin with the keyword Sub and end with the statement End Sub Subroutines takes parameters (optional)

Syntax:

Sub <ProcedureName>()

VBA Statements

End Sub

Example –

```
Sub ShowTime()
    Range("C1") = Now()
End Sub
```

Calling the Procedure:

Syntax:

<ProcedureName>()

Example –

```
Sub z(a)
    MsgBox a
End Sub

Sub x()
    Call z("ABC")  -> Calling a subroutine name z
End Sub
```

Passing the Parameters to the Procedure

Parameters can be passed to the procedure by using following method

1. Callby Value
2. Callby Reference

Callby value Method:

In this method variables are used to pass parameters. The procedure will access the copy of the original values in the memory. As a Result, Changes done to the values will not have any impact on the original values.

ByVal Keyword is used to pass the values in this method.

Examples –

```
Sub TestPassing2 ()
    Dim y As Integer
    y = 50
    AddNo2 y
    MsgBox y
End Sub
```

```
Sub AddNo2 (ByVal x As Integer)
        x = x + 10
End Sub
```

Callby Reference Method

In this method the procedure will access the actual variable in memory. As a result, the variable's value can be changed by the procedure. Passing by reference is the default in VBA.

ByRef KeyWord is used to pass the parameters

Example –

```
Sub TestPassing1()
        Dim y As Integer
        y = 50
        AddNo1 y
        MsgBox y
End Sub

Sub AddNo1(ByRef x As Integer)
        x = x + 10
End Sub
```

Function Procedure:

Like, Subroutine function procedure takes parameters and performs a specific task. **Function always returns the value.** Function statements will be placed between the Function and End Function statements

Syntax:

```
Function <FunctionName>(Paramaters)
        VBA Statements
End Function
```

Example –

```
Function sumNo(x, y)
        sumNo = x + y
End Function
```

Working with Message and Input Boxes:

InputBox: It will display a message box where the user can enter a value or a message in the form of text.

Syntax:

myMessage=InputBox(Prompt, Title, default_text, x-position, y-position)

Prompt - The message displayed normally as a question asked.

Title - The title of the Input Box.

Default-text - The default text that appears in the input field where users can use it as his intended input or he may change to the message he wish to key in.

x-position and y-position - the position or the coordinate of the input box

myMessage – It is a variant data type but typically it is declared as string, which accept the message input by the users.

Example –

```
Dim userMsg As String
userMsg = InputBox("What is your message?", "Message Entry Form", "Enter your messge here", 500, 700)
```

MsgBox() Function:

It produces a pop-up message box and prompt the user to click on a command button before he /she can continues.

Syntax:

yourMsg=MsgBox(Prompt, Style Value, Title)

Prompt – Displays the Message in the messagebox

Style – It determines what type of command buttons appear on the message box.

Style Values	Named Constant	Buttons Displayed
0	vbOkOnly	Ok button
1	vbOkCancel	Ok and Cancel buttons
2	vbAbortRetryIgnore	Abort, Retry and Ignore buttons.
3	vbYesNoCancel	Yes, No and Cancel buttons
4	vbYesNo	Yes and No buttons
5	vbRetryCancel	Retry and Cancel buttons

Title: It Displays the Title for the message box

Example –

```
yourMsg=MsgBox( "Click OK to Proceed", 1, "Startup Menu")
OR
```

yourMsg=Msg("Click OK to Proceed", vbOkCancel,"Startup Menu") yourMsg is a variable that holds values that are returned by the MsgBox ( ) function.

Dim testmsg As Integer testmsg = MsgBox("Click to test", 1, "Test message")

Example MessageBox and InputBox:

```
Sub InputDemo()
    Dim varUserInput As Variant
    varUserInput = InputBox("Enter something in this Input Box :", _ "Your Title", "Use a
    Default Entry if You Want")
    If varUserInput <> " " Then MsgBox varUserInput
End Sub
```

Variables and Data Types:

What are Variables?

1. Variables are used to temporarily store information during course of the program execution.
2. Each variable has a specific type, which indicates how much memory the data requires and the operations that can be performed on that kind of data.

Variable Declaration Syntax:

Dim <Variable Name> As <Datatype>

Dim – Stands for Dimension used to declare the variable and it allocates the memory of the variable.

Variable Name – Name assigned to the variable

As – Keyword

Datatype – Indicates what type of value must be stored in the variable

What is a Data Type?

It indicates that what type of value will be stored in the variable.

VBA's Built-in Data Types

Data Type	***Bytes***	***Used Range of Values***
Boolean	2	True or False
Integer	2	–32,768 to 32,767
Long	4	–2,147,483,648 to 2,147,483,647
Single	4	–3.402823E38 to 1.401298E45
Double (negative)	8	–1.79769313486232E308 to 4.94065645841247E-324
Double (positive)	8	4.94065645841247E–324 to 1.79769313486232E308

Currency	8	–922,337,203,685,477.5808 to 922,337,203,685,477.5807
Date	8	1/1/100 to 12/31/9999
String	1	per char Varies
Object	4	Any defined object
Variant	Varies	Any data type
User defined	Varies	Varies

Variable Naming Convention:

1. Must begin with a letter
2. After starting with a letter, can be made of letters, underscores, and digits in another order
3. Cannot have a period
4. Can have up to 255 characters.
5. Must be unique inside of the event (or procedure, function or module (we will learn what these things are)) it is used in.
6. Variable name must be prefixed with variable‟s data type name. The following table provides the list of prefixes

Data Type	Prefix	Example
Boolean	bln	blnFound
Byte	byt	bytTracks
Date/Time	dtm	dteStartOfShift
Double	dbl	dblDistance
Error	err	errCantOpen
Integer	int	intNbrOfStudents
Long	lng	lngPopulation
Object	obj	objConnection
Single	sng	sngAge
String	str	strCountryName
Currency	cur	curHourlySalary
Variant	var	varFullName

Example –

```
Dim intEmpNo As Integer
Dim strEmpName As String
```

Assigning Values to the Variable:

Assignment operator is used to place the value in the variable. It stores only one value.

Syntax:

<Variable Name> = <Value>

Example –

intEmpNo = 2004 strEmpName = “Sanjay”

Note: In VBA variable declaration is optional. Add **Option Explicit** Statement to make the variable declaration mandatory.

Scope and Visibility of the Variable:

Variable can be declared at the

1. **Procedure Level (Private)**

   Procedure Level – The variable declared at this level can be used only within the Procedure.

2. **Module Level (Public)**

   Module Level – Variables at this level can be accessed by all the procedures Public variables can be accessed outside the procedures.