URL: https://analysistabs.com/vba-interview-questions-answers/excel/

Excel VBA Oral Interview Questions

**What is the difference between ActiveWorkbook and ThisWorkbook?**

Solution:

ThisWorkbook refers to the workbook where the code is being written while ActiveWorkbook refers to the workbook which is in the active state with active window. In the case of only one workbook open, ActiveWorkbook is same as ThisWorkbook.

**What is the basic object model of Excel?**

Solution:

Please find the below basic Object model of Excel.  
Application –> Workbooks –> Worksheets –> Range / Chart

**Why we need to use macros?**

Solution:

A macro is nothing but set of instructions which are stored in Visual Basic module in a VBA Editor. It helps in automating common repetitive tasks on daily, weekly or monthly basis by running macro. Using macros, you can save lot of time, increase productivity and on time delivery to customers.

**What is the shortcut to go to VBA editor screen?**

Solution:

We can use ‘Alt+F11’ key to go to VBA editor screen.

How do I stop recording macro?

Solution:

Please find the following steps to stop recording macro in the workbook.  
Step 1: Go To Developer tab from the main ribbon of Excel window.  
Step 2: Click on ‘Stop Recording’ command button to stop from the recording macro.

How do I delete macros from the workbook?

Solution:

Please find the following steps to delete macros from the workbook.  
Step 1: Go To Developer tab from the main ribbon of Excel window.  
Step 2: Click on the Macros command button to see the available macros in the active workbook.  
Step 3: Once you click on the Macros command button, Macro dialog box will appear on the screen.  
Step 4: Select macro name which you want to delete macro and then click on ‘Delete’ command button.  
Step 5: Now, It will show the confirmation dialog box. Click on Ok to delete the macro.

**How to run macros automatically while opening workbook in Excel VBA?**

Solution:

To get Workbook\_Open() Event in Excel, please find the following steps.  
1. Go To VBA Editior.  
2. Click on ‘ThisWorkbook’ from the Project Explorer.  
3. Now, you can see two drop down lists in the right side.  
4. Select ‘Workbook’ from the first drop down list and then choose ‘Open’ from the second drop down list.  
5. Now, you can see the following code.

Private Sub Workbook\_Open()

‘Your Statements…..

End Sub

6. You can add the code in-between the above lines to run a macro.  
7. Save and close the workbook  
8. Now, reopen the workbook to test the macro.  
Example:

Private Sub Workbook\_Open()

MsgBox “Workbook has Opened Successfully.”, vbInformation

End Sub

In the above example, the macro will run automatically when we are opening workbook. Now, it will display message like “Workbook has Opened Successfully.”.

Or we can also define a procedure named Auto\_Open() in any code module, this will execute while opening the macro file.

**What is the difference between ByVal and ByRef?**

Solution:

ByVal vs ByRef in VBA is also one of the most frequently asked Excel VBA Interview Questions and Answers.

**ByVal**:  
Specifies that an argument is passed in such a way that the called procedure or property cannot change the value of a variable underlying the argument in the calling code.  
**ByRef**:  
Specifies that an argument is passed in such a way that the called procedure can change the value of a variable underlying the argument in the calling code.  
**Note**: Default value is ByRef. It is good practice to include the ByRef declaration if you are going to change the value of the parameter.

**What is the available looping statement?**

Solution:

Please find the different looping statements which are available in Excel VBA.  
For…. Next loop, Do While…. Loop, Do until Loop, Do….Loop Until..,Do While Not…Loop, While…. Wend loop.

**How to create object Variable for workbook, worksheet, etc?**

Solution:

We can create object variable and it can use in entire procedure. Please find the following examples to create object for workbook, worksheet, etc.  
Create object for Workbook:

Sub Create\_Object\_Workbook()

Dim Wb As Workbook

Set Wb = ActiveWorkbook

MsgBox Wb.Sheets(1).Name

End Sub

Explanation: In the above example, I have created and assigned to Active Workbook to ‘Wb’ objet. And then I have used it in the next statement(MsgBox Wb.Sheets(1).Name) to display first worksheet name.

**What is Array?**

Solution:

An array is a set of variables that are similar type. Using an array, you can refer to a specific value of an array by using array name and the index number (also called subscript).

We can create and define size of an array variable in the following way.  
Dim ArrayName (IndexNumber) As Datatype  
Example: Dim aValue(2) As Integer  
In the above statement ‘aValue’ is an array name and ‘2’ indicates an array size.

**How to assign values to an array?**

Solution:

We can assign values to an array in the following way.  
‘Declare an array variable  
Dim aValue (2) As Integer  
aValue(0)=“first”  
aValue(1)= “Second”  
aValue(2)= “Third”  
‘Or  
Dim aValue () As Integer={“ first “,”Second”,”Third”}

**What is meant by Data type?**

Solution:

Data Type: A data type tells, what kind of variable we are going to use in a procedure or function. The information that specifies how much space a variable needs called a data type.  
Before using variable, we need to know how much space the variable will occupy in memory, because different variables occupy different amount of space in memory.

We can declare the variable in the following way.

Dim VariableName as Datatype

Example:  
Dim iCnt as Integer

Where iCnt represents VariableName and Integer represents Datatype.

**What are the data types in VBA?**

Solution:

Data types in VBA are classified into two categories

|  |  |
| --- | --- |
| **Numeric Data type** | **Non-numeric Data type** |
| * Byte | * String (fixed length) |
| * Integer | * String (variable length) |
| * Long | * Date |
| * Single | * Boolean |
| * Double | * Object |
| * Currency | * Variant (numeric) |
| * Decimal | * Variant (text) |

**What are the different UserForm Control and ActivX Control?**

Solution:

**UserForm Controls:**  
Button, Combo Box, Check Box, Spin Button, List Box, Option Button, Group Box, Label, Scroll Bar, etc,.  
**ActiveX Controls:**  
Command Button, Combo Box, Check Box, List Box, Text Box, Scroll Bar, Spin Button, Option Button, Label, Image, Toggle Button.

**What is the meant by scope of variables?**

Solution:

When we are working with variables, it is important to understand the Scope of a Variable. The Scope describes the accessibility or life time or visibility of a variable.  
There are four levels of Scope:  
• Procedure-Level Scope  
• Module-Level Scope  
• Project-Level Scope  
• Global-Level Scope

Procedure-Level Scope:

Also called as Local Variables, all Procedure-Level variables are accessible only within the procedure or Function in which they are declared. As soon as the procedure finishes, the variable lost its scope.

In the following example, iCntr is a Local Variable which can be only accessible in this procedure.

Sub sbScopeProcedureLevel()

Dim iCntr As Integer

iCntr = 2000

MsgBox "Example of a Procedure level Variable: " & iCntr

End Sub

### Module-Level Scope:

All Procedure-Level variables are accessible only within the Module in which they are declared. These are variables that are declared outside the Procedure itself at the very top of any Module. Its value is retained unless the Workbook closes or an End Statement is used.

In the following example, lRow can be accessible any procedure in the Module in which it is declared.

Option Explicit

'Module Level Variables

Dim lRow As Long

Sub sbProcedure1()

MsgBox "Example of a Module Level Variable " & lRow

End Sub

Sub sbProcedure2()

MsgBox "Example of a Module Level Variable " & lRow

End Sub

### Global-Level Scope:

All Global-Level variables are accessible in anywhere in the Project (.i.e; in any Module, User Form, Classes) within the Workbook in which they are declared. And also accessible to outside of this project or workbook. These are variables that are declared using ‘Public’ keyword at the very top of any Public Module .

In the following example, lRow can be accessible any procedure in the project or workbook and also out-side of the module.

'Code in the Module 1:

Option Explicit

'Module Level Variables

Public lRow As Long

Sub sbProcedure1()

lRow = 220

MsgBox "Example of a Public Level Variable " & lRow

End Sub

'Code in the Module 2:

Sub sbProcedure2()

MsgBox "Example of a Public Level Variable " & lRow

End Sub

### Project-Level Scope:

### We set Project -Level Scope to the variables if we want to make the public variable to be accessed only in the project in which they are declared and not out side of this project. To set this option we need to add “Option Private Module” statement at the top of the declaration area.

### In the following example, lRow can be accessible any procedure in the project or workbook only in which it is declared.

'Code in the Module 1:

Option Explicit

Option Private Module

‘Module Level Variables

Public lRow As Long

Sub sbProcedure1()

lRow = 220

MsgBox “Example of a Public Level Variable ” & lRow

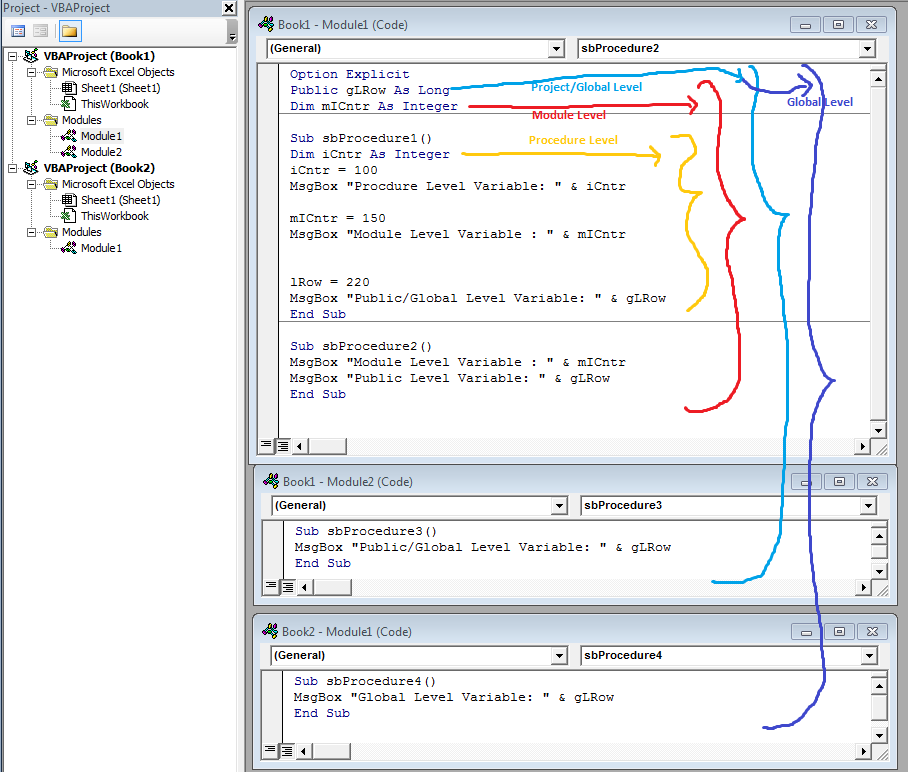
End Sub

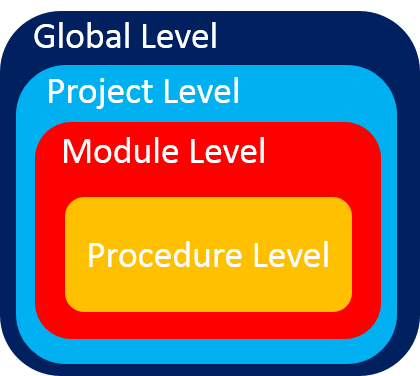
'Code in the Module 2:

Sub sbProcedure2()

MsgBox “Example of a Public Level Variable ” & lRow

End Sub





**How to find last row in the worksheet?**

Solution:

We need to find Last used Row with data if we want to perform certain task on each row of worksheet. Please find the following statements to find last row in the worksheet.

Find last row in the worksheet:

Dim lastRow As Long

lastRow = ActiveSheet.Cells.SpecialCells(xlLastCell).Row

**How to find last column in the worksheet?**

Solution:

We need to find Last used Column with data if we want to perform certain task on each column in the worksheet. Please find the following statements to find last column in the worksheet.

Find last column in the worksheet:

Dim lastColumn As Long

lastColumn = ActiveSheet.Cells.SpecialCells(xlLastCell).Column

**How to fasten the VBA macro?**

Solution:

We can fasten the execution of macros or VBA Procedures by following the below tips.  
1. Declare the variables and avoid using ‘Variant’ Data Type.  
2. Turn Off Screen Updating  
3. Turn Off Automatic Calculations  
4. Disable Events  
5. Use With Statement  
6. Use vbNullString instead of “”.  
7. Release memory objects at the end of the procedure.

**How to enable/disable screen updating? Or how do hide the process of running macro?**

Solution:

Here is the approach to enable or disable screen updating or screen flickering.  
In order to stop the screen flickering, stop the screen updating at Staring of the procedure:

Application.ScreenUpdating = False

You have to set back screen updating as True Before ending of the procedure:

Application.ScreenUpdating = True

**How to stop trigger or display alert or error warning while running macros?**  
Solution:

Here is the approach to stop trigger or display alerts or error warnings.  
In order to stop triggers, disable the display alerts at Staring of the procedure:

Application.DisplayAlerts = False

You have to set enable display alerts before ending of the procedure:

Application.DisplayAlerts = True

**What is the different types of error handling techniques?**

Solution:

Please find the below different types of error handling techniques.  
1. On Error Resume Next  
2. On Error Goto Err\_Lbl  
3. On Error Goto 0

**How to check whether file exists or not in a specified location?**

Solution:

You can find a specific file exist or not in the following two ways.  
**1. Using FileSystemObject:**  
Here is the example to check file exist or not using ‘FileSystemObject’.

Sub Check\_File2()

Dim FSO

Dim sFileName As String

sFileName = "C:/Test/Workbook.xls"

Set FSO = CreateObject("Scripting.FileSystemObject")

If Not FSO.FileExists(sFileName) Then

MsgBox "File Does Not Exists."

Else

MsgBox "File Exists."

End If

End Sub

**2. Using Dir Function:**  
Here is the example to check file exist or not using ‘Dir’ function.

Sub Check\_File1()

Dim sFileName As String

sFileName = "C:/Test/Workbook.xls"

If Dir(sFileName) <> "" Then

MsgBox "File Exists."

Else

MsgBox "File Does Not Exists."

End If

End Sub

**How to save workbook using Excel VBA?**

Solution:

You can save the workbook using following example. In the below example we are adding new workbook and then assigned it to object named Wkb. Finally we are saving workbook with using Save method of workbook object.

Sub Save\_Workbook()

Dim Wkb As Workbook

Set Wkb = Workbooks.Add

Wkb.Save

End Sub

**How to change the existing file name?**

Solution:

You can change the existing file name of workbook using the following example. In the below example we are adding new workbook and then assigned it to object named Wkb. Finally we are changing workbook name with using SaveAs method of workbook object.

Sub SaveAs\_Workbook()

Dim Wkb As Workbook

Set Wkb = Workbooks.Add

ActiveWorkbook.SaveAs Filename:="C:\Test.xlsm"

End Sub

**How to delete a file in a specific location?**

Solution:

Please find the following statements to delete a file from the specified location.

Sub sbDelete\_File()

Dim FSO

Dim sFile As String

sFile = "C:\Test.xlsm"

'Set Object

Set FSO = CreateObject("Scripting.FileSystemObject")

'Check File Exists or Not

If FSO.FileExists(sFile) Then

FSO.DeleteFile sFile, True

MsgBox "Deleted The File Successfully", vbInformation, "Done!"

Else

MsgBox "Specified File Not Found", vbInformation, "Not Found!"

End If

End Sub

Note: Before deleting file from the specified location, we have to check whether file is exists or not in the specified location. In the above example we are using statement:’ FSO.FileExists(sFile)’ to check for the file.  
Please find the following link for more information on deleting a file in a specified location

**How to copy a file from one location to another location?**

Solution:

We can use CopyFile method of FileSystemObject in Excel VBA. First we will check whether file is existing in a location. Then we can copy the file if it exists to a required location.

'In this Example I am Copying the File From "C:Temp" Folder to "D:Job" Folder

Sub sbCopyingAFile()

'Declare Variables

Dim FSO

Dim sFile As String

Dim sSFolder As String

Dim sDFolder As String

'This is Your File Name which you want to Copy

sFile = "Sample.xls"

'Change to match the source folder path

sSFolder = "C:\Temp\"

'Change to match the destination folder path

sDFolder = "D:\Job\"

'Create Object

Set FSO = CreateObject("Scripting.FileSystemObject")

'Checking If File Is Located in the Source Folder

If Not FSO.FileExists(sSFolder & sFile) Then

MsgBox "Specified File Not Found", vbInformation, "Not Found"

'Copying If the Same File is Not Located in the Destination Folder

ElseIf Not FSO.FileExists(sDFolder & sFile) Then

FSO.CopyFile (sSFolder & sFile), sDFolder, True

MsgBox "Specified File Copied Successfully", vbInformation, "Done!"

Else

MsgBox "Specified File Already Exists In The Destination Folder", vbExclamation, "File Already Exists"

End If

End Sub

**How to move a file from one location to another location?**

Solution:

You can move a file from one location to another location in the following way.

Sub Move\_File()

Dim sFileName As String

Dim dFileName As String

sFileName = "D:\Test.xlsx" 'Source File Location Name

dFileName = "E:\Test.xlsx" 'Destination File Location Name

Name sFileName As sFileName

End Sub

**How to delete rows in the worksheet?**

Solution:

Here is the example to delete entire row(fifth row) in the active worksheet.

Sub Delete\_EntireRow()

Rows(5).EntireRow.Delete ‘ 5th row

‘or

Rows(“1:5”).EntireRow.Delete ‘ first 5 rows

End Sub

### Delete rows based on Date using VBA: Examples

The following VBA code is to delete rows based on Date from the excel worksheet. This code will delete the rows (1 to 20) if it satisfy the condition if cell value is equals to today’s date.

Sub sbDelete\_Rows\_Based\_On\_Date()

Dim lRow As Long

Dim iCntr As Long

lRow = 20

For iCntr = lRow To 1 Step -1

If Format(Cells(iCntr, 1), &quot;dd-mm-yyyy”) = Format(Now(), &quot;dd-mm-yyyy&quot;) Then

Rows(iCntr).Delete

End If

Next

End Sub

#### Explained VBA Code to Delete Rows based on Date

‘Strating program and sub procedure to write VBA code to delete rows based on Date  
Sub sbDelete\_Rows\_Based\_On\_Date()

‘Declaring the variable lRow as long to store the last row number  
Dim lRow As Long

‘Declaring the variable iCntr as long to use in the For loop  
Dim iCntr As Long

‘Assigning the last row value to the variable lRow  
lRow = 20

‘Using for loop  
‘We are checking the each cell value if the cell values is equals to today’s date  
‘And deleting the row if true  
For iCntr = lRow To 1 Step -1  
If Format(Cells(iCntr, 1), “dd-mm-yyyy”) = Format(Now(), “dd-mm-yyyy”) Then  
Rows(iCntr).Delete  
End If  
Next

‘Ending the macro to delete the rows based on Date using VBA  
End Sub

**How to delete columns in the worksheet?**

Solution:

Here is the example to delete entire column(s) in the active worksheet.

Sub Delete\_EntireRow()

Columna(“E”).EntireColumn.Delete ‘ 5th column

Columna(“A:E”).EntireColumn.Delete ‘ first 5 columns

End Sub

Delete Columns with specific data using VBA: Examples

The following VBA code is to delete Columns with specific data from the excel worksheet. This code will delete the Columns (1 to 20) if cell value is “Your Data”.

Sub sbDelete\_Columns\_With\_Specific\_Data ()

Dim lColumn As Long

Dim iCntr As Long

lColumn = 20

For iCntr = lColumn To 1 Step -1

If Cells(1, iCntr) = “Your Data” Then ‘ You can change this text

‘If Cells(1, iCntr) = “22-12-2013” Then ‘ to check specific date

Columns(iCntr).Delete

End If

Next

End Sub

#### Explained VBA Code to Delete Columns with specific data

Starting program and sub procedure to write VBA code to delete Columns with specific data.

Sub sbDelete\_Columns\_With\_Specific\_Data\_C()

‘Declaring the variable lColumn as long to store the last Column number  
Dim lColumn As Long

‘Declaring the variable iCntr as long to use in the For loop  
Dim iCntr As Long

‘Assigning the last Column value to the variable lColumn  
lColumn = 20

‘Using for loop  
‘We are checking the each cell value if it cell value equals “Your String”  
‘And deleting the Column if true  
For iCntr = lColumn To 1 Step -1  
If Cells(1, iCntr) = “Your Data” Then  
Columns(iCntr).Delete  
End If  
Next

End Sub

**How to change the cell color?**

Solution:

Please find the following example to change the cell(3rd row, 2nd column) color.

Sub Cell\_Color()

'Using Cell Object

Cells(3, 2).Interior.ColorIndex = 5 ' 5 indicates Blue Color

End Sub

### Change Background Color of Cell Range in Excel VBA – Examples

###### Example 1

In this Example below I am changing the Range B3 Background Color using Cell Object

Sub sbRangeFillColorExample1()

'Using Cell Object

Cells(3, 2).Interior.ColorIndex = 5 ' 5 indicates Blue Color

End Sub

###### Example 2

In this Example below I am changing the Range B3 Background Color using Range Object

Sub sbRangeFillColorExample2()

'Using Range Object

Range("B3").Interior.ColorIndex = 5

End Sub

###### Example 3

We can also use RGB color format, instead of ColorIndex. See the following example:

Sub sbRangeFillColorExample3()

'Using Cell Object

Cells(3, 2).Interior.Color = RGB(0, 0, 250)

'Using Range Object

Range("B3").Interior.Color = RGB(0, 0, 250)

End Sub

###### Example 4

The following example will apply all the colorIndex form 1 to 55 in Activesheet.

Sub sbPrintColorIndexColors()

Dim iCntr

For iCntr = 1 To 56

Cells(iCntr, 1).Interior.ColorIndex = iCntr

Cells(iCntr, 1) = iCntr

Next iCntr

End Sub

**How to change the range color?**

Solution:

Please find the following example to change the range(A1 toB5) color.

Sub Cell\_Color()

'Using Range Object

Range("A1:B5").Interior.ColorIndex = 5 ' 5 indicates Blue Color

End Sub

**How to hide/unhide a rows?**

Solution:

Please find the following example, it will take you through how to hide or unhide single or multiple row(s).  
**Hide Row(S):**

Sub Hide\_Row()

Rows("5").EntireRow.Hidden = True 'Hides 5th Row(Single Row)

Rows("1:5").EntireRow.Hidden = True 'Hides 1 to 5 Rows(Multiple Rows)

End Sub

**UnHide\_Row(S):**

Sub UnHide\_Row()

Rows("5").EntireRow.Hidden = False 'UnHide 5th Row(Single Row)

Rows("1:5").EntireRow.Hidden = False 'UnHides 1 to 5 Rows(Multiple Rows)

End Sub

##### Hide-UnHide Rows in Excel Worksheet using VBA – An Example

The following example will show you how to Hide and Unhide the rows in excel worksheet using VBA. We can Hide or Unhide the multiple rows at a time. In this example I am hiding and Unhiding Rows 5 to 8.

###### Code:

Sub sbHidingUnHideRows()

'To Hide Rows 22 to 25

Rows("5:8").EntireRow.Hidden = True

'To UnHide Rows 22 to 25

Rows("5:8").EntireRow.Hidden = False

End Sub

**How to hide/unhide a columns?**

Solution:

Please find the following example, it will take you through how to hide or unhide single or multiple column(s).

Sub sbHidingUnHideColumns()

'To Hide Columns B to D

Columns("B:D").EntireColumn.Hidden = True

'To UnHide Columns B to D

Columns("B:D").EntireColumn.Hidden = False

End Sub

**How to hide/unhide worksheet?**

Solution:

Please find the following examples to hide or unhide worksheets.  
Hide Sheet:

Sub Hide\_Sheet()

Sheets("Sheet1").Visible = True

'or

Sheets("Sheet1").Visible = 2 'to very hide the worksheet

End Sub

UnHide Sheet:

Sub UnHide\_Sheet()

Sheets("Sheet1").Visible = False

End Sub

**How do I open workbook?**

Solution:

Here is the example to open Workbook named: “C:\WorkbookName1.xls”

Sub Open\_Workbook()

Workbooks.Open " C:\WorkbookName1.xls"

'OR

Workbooks.Open Filename:="C:\WorkbookName1.xls"

End Sub

Sub Set\_Open\_ExistingWorkbook()

Dim wkb As Workbook

Set wkb = Workbooks.Open("C:\WorkbookName.xls")

'OR

Set wkb = Workbooks.Open(Filename:="C:\WorkbookName1.xls")

End Sub

**How do I close workbook?**

Solution:

Here is the example to close opened Workbook. Here Workbook name is: “D:\Test.xlsx”

Sub ExampleCloseWorkbook()

'Procedure to close an Excel worbook

'Close is an workbook method to close a workbook

ActiveWorkbook.Close

End Sub

##### Close an Opened Workbook using object

Sub ExampleOpenAnExistingWorkbookSet()

'Declaration - declare an object for setting the workbook

Dim wkb As Workbook

'Open Workbook and set an object(wkb)

Set wkb = Workbooks.Open(&quot;C:\WorkbookName.xls&quot;)

'Close opened workbook using object

wkb.Close SaveChanges:=True

End Sub

**What is the difference between VB6, VBA and VB Script?**

Solution:

All these three programming languages are developed based on ‘BASIC’ programming language.  
**VB6**: VB6 is a Visual programming language developed to help the users to easily develop the Windows Applications.  
**VBA**: VBA is Visual Basic Programming for developing MS Office Applications. It’s available in all MS Office Tools, such as MS Excel, MS Access, MS PowerPoint and MS Word.  
**VB Script:** Visual Basic Programming for Web Browser (Microsoft Internet Explorer). You can write the VBScript in html pages to program the different objects in Web Pages.

**How to protect or unprotect worksheet?**

Solution:

Please find the following examples to protect or UnProtect worksheet.  
**Protect Sheet:**

Sub Protect\_Sheet()

Sheets("Sheet1").Protect "YourPassword", True, True

End Sub

In the above example we are protecting the worksheet by using ‘Protect’ method of Worksheet object.  
**UnProtect Sheet:**

Sub UnProtect\_Sheet()

Sheets("Sheet1").UnProtect "YourPassword”

End Sub

In the above example we are unprotecting the worksheet by using ‘UnProtect’ method of Worksheet object.

### Protect All Worksheets in Workbook using VBA – Exmaple:

Sub sbProtectAllSheets()

Dim pwd1 As String, pwd2 As String

pwd1 = InputBox("Please Enter the password")

If pwd1 = "" Then Exit Sub

pwd2 = InputBox("Please re-enter the password")

If pwd2 = "" Then Exit Sub

'Check if both the passwords are identical

If InStr(1, pwd2, pwd1, 0) = 0 Or \_

InStr(1, pwd1, pwd2, 0) = 0 Then

MsgBox "You entered different passwords. No action taken"

Exit Sub

End If

For Each ws In Worksheets

ws.Protect Password:=pwd1

Next

MsgBox "All sheets Protected."

Exit Sub

End Sub

### UnProtect All Worksheets in Workbook using VBA – Example:

Sub sbUnProtectAll()

On Error GoTo ErrorOccured

Dim pwd1 As String

pwd1 = InputBox("Please Enter the password")

If pwd1 = "" Then Exit Sub

For Each ws In Worksheets

ws.Unprotect Password:=pwd1

Next

MsgBox "All sheets UnProtected."

Exit Sub

ErrorOccured:

MsgBox "Sheets could not be UnProtected - Password Incorrect"

Exit Sub

End Sub

**How to find last used row in a worksheet?**

Solution:

Here is the example to find last row in a worksheet.

Sub Find\_LastRow\_In\_Worksheet()

Dim lastRow As Long

lastRow = Sheets("Sheet1").Cells.SpecialCells(xlLastCell).Row

End Sub

**How to find last used column in a worksheet?**

Solution:

Here is the example to find last column in a worksheet.

Sub Find\_LastColumn\_In\_Worksheet()

Dim lastCol As Long

lastCol = Sheets("Sheet1").Cells.SpecialCells(xlLastCell).Column

End Sub

**What is the difference between Function and Subroutine?**

Solution:

A set of statements to perform a specific task which is placed in a procedure called Function or Subroutine.  
Here is the difference between Function and Subroutine.  
**Function:**  
Function will perform calculations based on specified conditions and then return a single value as a result. You can use function anywhere in the procedure and you can use number of times based on the requirement. You can use ‘Exit Function’ or ‘Return’ from immediate exit from a function procedure. Please find the following function statements.

Function Function\_Name()

Statements…

End Function

**Subroutine:**  
Subroutine cannot return any value and it will perform set of actions. We can declare any number of procedures or functions in a module. Procedures or functions can be called in another procedure procedure. You can use ‘Exit Sub’ from immediate exit from a Subroutine or procedure. Please find the following subroutine statements.

Sub Procedure\_Name()

Statements…

End Sub

Use function procedure to return any value, Otherwise use a Sub procedure.

**How to debug a VBA code?**

Solution:

Here is the example to find last row in a worksheet.

Sub Find\_LastRow\_In\_Worksheet()

Dim lastRow As Long

lastRow = Sheets("Sheet1").Cells.SpecialCells(xlLastCell).Row

End Sub

**What is the method for returning more than one values from a function in VB?**

Solution:

Any of the three methods can be used:  
i) Create a class with the properties you require to return and then return the object of the class from the function.  
ii) Using ByRef for the values.  
iii) Return an array of the values

**Does VBA support oops concepts?**

Solution:

VBA is Objected Based Programming Language, not Object oriented language. It will not support all ruled of OOPs concepts, like Inheritance, Multithreading.

**What is the difference between used range and current range?**

Solution:

Current range will be the range used surrounded by a range. And the range which you have used in your entire worksheet is called Used Range.  
For Example, you might have entered some data at Range A1:D10 and then Range A20:D30.  
Current Range of A1 (or any range in A1:D10) will be Range A1:D10. Similarlly current range of A20 .. or D30 will be A20:D30.  
And the used range of the worksheet will be A1:D30.

Sub CurrentRange()

MsgBox Range("A1").CurrentRegion.Address

End Sub

One worksheet can have one or more current regions, and only one used range. So, used range will be collection of current regions.

Current region will be used to change the formats or calculations to particular block of range (region). And used range will be used to deal with the entire used range in the worksheet.  
Current region will connected with at least on data cell (non empty cell). And the used ranges will consider if any range or cell is used once and even if the cell is blank.  
You can used the below procedure to find the used range in a worksheet.

Sub UsedRange()

MsgBox ActiveSheet.UsedRange.Address

End Sub

As explained above, once you use any range it will be considered as used cell even after deleting the content in the cell. To get the used range with the data, please refer our examples on finding actual used range in the worksheet.