**Excel VBA**

**INTRODUCTION**

* VB – standalone program that runs independently
* VBA – part of Excel Program and cannot work alone
* VBS – variant of Visual Basic Language used for Internet Applications
* Files need to be open in order to see them in VBA
* Opening VBA: Developer 🡪 Visual Basics

**RECORDING MACRO**

* You have to select ‘Use Relative Referrences’
* Then you can Record Macro
* When recording macro, we can set a shortcut, i.e. Ctrl + Shift + A
  + Ctrl + A -Z is already taken by Excel
* We can store it in This Workbook
* When you click OK the recording starts
* When starting the macro, we need to select the cell that we selected while recording
* When selecting Macro and clicking Edit, we can see the code in VBA
* You should record just simple things
* If we want to save notebook with macros we need to save it as Excel Macro-Enabled Workbook when saving the file
  + The extension will be .xlsm
  + The file with macros will have an explanation mark in the icon

**WRITING SIMPLE MACRO**

* First we need to create a Module: Insert 🡪 Module
* Every macro must be written in Sub
  + Sub first\_Macro()
  + End Sub
* MsgBox “TutorialPoint” to create a message box
  + Or MsgBox 100 to show a number
* To execute the code, keep the cursor in between Sub and End Sub and click Run 🡪 Run Sub
  + Shortcut f5

**CELL REFERENCING**

* If we want to write value to a cell we have active, meaning that the cursor is in that cell in Excel
  + Activecell.Value = “tutorial”
  + If we write it twice with different values, the last one will be accepted
* To see what is happening step by step, we can use Step Into
  + Debug 🡪 Step Into (shortcut f8)
* If we want to write a value to a cell that is not selected:
  + [b5] will use that cell
    - i.e. [b5].Value = 70
  + [c1:c10] will select the range
* Another way of selecting a cell:
  + Cells(row index, column index)
    - i.e. Cells(8,2) will select the cell b8
  + Used for if conditions, and loop and for
* the most common way to select a cell is with Range
  + Range(“a1”).Value = “India”
  + Range(“a2:a10”) to select a range

**COPY PASTE**

* Range(“a1:a10”).Value = “India”
* 1st method:
  + Range(“b1:b10”) = Range(“a1:a10”).Value
  + This will copy the values
* 2nd method:
  + Range(“a1:a10”).Copy
  + Range(“b1:b10”).PasteSpecial
  + Application.CutCopyMode = False
    - This will make sure that the copy part is not selected anymore

**FONT**

* Range(“a1:a10”).Value = “”Tutorial
* Range(“a1:a10”).font.Name = “Arial”
* Range(“a1:a10”).font.Bold = True
* Range(“a1:a10”).font.Size = 20
* Range(“a1:a10”).font.Italic = True
* Range(“a1:a10”).font.Underline = True
* Range(“a1:a10”).font.Strikethrough = True

**WITH BLOCK**

* It reduces the time so we don’t have to write the same things over again
* With Range(“a1:a10”).font
  + .Name = “Arial”
  + . Size = 20
* End With

**BORDERS**

* Range("a1:a10").borders.LineStyle = xlDot
  + Other styles:
    - xlDot
    - xlDash
    - xlContinuous
    - xlDouble
    - xlNone
* Range("a1:a10").borders.Color = vbGreen
* Range("a1:a10").borders.Weights = 3

**ALIGNMENT**

* Range("a1:a10").HorizontalAlignment = xlLeft
  + xlRight
  + xlCenter
* Range("a1:a10").VerticalAlignment = xlTop
  + xlBottom
  + xlCenter

**FONT COLOR**

* Range("a1:b5").font.Color = vbGreen
  + There are only 8 colors that we can use with this command
  + vbGreen, vbBlack, vbWhite, vbYello, vbRed, vbBlue, vbCyan, vbMagneta
* Range("a1:b5").font.ColorIndex = 1
  + Indexes fomr 1 to 56 to change the color

**BACKGROUND COLOR**

* The colors and indexes are the same as in font color
* Range("a1:b5").Interior.Color = vbRed
* Range("a1:b5").Interior.ColorIndex = 42

**PASTE SPECIAL**

* Used, for example, when we want to copy the format
* Works with a space instead of a dot
* Range("a1:a5").Copy
* Range("b1:b5").pastespecial xlPasteFormats
  + xlPasteColumnWidth
  + xlPasteValues
* Application.CutCopyMode = False

**ORIENTATION**

* Tilting the text to some angle
* We just write the degrees we want
* Range(“a1”).orientation = 20
  + Angle 20 degrees

**WRAP TEXT**

* The column width stays the same but the row height will adjust to the text
* Range(“a1:a10”).WrapText = True

**MERGE UNMERGE**

* Range(“a1:d1”).Merge
* Range(“a1:d1”).UnMerge

**CLEAR CELLS**

* If we want to clear the cells
* Range(“a1:a10”).ClearFormats
  + .ClearComments
  + .ClearHyperlinks
  + .Clear
    - Clears everything, even the text

**DELETE CELLS**

* The cell will disappear completely, not just clear the content
* Range(“b3”).Delete
  + The cell will be deleted and remaining cells will be shifted up
* Range(“a1:a10”).Delete
  + The range of cells will be deleted and the remaining cells will be shifted to the left
* Range(“a1”).EntireRow.Delete
* Range(“a1”).EntireColumn.Delete

**ROWS & COLUMN INSERT**

* Two options
* First option:
  + Range(“c:c”).Insert
    - Insert a column C
  + Range(“1:1”).Insert
    - Inserts a row 1
* Second option:
  + Range(“b5”).EntireColumn.Insert
    - Inserts a column B
  + Range(“b5”).EntireRow.Insert
    - Inserts a row 5

**ROWS & COLUMNS DELETE**

* Range(“a1”).EntireRow.Delete
* Range(“a1”).EntireColumn.Delete
* Range(“a1:a3”).EntireRow.Delete
  + Deletes 3 rows

**COLUMN WIDTH**

* We have two options:
  + Range(“a1”).ColumnWidth = 15
  + Range(“a1”).Columns.ColumnWidth = 15
* We can also autofit the width
  + Range(“a1”).Columns.AutoFit

**ROW HEIGHT**

* We have two options
  + Range(“a1”).RowHeight = 10
  + Range(“a1”).Rows.RowHeight = 10
* We can also autofit the height
* Range(“a1”).Rows.AutoFit

**ACTIVAE & SELECT**

* Both active and select work the same way and just select the cell or the range of cells
* Range(“a1”).Select
* Range(“a1”).Activate

**COLUMNS HIDE & UNHIDE**

* Range(“a1”).Columns.Hidden = True
* Range(“a1”).Columns.Hidden = False
* Range(“a:b”).Columns.Hidden = True

**ROW HIDE & UNHIDE**

* Range(“a1”).Rows.Hidden = False
* Range(“a1”).Rows.Hidden = False
* Range(“1:3”).Rows.Hidden = True

**SHEETS INTRODUCTION – SHEET REFERENCING**

* Referencing cells in the other sheet
* Sheets(1). Range(“a1”).Value = “test”
  + This will apply the value to the first sheet
* Sheets(“TutorialPoint”). Range(“a1”).Value = “test”
  + We can also use a name of that sheet

**ADD SHEETS**

* Two simple methods
  + Sheets.Add
  + Worksheets.Add
  + These work the same and will be added before whichever sheet is selected
* If we want to choose specific location
  + Sheets.Add after:=Sheets(“TutorialsPoint”)
  + Sheets.Add before:=Sheets(“TutorialPoint”)

**ADD SHEETS WITH NAME**

* Sheets.Add.Name = “test2”
* If we want to create a sheet with a name
  + Sheets.Add after:=Sheets("Hidden")
  + ActiveSheet.Name = "added-sheet"

**RENAME SHEETS**

* Two methods
  + Sheets(1).Name = “new\_name”
  + Sheets(“old\_name”).Name = “new\_name”

**GET SHEET NAMES**

* Showing names of the sheets and displaying them in a message box
* MsgBox(Sheets(1).Name)
* MagBox(Sheets(2).Name)
* And so on

**COPY SHEETS**

* Sheets(“Hello”).Copy after:=”Sheets(“Properties”)
* The copied sheets will have (2) after the name

**MOVE SHEETS**

* Sheets(“Hello”).Move before:=Sheets(“Properties”)

**CHANGE SHEET TAB COLOR**

* Sheets(“Details”).Tab.Color = vbBlack
  + vbRed
  + vbBlue
  + vbCyan
  + vbMagneta
* Sheets(“Details”).Tab.ColorIndex = 20
* Sheets(“Details”).Tab.Color = False

**HIDE AND UNHIDE SHEETS**

* Sheets(“Details”).Visible = True
* Sheets(“Details”).Visible = False

**SHEETS PROTECTION**

* Giving password to the sheet
* Sheets(“Details”).Protect Password:=123
* Sheets(“Details”).Unprotect Password:=123
  + After this there won’t be any password

**ACTIVATE SHEET**

* There are two options that do the same
  + Sheets(“Properties”).Activate
  + Sheets(“Properties”).Select

**CREATE WORKBOOK**

* Workbooks.Add
* If we want to add a name:
  + Workbooks.Add.SaveAs Filename:=”E:\Demobook.xlsx”

**GET WORKBOOK NAME**

* ActiveWorkbook returns the activated workbook
* ThisWorkbook returns the workbook where the code is written
* MsgBox (ThisWorkbook.Name)
  + Shows the name of the workbook where the code is written
* MsgBox (ActiveWorkbook.Name)
  + Shows the name of the active workbook
  + We can first activate the workbook we want
    - Workboos (“Book1.xlsx”).Activate

**SAVE & CLOSE WORKBOOK**

* If we want to write something to a specific workbook
  + Workbooks(“Book1.xlsx”).Sheets(1).Range(“a1:a10”).Value = “Excel”
* Saving
  + Workbooks(“Book1.xlsx”).Save
* Closing
  + Workbooks(“Book1.xlsx”).Close

**OPEN & CLOSE WORKBOOK**

* Opening
  + Workbooks.Open Filename:=”E:\Demobook.xlsx”

**DELETE WORKBOOK**

* Kill (“E:\Demobook.xlsx”)

**CREATE FOLDER**

* MkDir (“E:\Folder1”)

**VARIABLE USAGE**

* They hold certain values in the memory of the computer
* Var1 = “Tutorial”
* Range(“a1”).Value = var1
  + It returns Tutorial
* Range(“a1”).Value = var1 & var1
  + Returns TutorialTutorial

**COMMENTS**

* ‘this is a comment
* Two methods:
  + We have to use “’” to create a comment
    - ‘this is a comment
  + Or we can write “Rem” to start a comment
    - Rem this is a comment
* We can use comment blocks if we want to comment multiple lines
  + Edit 🡪 Comment Block

**FOR LOOP**

* If we want to repeat a statement multiple-times
* Based on number condition
* Example1:
  + Dim x As Integer ‘This will declare a variable x as an integer
    - For x = 1 To 10 ‘This will set the number condition
      * MsgBox 25 ‘This is what we want to repeat
    - Next ‘This ends the For loop
* We can go to Debug 🡪 Add Watch and set Expression as x (since we declared the variable x) to see how it changes with each iteration
* If we want to change the increment:
  + For x = 1 To 10 Step 2
  + By default, the step is set to 1
* If we want to iterate the cell location
  + Cells(x,1).Value = 10
    - We change the row index for x
* If we want the increments to decrease
  + We have to use step with minus “-“
  + For x = 20 To 1 Step -1

**FOR LOOP SHEET NAME**

* Dim x As Integer
  + For x = 1 To ThisWorkbook.Sheets.Count
    - MsgBox ThisWorkbook.Sheets(x).Name
  + Next

**FOR EACH NEXT LOOP**

* We use this loop when we want to do something with similar objects (i.e. sheet names, cells)
* Example:
  + Dim sht As Worksheet
    - For Each sht In ThisWorkbook.Sheets
      * MsgBox sht.Name
    - Next

**DO WHILE**

* Executes a task while the condition is true
* Example:
  + Dim i As Integer
  + i = 1
  + Do While Cells(i, 1) <> ""
  + Cells(i, 2) = Cells(i, 1) + 10
  + i = i + 1
  + Loop

**DO UNTIL**

* Executes a task until the condition becomes true
* Example:
  + Dim i As Integer
  + i = 1
  + Do Until i > 8
  + Cells(i, 3).Value = 20
  + i = i + 1
  + Loop

**TYPES OF ERRORS**

* Syntax errors (i.e. spelling)
  + Works only for one single line
* Compilation errors (i.e. missing a key word)
  + Works in a multiple-line statement
* Run time errors ()
  + cannot be found easily, needs to be tested or debugged
  + it won’t give you an error message on its own

**HERROR HANDLING**

* when we get an error, we can click on debug to highlight the part where the error occurred
* if we want the code to continue after showing us the error we can use the code:
  + On Error Resume Next
  + Example:
    - On Error Resume Next
      * MsgBox 10
      * MsgBox 10 / 0
      * MsgBox 10
    - We will get an error but the code will continue
* If we want to notify the user when there is some error, we can use GoTo and labels
  + Example:
    - On Error GoTo abc
      * MsgBox 10
      * MsgBox 10 / 0
      * MsgBox 10
    - Done:
      * Exit Sub ‘If there is no error, the code will exit
    - abc:
      * MsgBox “Error message is xyz” ‘this will be shown to the user if there is an error

**DEBUGGING**

* Step by step execution of your statements
* In the Toolbar we can find Locals Window
  + Here in Value we can see how the values of variables change
* If we want to keep a track only oh 1 variable
  + We can use Watch Window in the Toolbar
  + We select our variable and right click 🡪 add to watch
* Break point action
  + We can click on the column left to the code to insert break point
  + Then the code stops where we marked the break point
  + We can also select the line and go to Debug 🡪 Toggle Breakpoint
  + Shortcut is f9

**IMMEDIATE WINDOW**

* View 🡪 Immediate Window
* It is like a console in programming
  + It will show us the output of the code
* Example
  + ?5>3
    - Returns True
* We can also use it to do action
  + i.e. Sheet1.Name = “Tutorials”
    - after pressing “enter” it will immediately change
* if we want to ask a question, we should use the question mark
  + i.e. ?Sheet1.Name
* examples
  + ?Sheets.Count
    - Returns the number of sheets
  + ?Range(“a4”).Value
    - Returns the value