**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

**ERROR 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

**IF STATEMENT**

* Example:
  + If Range(“a1”).Value >= 5 Then Range(“b2”).Value = “Pass”
* It has only positive condition

**IF ELSE STATEMENT**

* Example:
  + If Range(“a1”).Value >= 35 Then
    - Range(“b1”).Value = “Pass”
  + Else
    - Range(“b2”).Value = “Fail”
  + End If

**IF ELSEIF ELSE STATEMENT**

* Example:
  + If Range(“a1”).Value <= 35 Then
    - Range(“b1”).Value = “Fail”
  + ElseIf Range(“a1”).Value <= 60 Then
    - Range(“b1”).Value = “C Grade”
  + Else
    - Range(“b1”).Value = “A Grade”
  + End If

**IF ELSEIF ELSE WITH AND OPERATOR**

* Example:
  + If Range(“a1”).Value >= 0 And Range(“a1”).Value < 35 Then
    - Range(“b1”).Value = “Fail”
  + ElseIf Range(“a1”).Value >= 35 And Range(“a1”).Value < 80 Then
    - Range(“b1”).Value = “B Grade”
  + Else
    - Range(“b1”).Value = “Invalid”
  + End If

**IF ELSE USING FOR LOOP**

* Used when we want to evaluate more cells (not just one)
* Example:
  + Dim x As Integer
  + For x = 2 To 20 ‘because we are starting from row 2, the other is a header
    - If Cells(x, 2).Value >= 35 Then
      * Cells(x, 3).Value = “Pass”
      * Cells(x, 3).InteriorColor = vbGreen ‘giving a color
    - Else
      * Cells(x, 3).Value = “Fail”
      * Cells(x, 3).InteriorColor = vbRed
    - End If
  + Next x

**SELECT CASE STATEMENT**

* Alternative to using multiple If statements
* Example:
  + Dim var1 As Integer
  + var1 = InputBox(“Enter Month Number”)
  + Select Case var1
    - Case 1: MsgBox “Month is Jan”
    - …
    - Case 12: MsgBox “Month is Dec”
    - Case Else: MsgBox “Invalid Month”
  + End Select

**MESSAGE BOX**

* MsgBox “Welcome”
* MsgBox 10000
* MsgBox #10/12/2020#
  + For showing the dates
* Button styles
  + MsgBox “Welcome”, 1
    - When we change the number, we are changing the styles
    - 1 – Ok, cancel
    - 2 – About, Retry, Ignore
    - 3 – Yes, No, Cancel
    - 4 – Yes, No
    - 5- Retry, Cancel
    - 6 – Ok
    - 7 – Ok
    - 8 – Ok
  + Instead of numbers, we can use Tab after comma to select the button we want
* More advanced button styles
  + 16 – error message - vbCritical
  + 32 – help button - vbInformation
  + 48 – warning with a sound - vbExclamation
  + 64 – information with a sound – vbQuestion
* What will happen when the user clicks on a button
  + We have to capture the output into a variable
  + If we want an output of a function, we have to use brackets
  + Example:
    - a = MsgBox (“Welcome”, 1)
    - MsgBox a
    - If a = 1 Then
      * MsgBox “Ok button was pressed”
    - Else
      * MsgBox “Cancel button was pressed”
    - End If
    - - because Ok has value 1 and Cancel has value 2
* Giving a title to the message box
  + MsgBox “Welcome”, vbOkOnly, “TutorialsPoint”
    - After the second comma we can write the title

**INPUT BOX**

* Inputbox “Enter the data”
* We have to store the input to a variable (with brackets)
* Example
  + a = input(“Enter the data”, title\_name. default\_text, xpos. ypos)
  + MsgBox a
    - Default\_text is the pre-filled answer
    - Xpos, ypos for example 200,200

**ROWS AND COLUMNS COUNT**

* Example:
  + Dim x As Integer
  + Dim y As Integer
  + x = Rows.Count
  + MsgBox x
  + y = Columns.Count
  + MsgBox y
* If the number is expected to be big we have to use Long instead of Integer
  + Integer: -32,768 to +32,728
  + Long: -2,147,483,647 to +2,147,483,647

**STRING FUNCTIONS LEFR-RIGHT**

* Range(“b2”).Value = Left(Range(“b2”).Value, 2)
  + This will return the 2 left most characters
* Example with for loop
  + Dim x As Integer
  + For x = 2 to 8
    - Cells(x, 2).Value = Left(Cells(x, 1).Value, 2)
    - Cells(x, 3).Value = Right(Cells(x, 1).Value, 2)
  + Next x

**STRING FUNCTIONS UCASE-LCASE**

* Uppercase and lowercase
* In excel we use UPPER() and LOWER()
  + In VBA it is UCase and LCase
* Example with for loop
  + Dim x As Integer
  + For x = 2 to 8
    - Cells(x, 2).Value = UCase(Cells(x, 1).Value)
    - Cells(x, 3).Value = LCase(Cells(x, 1).Value)
  + Next x

**STRING REVERSE**

* Functions that are not available to excel
* We need to use the function StrReverse()
* Example
  + Dim x as integer
  + For x = 2 to 8
    - Cells(x, 2).Value = StrReverse(Cells(x, 1).Value)
  + Next x

**CREATE FUNCTIONS**

* Example:
  + Function add2Numbers (x As Integer, y as Integer) As Integer
    - Add2Numbers = x + y
  + End Function
* Then we can use the function directly in Excel

**DATES**

* Date will show the current date
  + Example:
    - Dim a as variant
    - A = date
    - Msgbox a
* CDate to convert our date to a date
  + Dim b as variant
  + B = CDate(“20 Dec 2020”)
  + Msgbox b

**DATE ADD**

* Example:
  + Dim a As Variant
  + A = Date
  + MsgBox a
  + -> return a date
* Example:
  + Dim b As Variant
  + b = CDate(“20 Dec 2020”)
  + MsgBox b
* DateAdd – when we want to add dates
* Example:
  + Mydate = #11/12/2016#
  + MsgBox mydate
* Example, adding days, months and years:
  + MsgBox DateAdd(“yyyy”, 1, mydate)
  + MsgBox DateAdd(“m”, 1, mydate)
  + MsgBox DateAdd(“d”, 1, mydate)
    - Adding 1 year, month and day to the given date
* Example, adding hours, minutes and seconds
  + MsgBox DateAdd(“h”, 1, “31 dec 2020 12:00:00”)
  + MsgBox DateAdd(“n”, 1, “31 dec 2020 12:00:00”)
  + MsgBox DateAdd(“s”, 1, “31 dec 2020 12:00:00”)

**DATE PART**

* Separating day, month and year from a year
  + Using function DatePart
* Example:
  + Dim mydate As Variant
  + Mydate = #12/20/2016#
  + MsgBox DatePart(“yyyy”, mydate)
  + MsgBox DatePart(“d”, mydate)
  + MsgBox DatePart(“m”, mydate)
  + MsgBox DatePart(“q”, mydate)
    - q – displays quarter

**DD/MM/YYYY**

* another method of retrieving day, month and year from a date
* example:
  + Dim mydate As Variant
  + mydate = #12/20/1016#
  + MsgBox Day(mydate)
  + MsgBox Month(mydate)
  + MsgBox Year(mydate)

**TIME FUNCTION**

* Msgbox Now()
  + Displays day and time
* Msgbox Time()
  + Displays only time
* Msgbox Hour(Time)
  + Displays hour
* Msgbox Minute(Time)
  + Displays minute
* Msgbox Second(Time)
  + Displays second
* Instead of Time in the brackets, we can write whichever value we want
  + Msgbox Hour(“3:10:13”)
  + Msgbox Minute(“3:10:13”)
  + Msgbox Second(“3:10:13”)
* If we want proper format of time, wan use TimeSerial() function
  + Msgbox TimeSerial(3, 4, 5)
    - Returns 3:04:05 AM
  + We can also use TimeValue() function for the same results
    - But we are using different syntax
    - Msgbox TimeValue(“20:10:10”)
      * Returns 8:10:10 PM

**1D ARRAY STATIC**

* Array is a memory location that store similar types of data (it holds them)
  + i.e. one array can handle text, the other numbers…
* instead of creating hundreds of variables, we just create an array
  + seems similar to lists in Python
* it is basically a column – Series in Python
* example:
  + Dim marks (3) As Integer ‘THE NAME MARKS CAN BE CHANGED TO ANYTHING
    - ‘ALL ARRAYS START FROM 0, THAT’S WHY WE WROTE 3 IF WE NEED 4 VARIABLES
  + Marks(0) = 10
  + Marks(1) = 20
  + Marks(2) = 30
  + Marks(3) = 55
  + For i = 0 To 3
    - MsgBox marks(i)
  + Next
* All arrays start from 0

**2D ARRAY STATIC**

* It is basically a table – DataFrame in Python
* Also starts from 0
* i.e. a(1, 1) to find the value in the table
* example:
  + Dim a(2, 2) As Integer
  + a(0, 0) = 10
  + a(0, 1) = 20
  + a(0, 2) = 30
  + a(1, 0) = 40
  + a(1, 1) = 50
  + a(1, 2) = 60
  + a(2, 0) = 70
  + a(2, 1) = 80
  + a(2, 2) = 90
  + For i = 0 To 2
    - For j = 0 To 2
      * MsgBox a(i, j)
    - Next
  + Next

**DYNAMIC ARRAY**

* If we are not sure about the size of the array, we can leave it empty
  + i.e. Dim a() As Integer
    - after that we need to define it as ReDim a(2)
* if we ReDim again, the previous values will become 0
  + if we want to keep them, we have to use ReDim preserve
* example:
  + Dim a() As Integer
  + ReDim a(2)
    - a(0) = 10
    - a(1) = 20
    - a(2) = 30
  + ReDim preserve a(4)
    - MsgBox a(0)
    - MsgBox a(1)
    - MsgBox a(2)

**WORKSHEET SELECTION CHANGE**

* Event – any occurrence or any change in your worksheet (open/close, changing cells)
* Triggering the event
  + i.e. anywhere where we move a cursor we want to change the background color
* in VB, we just double click on the sheet we want to work in
  + then change General to Worksheet
  + 🡪 this is only when we want it to work only for one sheet
  + It will create a special Sub
    - We can change on the right side what kind of Sub we want
* Event - SelectionChange
* example:
  + Target.Interior.Color = vbRed
    - This will change the background color to red anywhere where we move the cursor

**DOUBLE CLICK**

* We define what will happen if we double click on something
* When selecting a worksheet (double clicking on it), on the right side we can choose BeforeDoubleClick
* Event - BeforeDoubleClick

**SHEET ASCTIVATE**

* When selecting a worksheet (double clicking on it), on the right side we choose Activate
* Whenever a person opens the sheet, the event will be triggered
* Event - Activate

**WORKBOOK OPEN**

* Event – Open
* We double click on ThisWorkbook on the left side
  + That way it is applicable to the whole workbook
* What should happen when the user opens the workbook
* Example:
  + a = InputBox(“Enter Password”)
  + If a = 123 Then
    - MsgBox = “Welcome”
  + Else
    - MsgBox = “Incorrect Password”
    - ThisWorkbook.Save
    - ThisWorkbook.Close
  + End If

**CREATE USER FORM**

* We need Properties Window and Project Explorer
  + Accessible from View
* Insert 🡪 User Form
* We can change how it looks like in Properties Window
* We need to select the screen and Execute to run it
* Toolbox
  + View 🡪 Toolbox

**COMMAND BUTTON**

* Used for User Form using Toolbox
* Label
  + Basically just a value within the User Form
* CommandButton
  + By double clicking we open a code eitor
  + Example:
    - UserForm2.Label.Caption = UserForm.Label.Caption + 1
      * This will take the value of Label and add 1 when clicking on the button
* We can drag items by holding CTRL and dragging

**RADIO BUTTON – OPTION BUTTON**

* Adding option buttons
* Toolbox 🡪 OptionButton
  + We have to use one for each option
* Code in Submit button (Command Button)
  + Dim answer As String
  + If UserForm3.OptionButton1.Value = True Then
    - answer = “Bus”
  + ElseIf UserForm3.OptionButton2.Value = True Then
    - answer = “Flight”
  + Else
    - answer = “Choose the option”
  + End If
  + MsgBox = “You have selected” & answer
* We have to use & when adding variable to a text
* When we are using True and False, we have to use Value (not Caption)

**CHECK BOX**

* Toolbox 🡪 Checkbox
* We can name cells in Excel
  + Click on cell and on the left side change the i.e. B3 to your name
* Example:
  + We changed the cell names to physics\_range and so no
  + If userform4.CheckBox1.Value = True Then
    - Thisworkbook.Sheets(1).Range(“physics\_range”).Value = Thisworkbook.Sheets(1).Range(“physics\_range”).Value + 1
  + EndIf

**LIST BOX**

* Toolbox 🡪 ListBox
* The list box will be empty at first, we have to add some data
  + Properties 🡪 RowSource
    - Here we will write the range that we have prepared in Excel
    - i.e. Sheet1!a1:a12
* it is better to change to DoubleClick on the right side because one click is better to just select
* example:
  + we open the editor directly in ListBox
  + MsgBox UserForm5.ListBox.Value

**FRAMES**

* Used to collect all the different options, such as checkbox, option button and so on
* Toolbox 🡪 Frame
* We draw a Frame and then whatever boxes, buttons and labels we want inside
* Whatever action you do with a frame will be applicable to everything inside the frame
* Example:
  + We created a frame with checkboxes and a button
  + Outside of the frame we created a button for showing/hiding the frame
  + Double click on the show/hide button outside the frame
    - UserForm6.Frame1.Visible = Not Userform6.Frame1.Visible
  + Here we are using Not operator to change the value from True to False or opposite

**MULTIPAGE**

* Creating different pages on the same User Form
* Toolbox 🡪 MultiPage
* By default, we will get 2 pages
  + We can click on the pages and click new page
* So we don’t have to use multiple user forms

**BACKGROUND**

* Adding background to User Form
* If we want a picture as a whole background
  + Properties 🡪 Picture
* If we just want to insert a picture
  + Toolbox 🡪 Image

**TEXT BOX**

* Toolbox 🡪 TextBox
* i.e. for log in with password
  + We should also add labels to specify what are the text boxes for
  + We also need a command button for logging in and cancelling
* Changing password button to be encrypted
  + Selecting the button
  + Properties 🡪 PasswordChar and type \*
* Double click on OK button
  + UserName = UserForm9.TextBox1.Value
  + Password = UserForm9.TextBox2.Value
  + If UserName <> “” And Password <> “” Then
    - MsgBox “Login Success”
    - UserForm9.Hide
  + Else
    - MsgBox “Kindly enter the credentials”
  + EndIf

**USER FORM EVENT**

* Calling a User Form without going to the VBA Editor
  + Creating a button or shape and assigning a macro to it
* Insert 🡪 Module
  + UserForm9.Show
* Drawing a shape 🡪 right click 🡪 assign macro