**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