Day 1

1pm-2pm

INTRODUCTION TO VBA

Microsoft Visual Basic Fundamentals

The Microsoft Visual Basic Interface

VBA in Visual Basic

Macros

Writing Code

Introduction to VBA

Visual Basic for Applications is a programming language incorporated in Microsoft Excel, Access, PowerPoint and even Word, which let you do all things you already know about them and much more. For example, you want that every time you open a specific Microsoft Word file it writes automatically the current date two lines below where you left last time. Or maybe you want a whole spreadsheet of Excel without formulas on it and still applying them as if they were there. How would you do that? All these things and much more are done with Visual Basic for Applications for Microsoft Office.

* Microsoft Visual Basic Fundamentals

1. Access the (VisualBasic Editor)VBE Window using Alt + F11
2. Add the developer tab in the ribbon

File🡪options🡪customize ribbon🡪enable developer tab

* The Microsoft Visual Basic Interface
* VBA in Visual Basic
* Macros

A Macro is an automated sequence which will apply every time you play it.

Let’s see a practical example of it:

Imagine that in your job you do the same process every morning. It takes some valuable time and even you’re getting bored of that.

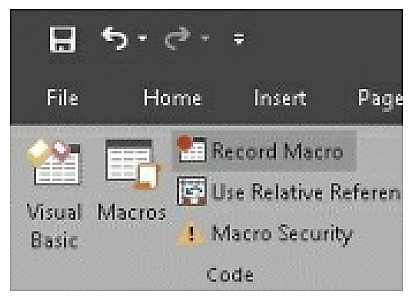
The process is the following:

a) You receive a Microsoft Excel file from your boss with some data and you need to write the date using Year, Month and Day in different columns. You do this because it is the format your job needs and you’ve been adding the same values every day for a few years.

Prepwork: Create a worksheet with columns: date | month | year | your email | mobile number

To create a Macro, follow the sequence below:

1. Click the Record a Macro Button.



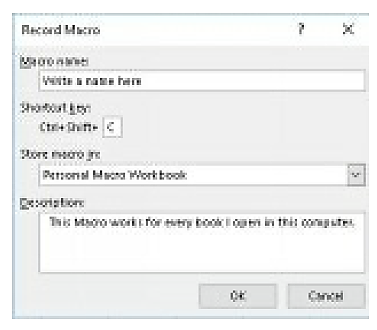
b) Write a name for your Macro. (Needed)

c) A shortkey which every time you press will Run the Macro. Be careful, don’t add Ctrl + C or Ctrl + v, otherwise it won’t copy or paste anymore, but run the Macro. In case you want a more specific shortcut, hold the shift key as you press a letter. For example, ctrl + shift + c. To make it work, don’t press ctrl as you add a short cut. (Optional)

d) Store Macro in: Personal workbook: Will be available for all the files you open with Excel on that computer; New Workbook will be available for a new file only. This workbook, will apply only to the current open file. (Needed to choose one)

e) Write a description about what that Macro does. (Optional)

f) Click Ok.



g) Start doing everything you always do, which would be adding the current date in this case.

Note:

Add current day number

=day(now())

Add current month number

=month(now())

Add current year

=year(now())

h) Once you finish, go back to the Record Macro Button, which now is called Stop Recording. Press it and now should be saved.

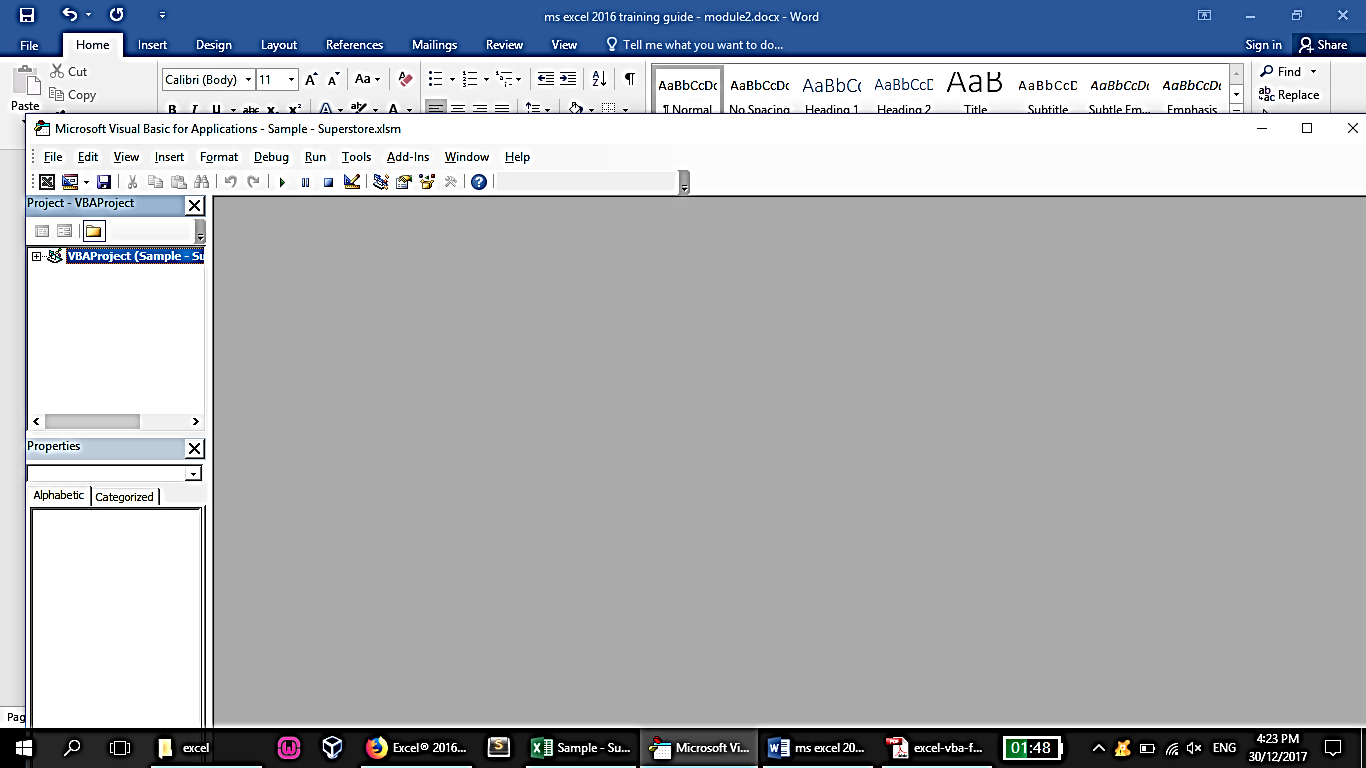
Note:

If you saved your macro in the “personal macro workbook” (making it available in all excel file in the entire computer), you might not be able to edit/delete the macro since the “personal macro workbook” is hidden by default (try it!). To resolve this, ribbon🡪view🡪unhide🡪PERSONAL.XLSB, then try again

* Writing Code (Quick sample)

**Accessing the VBA editor (or VBE)**

1. To access the VBE, press ALT+F11 or go to Developer tab in your ribbon (if you configured it to be displayed). You should see this window



1. The VBE sees all your macros as well as open excel workbooks. But it can’t be opened without first opening an excel workbook.

Note:

Increase font size in VBE

Tools🡪options🡪editor format

1. A VBA consists of Sub procedures and / or Functions like the ff:

Sub addnumbers()

myanswer = 20 + 20

Msgbox “the sum is ” & myanswer

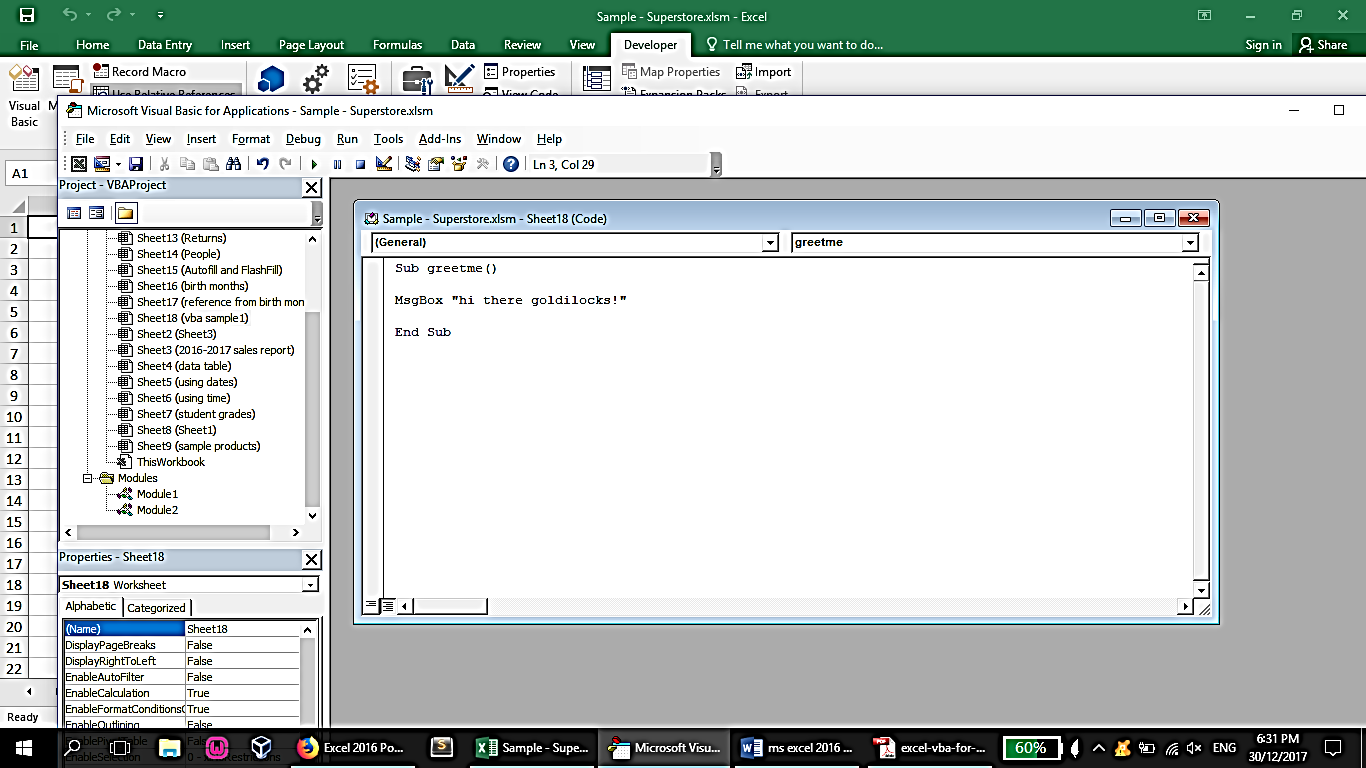
End Sub

Function faddnumbers(a,b)

faddnumbers=a+b

End Function

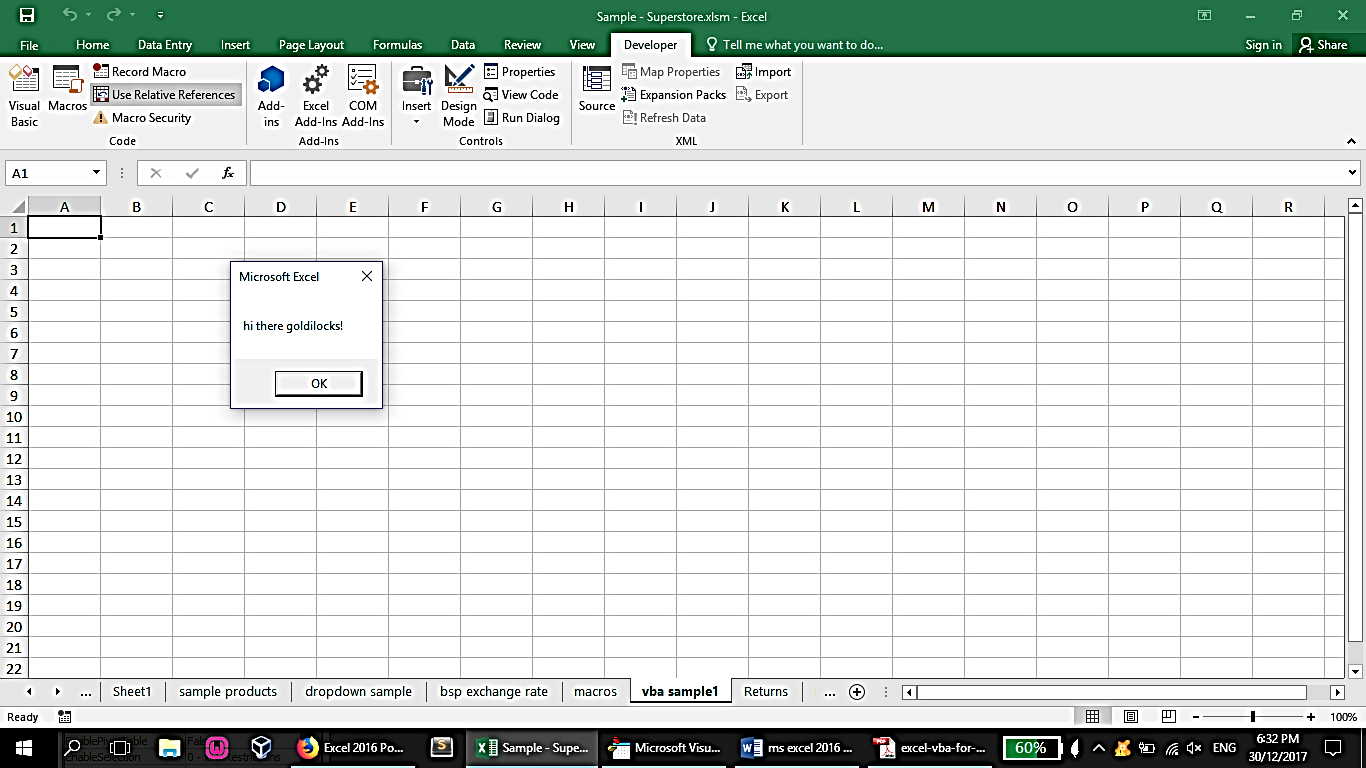
‘And these can be used inside a sheet or as a module (in VBE)



Note:

* Adding custom code in VBE usually requires that an Excel Workbook must be saved as a macro-enabled workbook (xlsm)
* To assign a shortcut key to it, back to worksheet🡪developer tab🡪macros🡪select the macro🡪options

‘Test it by running it from VBE



**Object References**

1. You can make reference to a workbook by:

Application.Workbooks(“Book1.xlsx”)

1. You can make reference to a sheet inside a workbook by:

Application.Workbooks(“Book1.xlsx”).Worksheets(“Sheet1”)

1. You can make reference to a cell range by:

Application.Workbooks(“Book1.xlsx”).Worksheets(“Sheet1”).Range(“A1”)

1. You can make reference to an active workbook’s worksheet (done automatically if you omit some parts like the ff code):

Worksheets(“Sheet1”).Range(“A1”)

**Understanding Objects**

1. Objects have properties

Worksheets(“Sheet1”).Range(“a1”).Interior.Color

Worksheets(“Sheet1”).Range(“a1”).Value

1. We can also set values for the properties

Worksheets(“Sheet1”).Range(“a1”).Interior.Color=rgb(0,0,250)

Worksheets(“Sheet1”).Range(“a1”).Value=”vba placed me here”

1. We can also invoke object methods

Worksheets(“Sheet1”).Range(“a1”).clearcontents

2pm-3pm

VARIABLES AND DATA TYPES

Variables

Introduction to Data Types

Integral Numeric Variables

Decimal Variables

Any-Type Variables

The Scope or Lifetime of a Variable

The Access Level of a Global Variable

**Understanding variables and data types**

1. Creating a variable (no specific data type)

myvariable = value

ex:

mybirthyear = 2007

mybirthplace = “manila”

1. VBA’s Built-In Data Types

VBA’s Built-In Data Types

Data Type Bytes Used Range of Values

Byte 1 0 to 255

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 –3.40 to –1.40E-45 for negative values;

1.40E-45 to 3.40 for positive values

Double 8 –1.79E308 to –4.94E-324 for negative values;

4.94E-324 to 1.79 for positive values

Currency 8 –922,337,203,685,477 to 922,337,203,685,477

Date 8 1/1/0100 to 12/31/9999

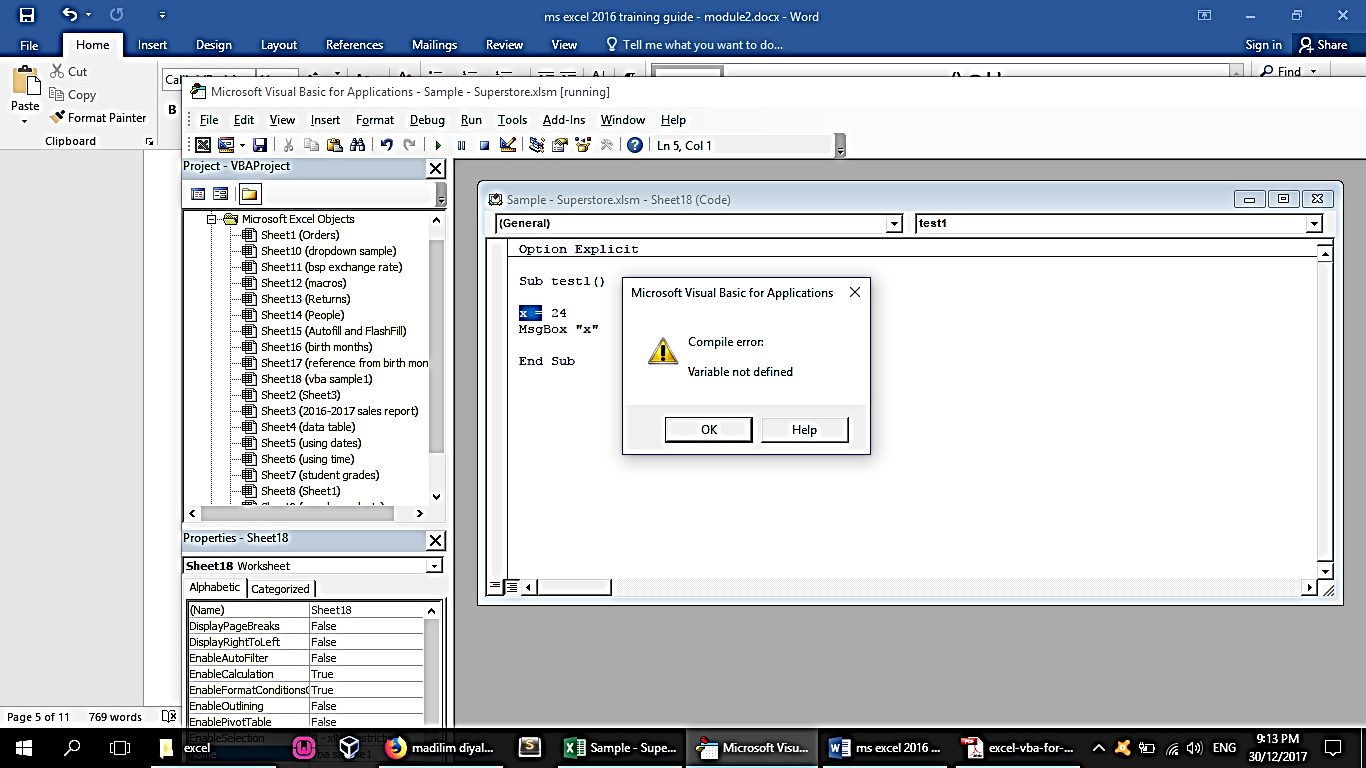
Object 4 Any object reference

String 1 per character Varies

Variant Varies Varies

1. Before you use variables in a procedure, it’s an excellent practice to declare your variables — that is, tell VBA each variable’s data type. Declaring your variables makes your macro run faster and use memory more efficiently. The default data type, Variant, causes VBA to repeatedly perform time-consuming checks and reserve more memory than necessary. If VBA knows a variable’s data type, it doesn’t have to investigate and can reserve just enough memory to store the data.
2. To force yourself to declare all the variables you use, include these two words as the first statement in your VBA module (When this statement is present, you won’t be able to run your code if it contains any undeclared variables.):

Option Explicit



1. Declaring variables

Dim [variable name] as [data type]

‘doesn’t support group declaration like the ff:

Dim [variable1],[variable2] as integer

‘the example above only sets [variable2] as integer.

‘use this for multiple declaration

Dim [variable1] as integer,[variable] as integer

1. Variable scopes

**Scope How the Variable Is Declared**

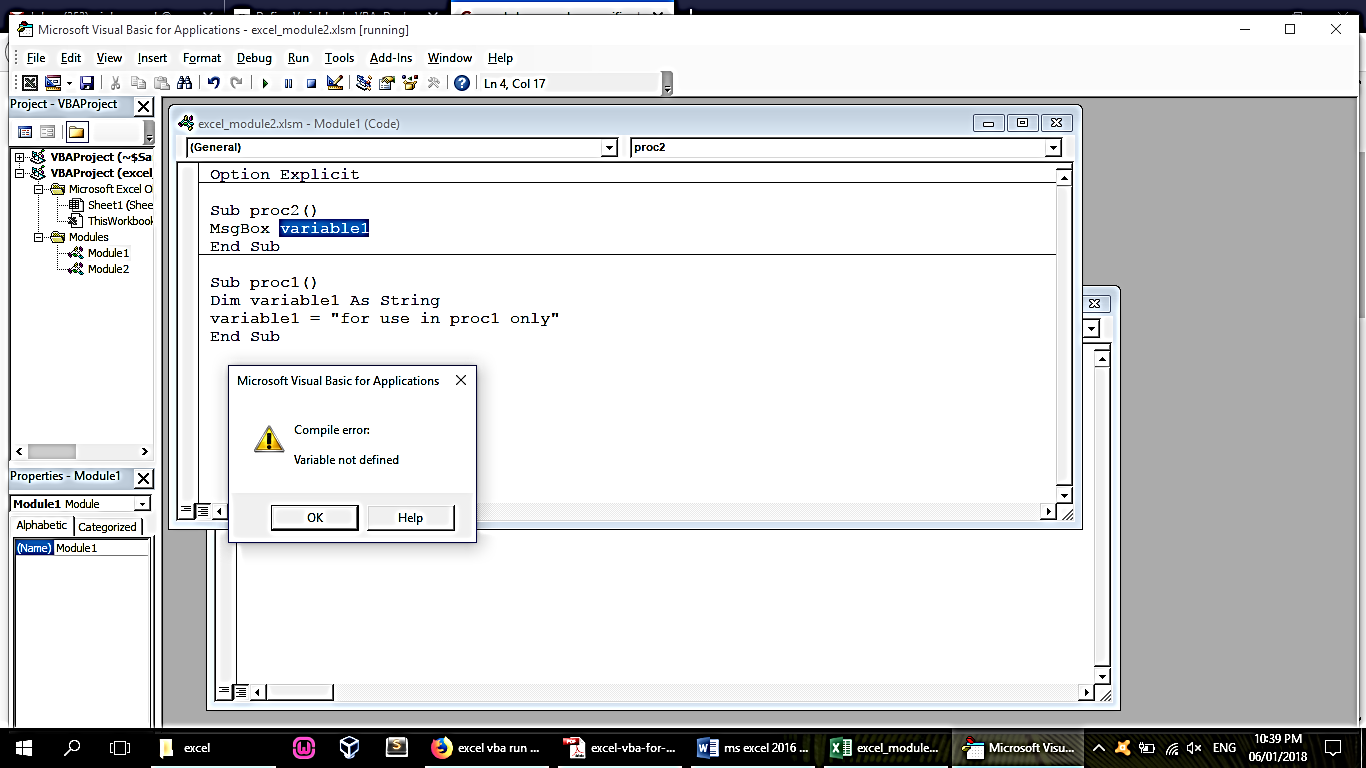
Procedure only By using a Dim or a Static statement in the procedure that uses the variable

Module only By using a Dim or a Private statement before the first Sub or Function statement in the module

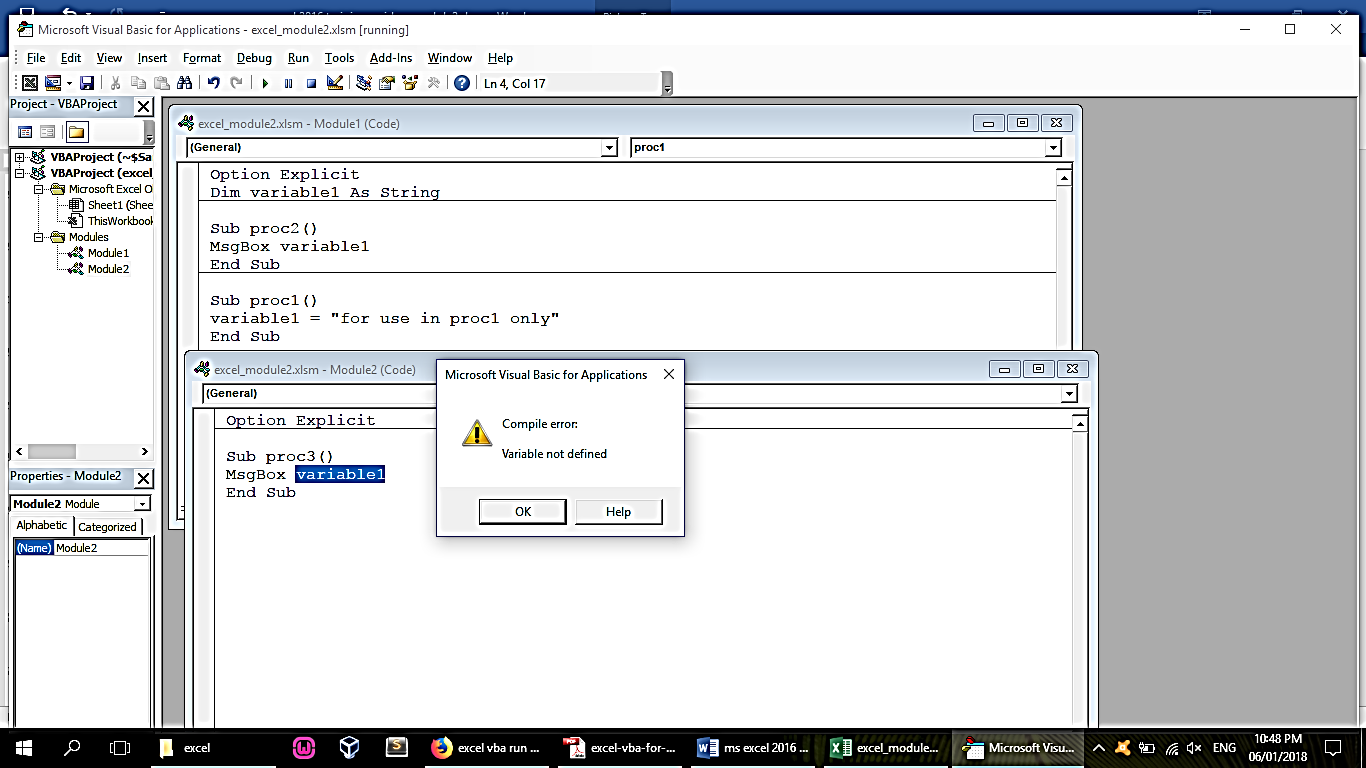
All procedures in all modules By using a Public statement before the first Sub or Function statement in a module

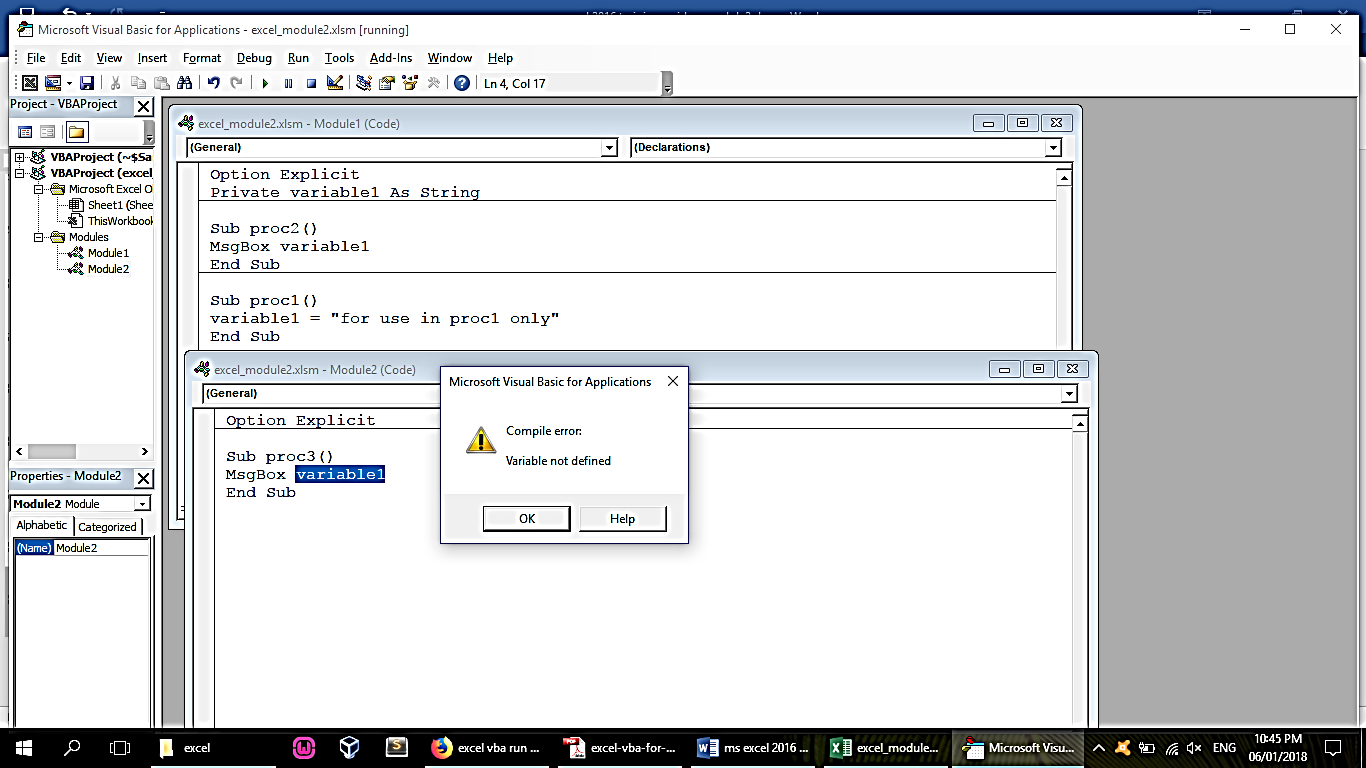
Examples:

1. Declaring variable inside a sub or function only (although usable in other subs on the same module or in other modules, value gets reset)

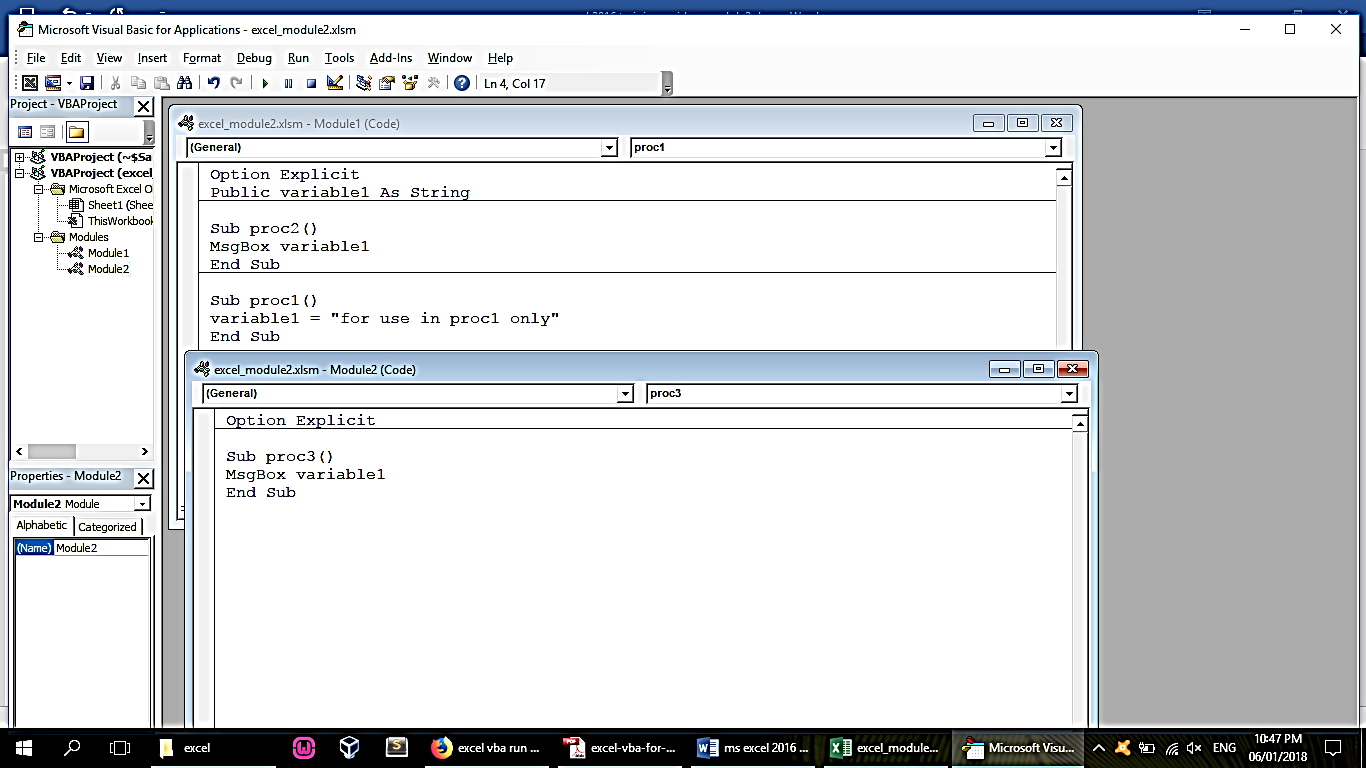


1. Variable for all sub and function in the entire module (but the value gets reset for every sub or function. Also not usable in other modules)



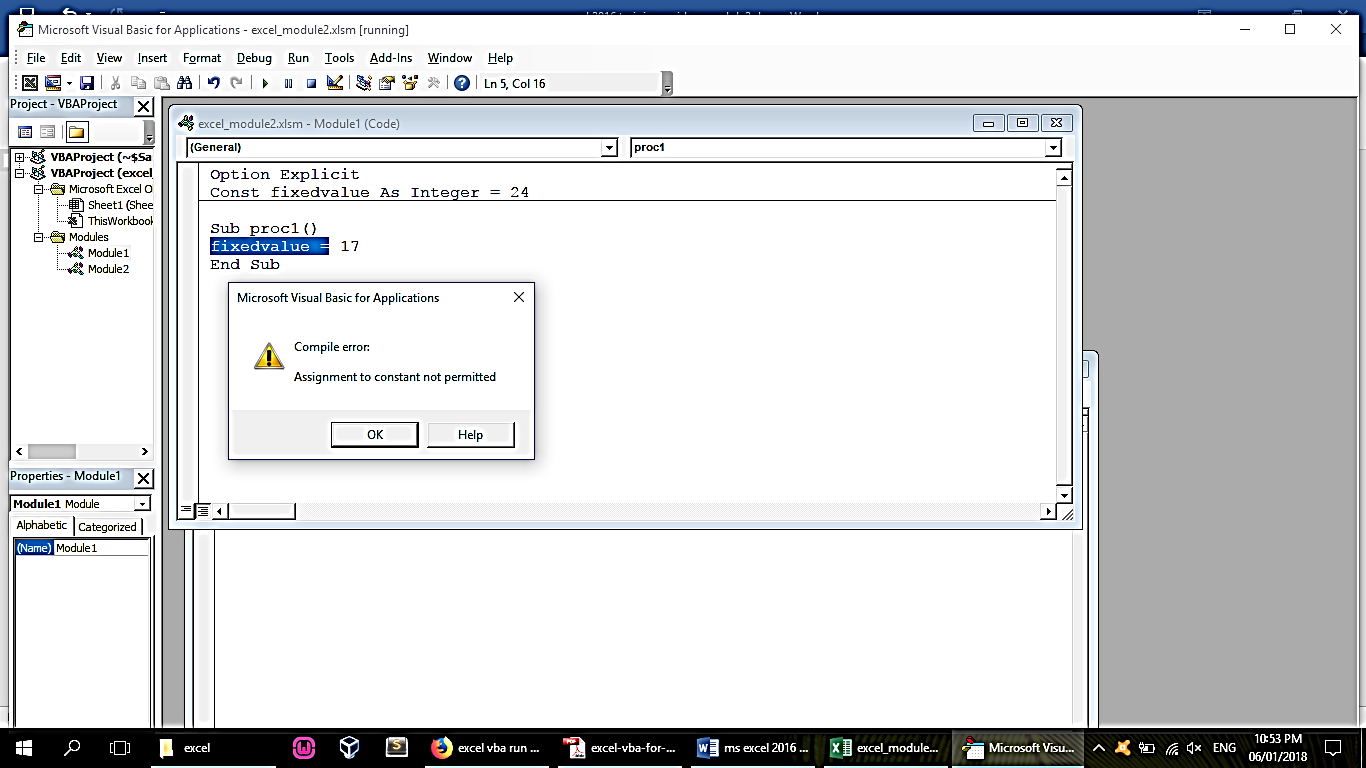


1. All module declaration (global)



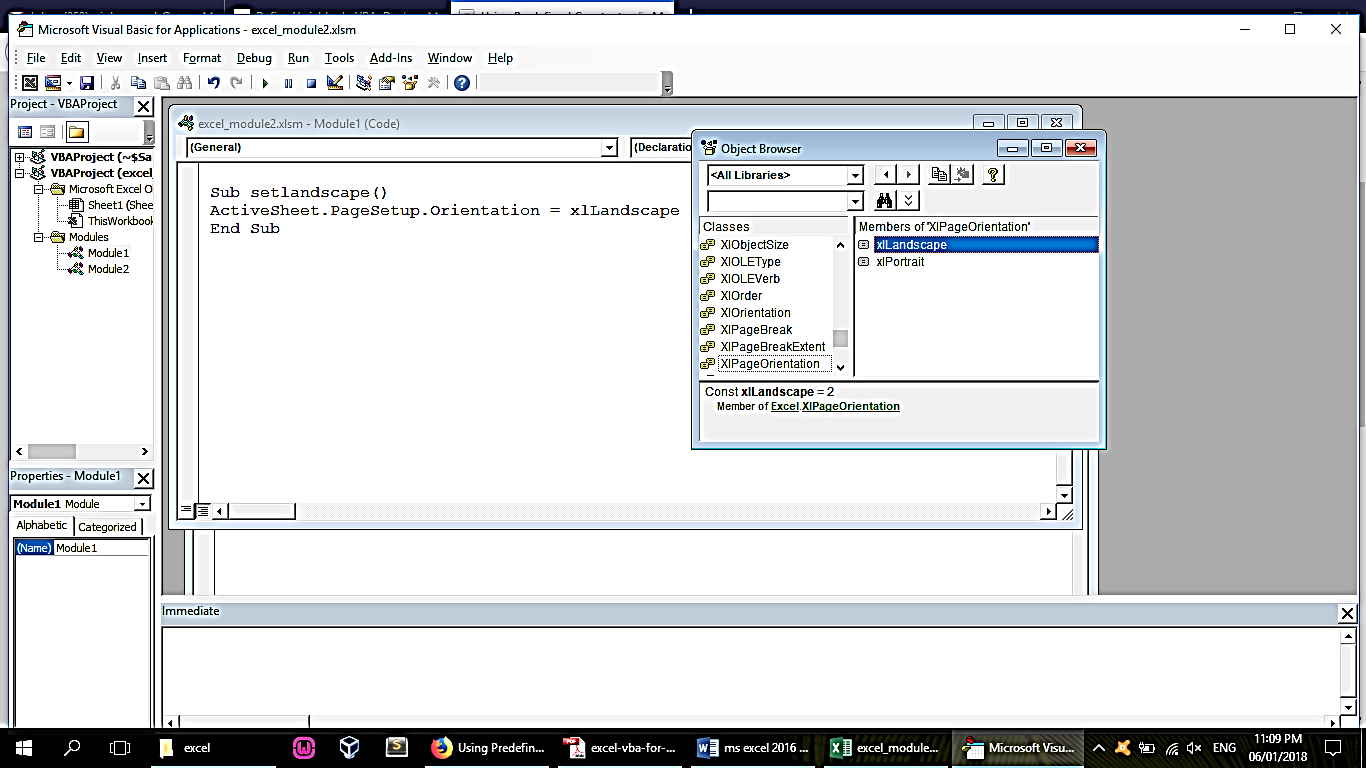
**Working with constants**

‘variables declared as constants cannot be re-assigned values



**Pre-made constants**

‘there are many predefined constants in excel functions. You may check their underlying values in the object explorer.



**Working with strings**

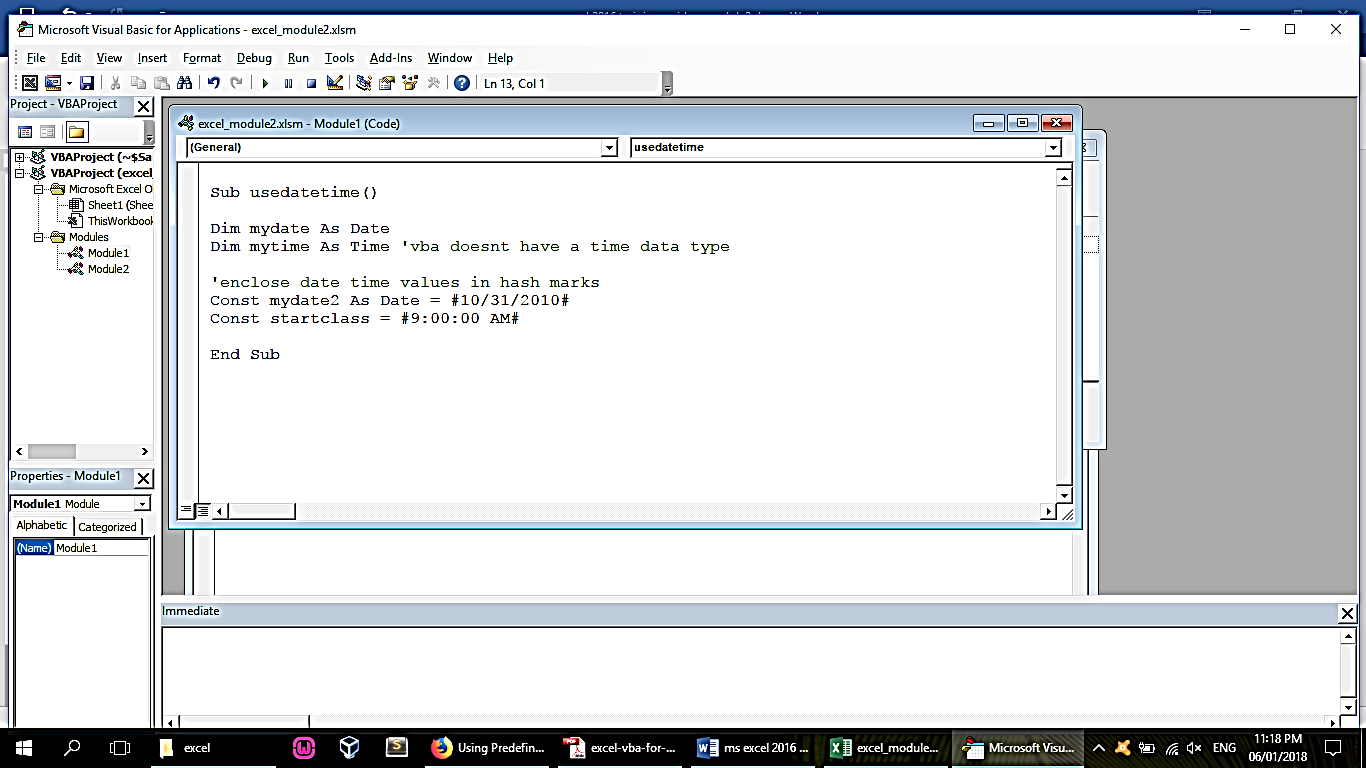
1. Declaring flexible-length string

Dim [var] as string

1. Declaring fixed-length string (example: 10 chars max)

Dim [var] as string \* 10

**Working with dates**



**Using Assignment Statements**

x = 1

x = x + 1

x = (y \* 2) / (z \* 2)

HouseCost = 375000

FileOpen = True

Range(“TheYear”).Value = 2013

**Function Operator Symbol**

Addition +

Multiplication \*

Division /

Subtraction –

Exponentiation ^

String concatenation &

Integer division

(the result is always an integer) \

Modulo arithmetic

(returns the remainder

of a division operation) Mod

**Working with Arrays**

1. An array is a variable that holds multiple values assigned to array indices. An array must be declared before use.
2. To declare an array:

Dim array1(1 to 10) as string

‘if start index not defined, it start with 0

Dim array2(10) as string

1. To assign values to array

array1(1) = one

array1(2) = two

1. To create a dynamic array

‘declare an empty array first in the global declarations

Dim dynamicarray() as string

‘then once you have recomputed the possible size (example: using counta) you can then resize the array using redim. ‘Using +1 is a good idea so you will always have an empty array index at the end, but note that the last array index will ‘always be empty.

Redim dynamicarray(newsize + 1)

1. Declaring multi-dimensional array

‘this creates 3 storage index for each of the first set of index (total of 9 indices)

Dim multidimarray(1 to 3,1 to 3) as string

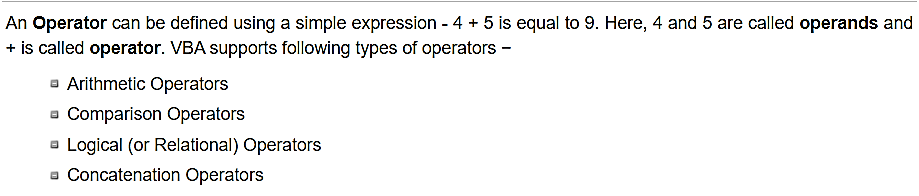
1. Assign value to multi-dimensional array

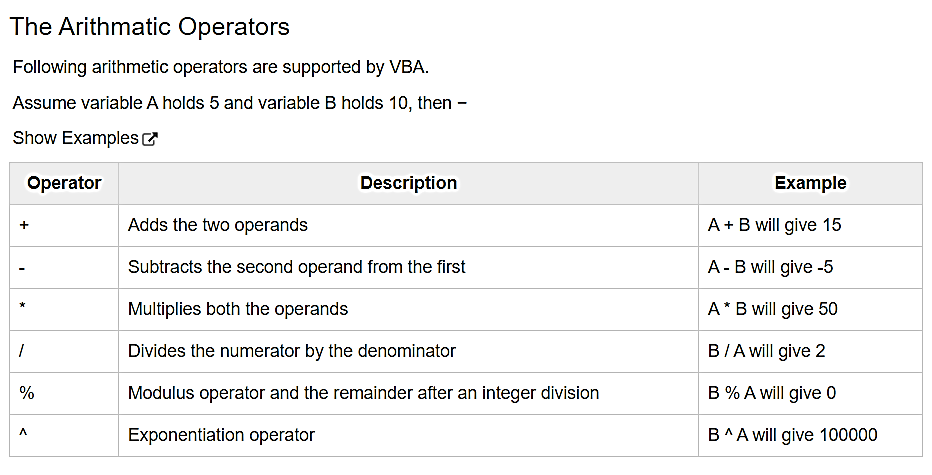
multidimarray(1,1) = A

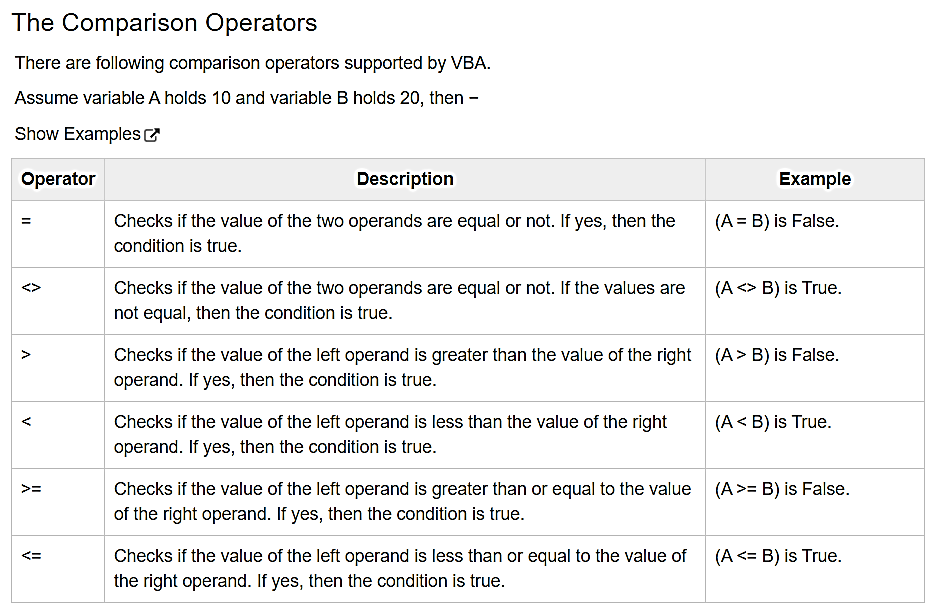
3pm-4pm

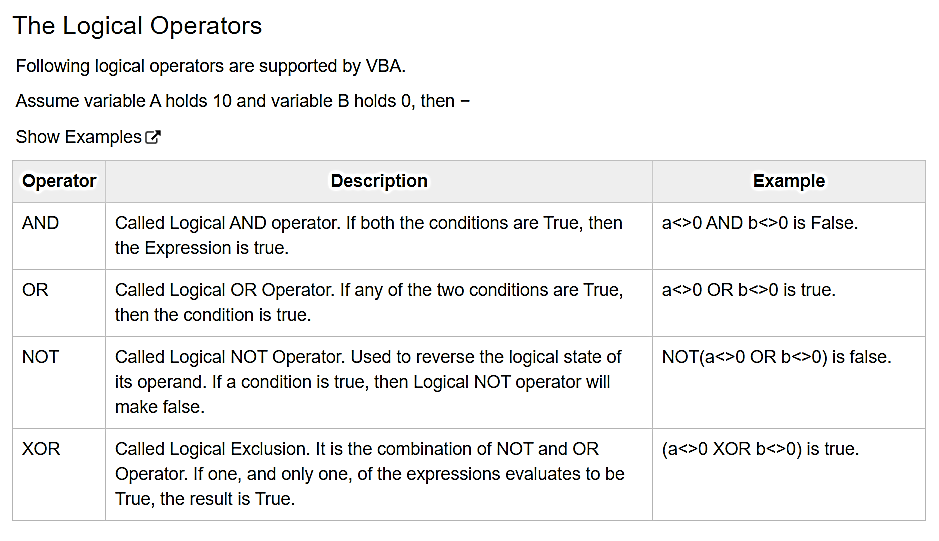
VBA Operators and Operands

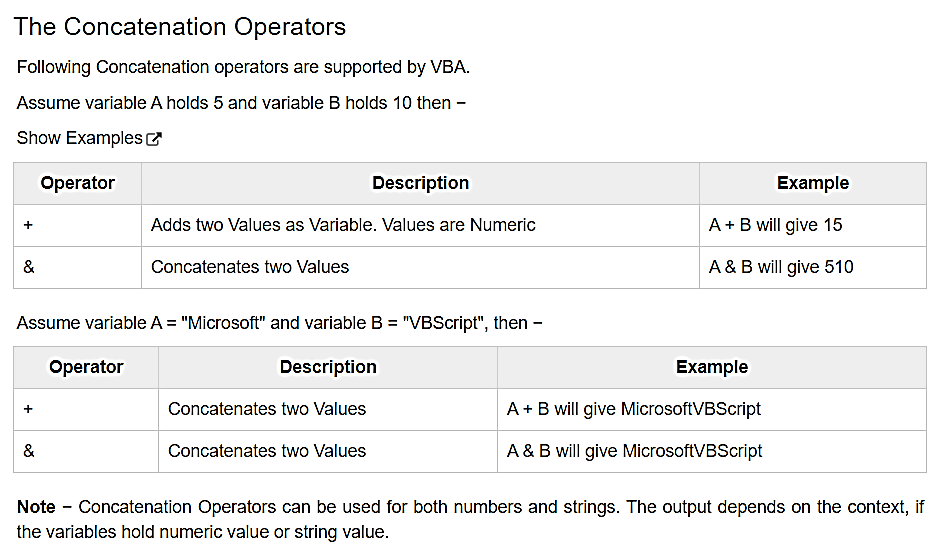
* VBA Operators











4pm-5pm

THE PROPERTIES WINDOW

The Appearance of the Properties Window

Types of Properties

**Buffer Topics**

Sample Useful Macros:

**Macro Activities**

Activity 1: You receive a Microsoft Excel file from your boss with some data and you need to write the date using Year, Month and Day in different columns. You do this because it is the format your job needs and you’ve been adding the same values every day for a few years.

Prepwork: Create a worksheet with columns: date | month | year | your email | mobile number

Note:

Add current day number

=day(now())

Add current month number

=month(now())

Add current year

=year(now())

Activity 2:

Autofit Columns

Sub AutoFitAllColumns()

Activate

Cells.Select

Cells.EntireColumn.AutoFit

End Sub

Autofit Rows

Sub AutoFitAllRows()

Cells.Select

Cells.EntireRow.AutoFit

End Sub

Activity 3: Highlight misspelled words

Sub HighlightMisspellings()

Dim cell As range

For Each cell In ActiveSheet.UsedRange

If Not Application.CheckSpelling(word:=cell.Text) Then

cell.Style = "Bad"

End If

Next cell

End Sub

Activity 4: Delete all empty worksheets

Sub DeleteBlankWorksheets()

Dim wsheet As Worksheet

On Error Resume Next

Application.DisplayAlerts = False

Application.ScreenUpdating = False

For Each wsheet In Application.Worksheets

If Application.WorksheetFunction.CountA(wsheet.UsedRange) = 0 Then

wsheet.Delete

End If

Next

Application.DisplayAlerts = True

Application.ScreenUpdating = True

End Sub

Activity 5: Sort all worksheets

Sub SortAllWorksheetsByName()

Dim i As Integer

Dim j As Integer

For i = 1 To Sheets.Count

For j = 1 To Sheets.Count - 1

If UCase$(Sheets(j).Name) > UCase$(Sheets(j + 1).Name) Then

Sheets(j).Move After:=Sheets(j + 1)

End If

Next j

Next i

End Sub

Activity 6: Hide all worksheets except the active

Sub HideAllExceptActiveSheet()

Dim ws As Worksheet

For Each ws In ThisWorkbook.Worksheets

If ws.Name <> ActiveSheet.Name Then ws.Visible = xlSheetHidden

Next ws

End Sub

Activity 7: Unhide all hidden worksheets

Sub UnhideAllWoksheets()

Dim ws As Worksheet

For Each ws In ActiveWorkbook.Worksheets

ws.Visible = xlSheetVisible

Next ws

End Sub

Activity 8: Save excel with timestamp in name

Sub SaveWorkbookWithTimeStamp()

Dim timestamp As String

timestamp = Format(Date, "dd-mm-yyyy") & "\_" & Format(Time, "hh-ss")

ThisWorkbook.SaveAs "C:UsersUsernameDesktopWorkbookName" & timestamp

End Sub

Activity 9: Convert all formulas into values (also do recorded macro)

Sub ConvertToValues()

With ActiveSheet.UsedRange

.Value = .Value

End With

End Sub

Activity 10: generate categorized sums and create chart (recorded Macro)

Activity 11:

Example: Select all values greater than 500 then convert to 0.

Sub FindReplace()

'Updateby Extendoffice

Dim Rng As Range

Dim WorkRng As Range

On Error Resume Next

xTitleId = "KutoolsforExcel"

Set WorkRng = Application.Selection

Set WorkRng = Application.InputBox("Range", xTitleId, WorkRng.Address, Type:=8)

For Each Rng In WorkRng

If Rng.Value > 500 Then

Rng.Value = 0

End If

Next

End Sub

Recorded Macro:

1. create another column and fill this new column with if condition for replacement

=if(b2>500,0,b2)

1. copy the results and paste special values to the original column to replace.