Excel
VBA programming
Visual Basic for Applications

Hervé Hocquard

http://www.labri.fr/perso/hocquard

Special type that can hold all kinds of value

THE VARIANT TYPE

The type of variant can handle any type of value. It is very flexible, particularly convenient when you do not know in advance which type to use.But be careful, do not overdo it, it is very slow because the checks are multiplied each time the corresponding variable is accessed..

Input: a, b (real)
Output: a / b if b ≠ 0, "division by zero" otherwise

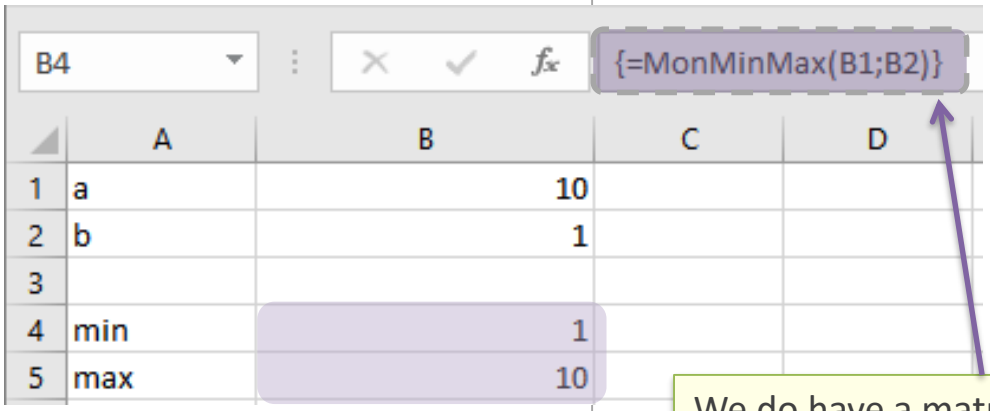
One blow, the function returns a real, another blow it must return a character string.

```
'use of the variant type
Public Function MyDivision(a As Double, b As Double) As Variant
'var. intermediate
Dim result As Variant
'calculation
If (b <> 0) Then
 result = a / b
Else
 result = "division by zero"
End If
'return the result
MyDivision = result
End Function
```

In the same variable result, of variant type, we can assign a real number and a character string.

You can use it to return an array. A function can therefore return several values at once, like the matrix functions of Excel (you must validate the entry of the function with the sequence of keys CTRL + SHIFT + ENTER).

```
'return multiple values
Public Function MonMinMax(a As Double, b As Double) As Variant
'an internal table - matrix 2 rows and 1 column
Dim t(1To 2, 1To 1) As Double
'identify the min and the max
If (a < b) Then
 t(1,1) = a
 t(2,1) = b
Else
 t(1,1) = b
 t(2,1) = a
End If
'return array
MonMinMax = t
End Function
```



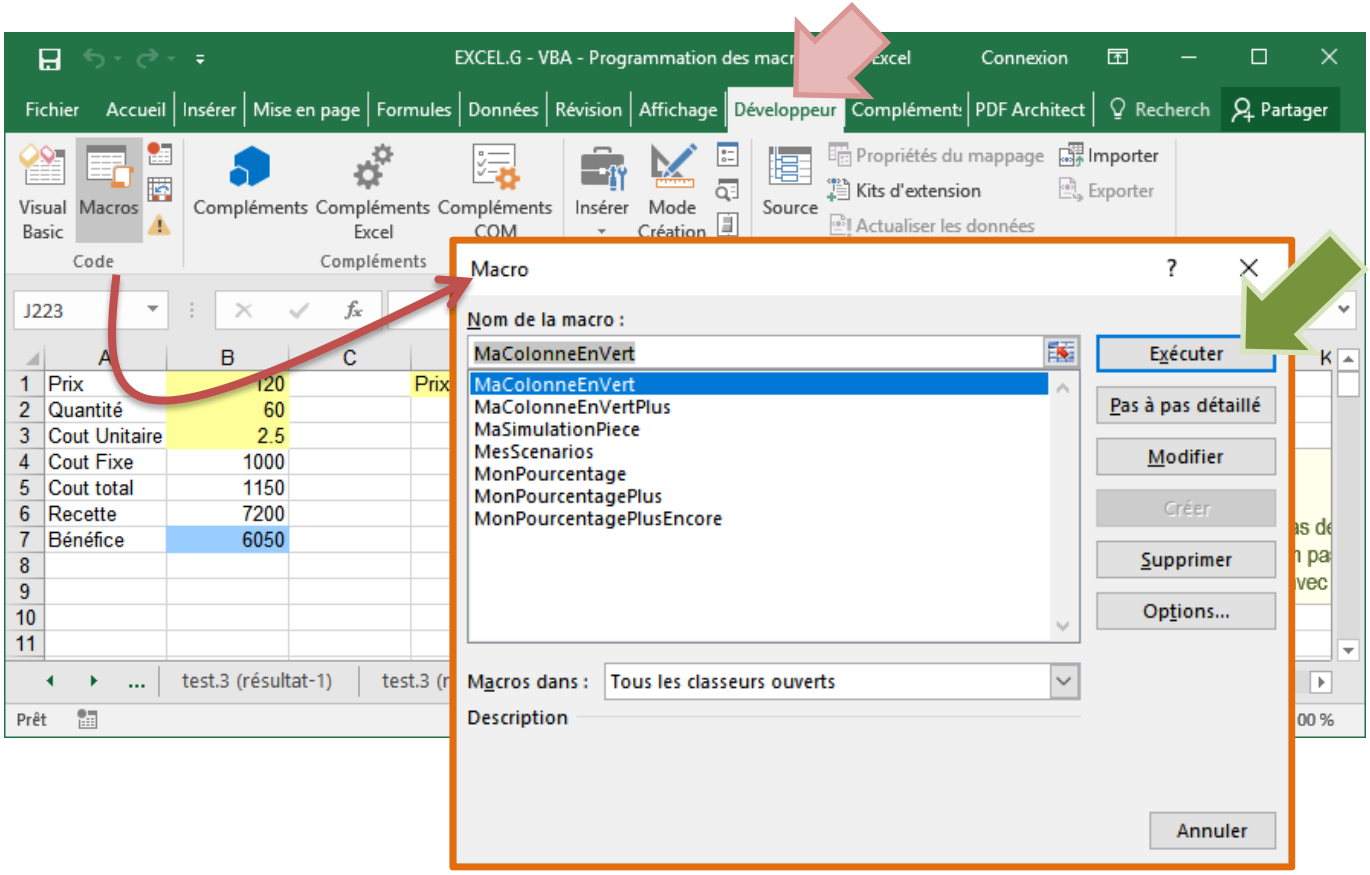
We do have a matrix function as can be seen from the braces {} which surround the call to the function.

Programming macros - Work directly on the sheets

MACROS (1)

Macros are also procedures that we create inside a module. But, unlike Function, those are Sub() without parameters that can directly manipulate (access and modify) Excel objects (workbooks, sheets, cells, charts, scenarios, pivot tables, etc.).

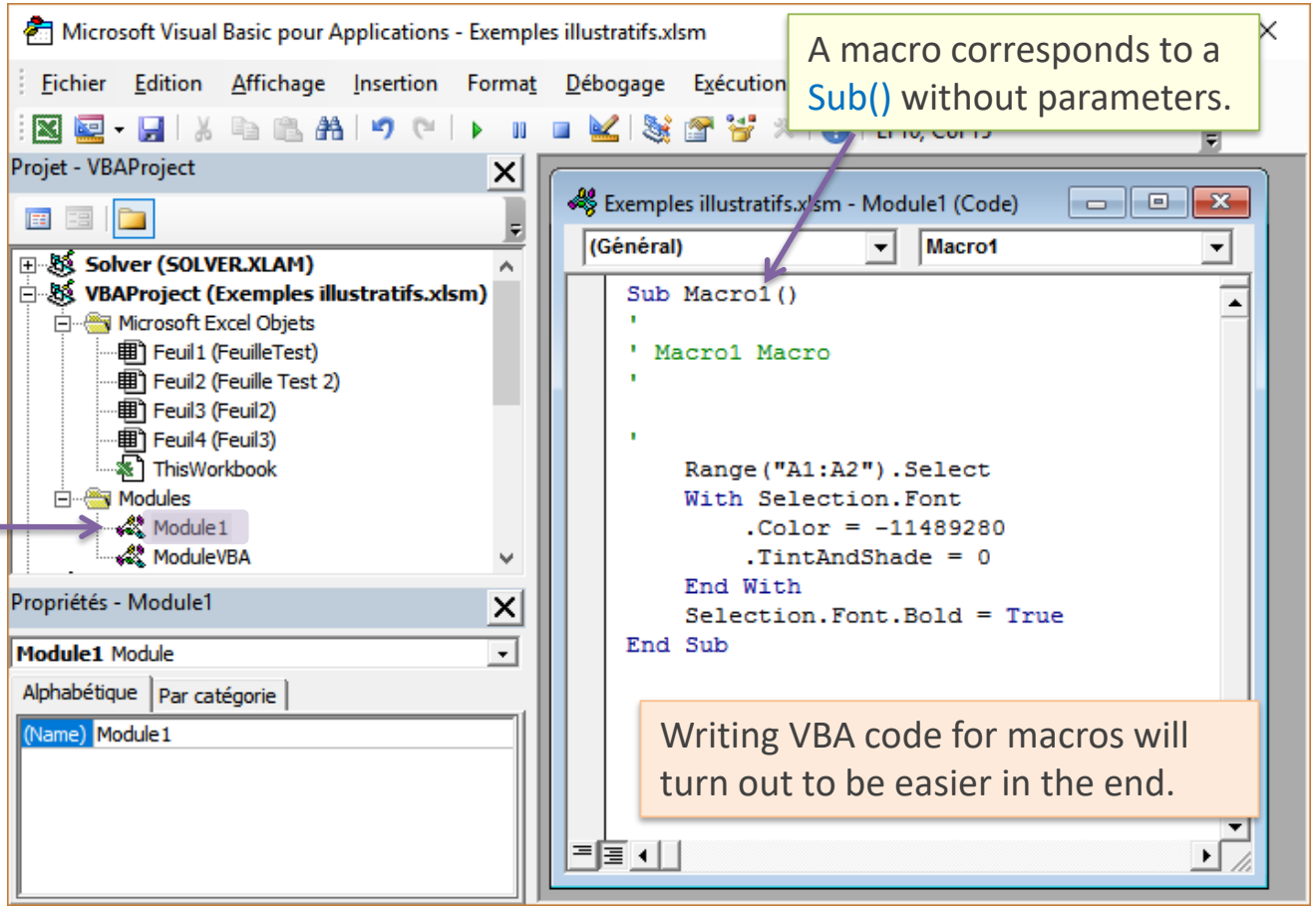
They do not perform the same. Instead of inserting them in a cell, they are launched globally via the MACROS button in the DEVELOPER ribbon.



An easy way to generate a macro is to run the macro recorder. VBA code is automatically generated.

Example: put in bold and green the contents of cells A1 and A2

	A
1	10
2	12



Benefits :

- There is nothing easier to produce code, you can create and run a macro without any programming concept.
- It gives us valuable information on the commands associated with Excel objects.

Disadvantages:

- We work with a fixed structure, if the configuration of the sheet changes, it is not possible to launch the macro.
- We do not benefit from the power of algorithmic structures.

Ultimately:

- It can help us write our code by giving us clues on the syntax of the commands and the appropriate objects to handle (e.g. automatically printing sheets, we launch the recorder once, we integrate its code into our own inside a loop).

Writing macros directly is simple once you understand the philosophy of the approach, and identify the main objects and access to their properties and methods (the recorder can help us with this).

Activate (select) the workbook whose file name is ''workbook1.xlsm''

Workbooks

```
Workbooks(''workbook1.xlsm'').Activate
```

In the current workbook, activate the worksheet whose name is ''Sheet1'' (visible in the tab at the bottom of the sheet)

Sheets

```
Sheets(''Sheet1'').Activate
```

You can combine the writings.

```
Workbooks(''workbook1.xlsm'').Sheets(''Sheet1'').Activate
```

Cells

```
Cells(1,1).Value = 15
```

In the current sheet of the current workbook, insert the value 15 in the cell row n ° 1, column n ° 1 i.e. in A1, the coordinates are absolute here.

!

```
Sheets(''Sheet1'').Cells(1,1).Value = 15
```

Again, we can combine.

Sample Macros - Simulating VAT Values

Write a macro that inserts different VAT values in **B2** and retrieves the price values including VAT in **B3**.

	A	B	C	D	E
1	PHT	100		TVA (%)	Prix TTC
2	TVA (%)	30			
3	PTTC	=B1*(1+B2/100)			
4					
5					
6					

The different VAT values tested must be transcribed as and when in column D.

The corresponding Price including VAT values must be listed in column E.

```
Sub SimulationVAT()
'variables
Dim pht As Double, pttc As Double
Dim VAT As Double
Dim i As Long
'start writing values on line 2
i = 2
'retrieve the value of the PHT
pht = Cells(1,2).Value 'in B1
'vary the VAT from 10% at 30% with a step of 5%
For VAT = 10 To 30 Step 5
 'insert the value of VAT in B2
 Cells(2,2).Value = VAT
 'Retrieve the ttc price in B3
 pttc = Cells(3,2).Value
 'registration of values
 'VAT in column D
 Cells(i,4).Value = vat
 'PTTC in column E
 Cells(i,5).Value = pttc
 'move to the next line
 i = i + 1
Next VAT
End Sub
```

Note : you must be on the appropriate sheet before running the macro, otherwise the program will not know where to look **Cells(…).**

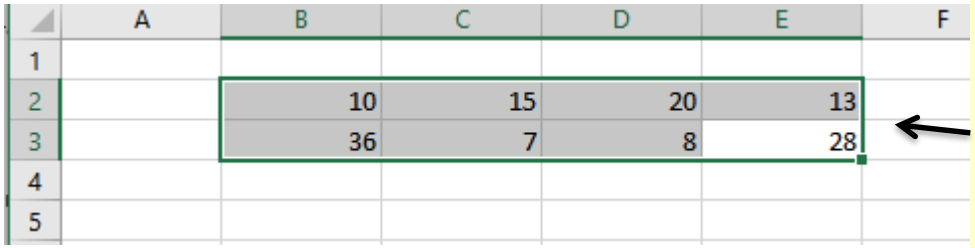
At the end of the simulation …

	A	B	C	D	E
1	PHT	100		TVA (%)	Prix TTC
2	TVA (%)	30		10	110
3	PTTC	130		15	115
4				20	120
5				25	125
6				30	130

Work on user selections

MACROS (2)

How to program a macro that directly manipulates a range of cells selected by the user? Please note, we are not in the same configuration as the custom functions here, we do not insert a result in a cell, we directly manipulate and modify the selected range.



	A	B	C	D	E	F
1						
2		10	15	20	13	
3		36	7	8	28	
4						
5						

Example: in this selection (cells must be selected before running the macro!), put cells containing an even value in green font.

```
Sub MyPairsValues()
'intermediate variable
Dim cell As Range
'loop on the selection
For Each cell In Selection
 'test the content
 If (cell.Value Mod 2 = 0) Then
 'change the font color
 cell.Font.ColorIndex = 4
 End If
Next cell
End Sub
```

Selection is an Excel object (Selection is therefore a key word). It is of the Range type (which we know well).

We can also write cell.Font.Color=vbGreen

Result…

10	15	20	13
36	7	8	28

Simple selection - We could have written ...

In a selection, again the coordinates are relative ie. the top left corner of a selection corresponds to the cell (row n ° 1, column n ° 1) whatever the position of the selection in the sheet.

!

No doubt, Selection is of type Range.

```
Sub MyValuesPairsBis()
'intermediate variables
Dim i As Long, j As Long
'loop on the lines
For i = 1 To Selection.Rows.Count
 'loop on the columns
 For j = 1 To Selection.Columns.Count
 'test the content
 If (Selection.Cells(i,j).Value Mod 2 = 0) Then
 'change the font color
 Selection.Cells(i,j).Font.ColorIndex = 4
 End If
 Next j
Next i
End Sub
```

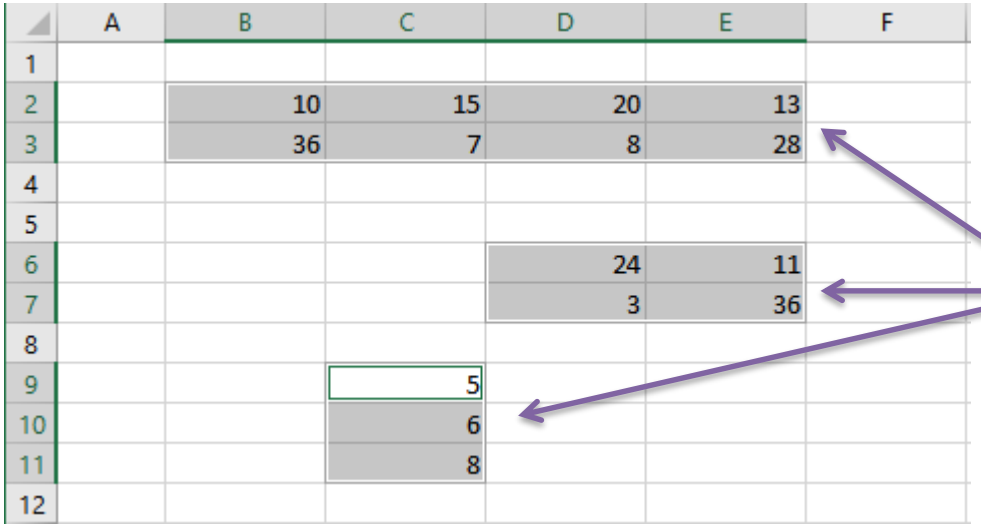
Identify the first cell containing the minimum value in a range, put its font in blue.

10	15	20	13
36	7	8	28

```
Sub MyMinBlue()
'variables intermediaries
'min will serve as a witness cell
Dim cell As Range, min As Range
'initialization of the witness on the 1st cell
Set min = Selection.Cells(1,1)
'Browse
For Each cell In Selection
 'compare with the content of the control cell
 If (cell.Value < min.Value) Then
 'update of the witness cell
 Set min = cell
 End If
Next cell
'set the color for the minimum cell
min.Font.ColorIndex = 5
End Sub
```

Range is an object. An assignment for an object variable must be made using the instruction Set.

A selection can also be multiple ie. containing several ''areas''



	A	B	C	D	E	F
1						
2		10	15	20	13	
3		36	7	8	28	
4						
5						
6				24	11	
7				3	36	
8						
9			5			
10			6			
11			8			
12						

An example of multiple selection with 3 zones.

Oddly enough, the same keyword Selection can be exploited.

```
Selection.Areas.Count
```

Number of ‘’zones” in the selection.

```
Selection.Areas(k)
```

Access to the area n ° k (which is of type Range). Areas is a collection of zones.

Multiple selection - An example

```
Sub MyMinZoneBlue()
'var. intermediaries
Dim zone As Range, min As Range, cell As Range
'for each zone
For Each zone In Selection.Areas
 'inside each zone
 'initialization
 Set min = zone.Cells(1,1)
 'cell path
 For Each cell In zone
 'compare
 If (cell.Value < min.Value) Then
 'update of the control variable
 Set min = cell
 End If
 Next cell
 'set the color for the minimum cell
 min.Font.ColorIndex = 5
'move to the next zone
Next zone
End Sub
```

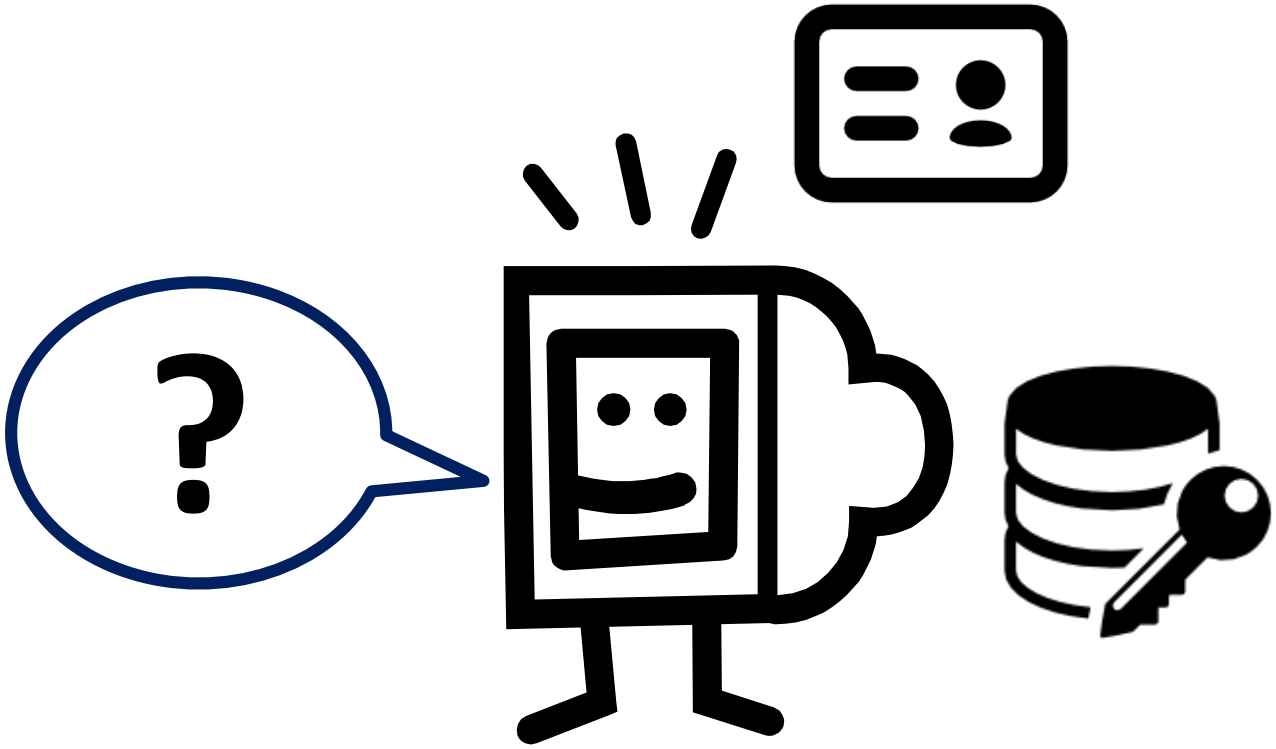
For each zone, put in blue font the cell containing the minimum value.

Selection.Areas is a collection. We can use a For Each. We could also have gone through an indexed access. Eg.

```
For k = 1 to Selection.Areas.Count
 Set zone = Selection.Areas(k)
 Etc ...
```

Result…

	A	B	C	D	E	F
1						
2		10	15	20	13	
3		36	7	8	28	
4						
5						
6				24	11	
7				3	36	
8						
9			5			
10			6			
11			8			
12						



To be continued with Arrays…

Thank you

Herve Hocquard(hocquard@labri.fr)

http://www.labri.fr/perso/hocquard/Teaching.html