

# Lesson 3

Sébastien Mathier

[www.excel-pratique.com/en](http://www.excel-pratique.com/en)

## Variables :

Variables make it possible to store all sorts of information.

Here's the first example :

```
'Display the value of the variable in a dialog box
Sub variables()
  'Declaring the variable
  Dim my_variable As Integer
  'Assigning a value to the variable
  my_variable = 12
  'Displaying the value of my_variable in a MsgBox
  MsgBox my_variable
End Sub
```

This first line of code declares the variable (this is generally placed at the beginning of the procedure).

```
Dim my_variable As Integer
```

- **Dim** : declares the variable
- **my\_variable** : the name chosen for this variable (no spaces allowed)
- **As** : declares the variable's type
- **Integer** : variable type

Declaring these variables is not absolutely necessary, but it is recommended. It makes it easier to find them, can help resolve problems, etc. In short, it's a good idea to get in the habit of declaring variables correctly.

A variable's type indicates the nature of its contents (text, numbers, date, etc.).

And then a value is given to the variable :

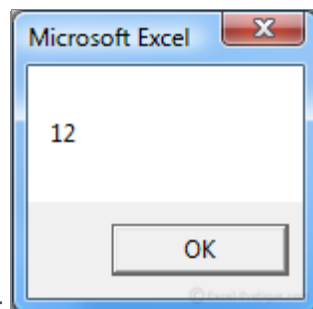
```
my_variable = 12
```

Finally, the value of the variable is displayed in a dialog box :

```
MsgBox my_variable
```

**MsgBox "value"** is the simplest way to display a value in a dialog box.

*We'll go into more detail about dialog boxes in later lessons ...*



The code will have this result :

If you don't understand what the point of using these variables is yet, don't worry, the examples introduced in the following lessons will prove their usefulness ...

## The types of variables :

Name	Type	Details	Symbol
Byte	<i>Numerical</i>	Whole number between 0 and 255.	
Integer	<i>Numerical</i>	Whole number between -32'768 and 32'767.	%
Long	<i>Numerical</i>	Whole number between - 2'147'483'648 and 2'147'483'647.	&
Currency	<i>Numerical</i>	Fixed decimal number between -922'337'203'685'477.5808 and 922'337'203'685'477.5807.	@
Single	<i>Numerical</i>	Floating decimal number between -3.402823E38 and 3.402823E38.	!
Double	<i>Numerical</i>	Floating decimal number between -1.79769313486232D308 and 1.79769313486232D308.	#
String	<i>Text</i>	Text.	\$
Date	<i>Date</i>	Date and time.	
Boolean	<i>Boolean</i>	True or False.	
Object	<i>Object</i>	Microsoft Object.	
Variant	<i>Any type</i>	Any kind of data (default type if the variable is not declared).	

Some examples with different types of variables :

```
'Example : whole number
Dim nbInteger As Integer
nbInteger = 12345

'Example : decimal number
Dim nbComma As Single
nbComma = 123.45

'Example : text
Dim varText As String
varText = "Excel-Pratique.com"

'Example : date
Dim varDate As Date
varDate = "06.04.2012"

'Example : True/False
Dim varBoolean As Boolean
varBoolean = True

'Example : object (Worksheet object for this example)
Dim varSheet As Worksheet
Set varSheet = Sheets("Sheet2") 'Set => assigning a value to an object variable

'Example of how to use object variables : activating the sheet
varSheet.Activate
```

The symbols in the table above can be used to shorten our variable declarations.

For reasons of readability, we will not be using these in the lessons, but here is an example anyway :

```
Dim example As Integer
Dim example%
```

These two lines are identical.

**Comment** : it is possible to force the declaration of variables by putting **Option Explicit** right at the beginning of a module (this way an error message will be displayed if you have forgotten to declare variables).

Practice exercise :

We will now create, step by step, a macro that retrieves a last name from cell A2, a first name from cell B2, an age from cell C2, and displays them in a dialog box.

Source file: **variable\_exercise.xls**

We'll begin by declaring the variables (all on the same line, separated by commas) :

```
Sub variables()
  'Declaring variables
  Dim last_name As String, first_name As String, age As Integer
End Sub
```

Then we assign values to the variables using **Cells** :

```
Sub variables()
  'Declaring variables
  Dim last_name As String, first_name As String, age As Integer

  'Variable values
  last_name = Cells(2, 1)
  first_name = Cells(2, 2)
  age = Cells(2, 3)
End Sub
```

Finally, we'll display the results in a dialog box, using the **&** operator to join the values (as in Excel).


```
Sub variables()
  'Declaring variables
  Dim last_name As String, first_name As String, age As Integer

  'Variable values
  last_name = Cells(2, 1)
  first_name = Cells(2, 2)
  age = Cells(2, 3)

  'Dialog box
  MsgBox last_name & " " & first_name & ", " & age & " years old"
End Sub
```

The results :

	A	B	C	D	E	F
1	<b>Last Name</b>	<b>First Name</b>	<b>Age</b>			
2	Smith	John	40		Button	
3	Smith					
4	Smith					
5	Smith					
6	Smith					
7	Smith					
8	Smith					
9	Smith					
10	Smith					
11	Smith	John	49			
12	Smith	John	50			




© Excel-Pratique.com

The next step is to display in a dialog box the row from the table that is indicated by the number in cell F5.

This is the goal :

	A	B	C	D	E	F	G
1	<b>Last Name</b>	<b>First Name</b>	<b>Age</b>				
2	Smith	John	40			Button	
3	Smith	John	41				
4	Smith	John	42				
5	Smith	John	43		No.	5	
6	Smith	John	44				
7	Smith	John	45				
8	Smith	John	46				
9	Smith	John	47				
10	Smith	John	48				
11	Smith	John	49				
12	Smith	John	50				
13	Smith	John	51				
14	Smith	John	52				
15	Smith	John	53				



© Excel-Pratique.com

Take a moment to try to solve this problem yourself before looking at the solution below ...

.....



	A	B	C	D	E	F	G
1	Last Name	First Name	Age				
2	Smith	John	40			Button	
3	Smith	John	41				
4	Smith	John	42				
5	Smith				No.	2	
6	Smith						
7	Smith						
8	Smith						
9	Smith						
10	Smith						
11	Smith						
12	Smith						
13	Smith	John	51				
14	Smith	John	52				
15	Smith	John	53				



By the way, please note that we can reduce this procedure to a single line of code :

```

Sub variables()
MsgBox Cells(Range("F5")+1,1) & " " & Cells(Range("F5")+1,2) & ", " & Cells(Range("F5")+1,3)
& " years old"
End Sub

```

Although the code will work perfectly, it is much less readable than the previous version and more difficult to rework (to make sure that our code is easy to understand, we won't be abbreviating it in this way in these lessons).

## Arrays :

While variables only allow us to store one value each, arrays make it possible to store many values and they work almost exactly the same way.

Here are some examples of declarations:

```
'Sample variable declaration
Dim var1 As String

'Sample 1 dimensional array declaration
Dim array1(4) As String

'Sample 2 dimensional array declaration
Dim array2(4, 3) As String

'Sample 3 dimensional array declaration
Dim array3(4, 3, 2) As String
```

The 1 dimensional array :

```
'Sample 1 dimensional array declaration
Dim array1(4) As String
```

There is only one number in parentheses in this declaration, which means that it is a one dimensional array. This number sets the size of the array. **array1(4)** is an array in which the cells are numbered from 0 to 4, which means that it is an array with 5 cells:

0
1
2
3
4

```
'Assigning values to the 5 cells
array1(0) = "Value of cell 0"
array1(1) = "Value of cell 1"
array1(2) = "Value of cell 2"
array1(3) = "Value of cell 3"
array1(4) = "Value of cell 4"
```

**Important** : the first cell in an array is numbered 0.

Another example, a two dimensional array :

```
'Sample declaration of a 2 dimensional array
Dim array2(4, 3) As String
```

0-0	0-1	0-2	0-3
1-0	1-1	1-2	1-3
2-0	2-1	2-2	2-3
3-0	3-1	3-2	3-3
4-0	4-1	4-2	4-3

```
'Assigning values to three colored cells
array2(0, 0) = "Red cell value"
array2(4, 1) = "Green cell value"
array2(2, 3) = "Blue cell value"
```



## Constants :

Like variables, constants can be used to store values, but the difference is that they can't be modified (thus their name).

For example, we could add a constant to avoid having to repeat a number like 6.87236476641 :

```
Sub const_example()  
    Cells(1, 1) = Cells(1, 2) * 6.87236476641  
    Cells(2, 1) = Cells(2, 2) * 6.87236476641  
    Cells(3, 1) = Cells(3, 2) * 6.87236476641  
    Cells(4, 1) = Cells(4, 2) * 6.87236476641  
    Cells(5, 1) = Cells(5, 2) * 6.87236476641  
End Sub
```

This makes the code much easier to read (important parts in particular) and makes it much easier to change the value of the constant, should you need to :

```
Sub const_example()  
    'Declaration of a constant + assignment of value  
    Const ANNUAL_RATE As Double = 6.87236476641  
  
    Cells(1, 1) = Cells(1, 2) * ANNUAL_RATE  
    Cells(2, 1) = Cells(2, 2) * ANNUAL_RATE  
    Cells(3, 1) = Cells(3, 2) * ANNUAL_RATE  
    Cells(4, 1) = Cells(4, 2) * ANNUAL_RATE  
    Cells(5, 1) = Cells(5, 2) * ANNUAL_RATE  
End Sub
```

## The scope of variables :

If a variable is declared at the beginning of a procedure (Sub), it can only be used within this same procedure. The value of the variable will not be maintained after the execution of the procedure.

```
Sub procedure1()  
    Dim var1 As Integer  
    ' => Use of a variable only within a procedure  
End Sub  
  
Sub procedure2()  
    ' => var1 cannot be used here  
End Sub
```

In order to use a variable in any of the procedures within a module, all you have to do is declare it at the beginning of the module. And if you declare a variable this way, its value will be maintained until the workbook is closed.

```
Dim var1 As Integer  
  
Sub procedure1()  
    ' => var1 can be used here  
End Sub  
  
Sub procedure2()  
    ' => var1 can also be used here  
End Sub
```

If you want to be able to use a variable in any module, on the same principle as the previous example, all you have to do is replace **Dim** with **Global** :

```
Global var1 As Integer
```

To maintain the value of a variable after the execution of the procedure in which it appears, replace **Dim** with **Static** :

```
Sub procedure1()  
    Static var1 As Integer  
End Sub
```

To maintain the values of all the variables in a procedure, add **Static** before **Sub** :

```
Static Sub procedure1()  
    Dim var1 As Integer  
End Sub
```

Create your own type of variable :

We won't spend very much time on this point. Here is an example :

```
'Creation of a variable type  
Type guests  
    last_name As String  
    first_name As String  
End Type  
  
Sub variables()  
    'Declaration  
    Dim p1 As guests  
  
    'Assigning values to p1  
    p1.last_name = "Smith"  
    p1.first_name = "John"  
  
    'Example of use  
    MsgBox p1.last_name & " " & p1.first_name  
End Sub
```