Lesson 5

Sébastien Mathier

www.excel-pratique.com/en

While :

Loops make it possible to repeat instructions a number of times, which can save a lot of time.

The following code puts sequential numbers into each of the cells in column A (from row 1 to 12) :

```
Sub while_loop()

    Cells(1, 1) = 1
    Cells(2, 1) = 2
    Cells(3, 1) = 3
    Cells(4, 1) = 4
    Cells(5, 1) = 5
    Cells(6, 1) = 6
    Cells(7, 1) = 7
    Cells(8, 1) = 8
    Cells(9, 1) = 9
    Cells(10, 1) = 10
    Cells(11, 1) = 11
    Cells(12, 1) = 12

End Sub
```

This code is very repetitive ...

Imagine if we had to number hundreds of cells instead of just 12 ... Now you understand why loops can be useful.

Here is an example of an empty **While** loop :

```
Sub while_loop()

    While [condition]
        'Instructions
    Wend

End Sub
```

As long as the condition is true, the instructions in the loop will continue to be executed (careful not to create an infinite loop).

And here is the repetitive macro introduced above, converted into a **While** loop :

```
Sub while_loop()

    Dim num As Integer
    num = 1 'Starting number (in this case, this is both the row number and the number that will be placed in each cell)

    While num <= 12 'As long as the num variable is <= 12, the instructions will loop
       Cells(num, 1) = num 'Numbering
       num = num + 1 'The number is increased by 1 each time the instructions loop
    Wend

End Sub
```

Using this loop macro, all we would have to do if we wanted to number 500 lines instead of just 12 would be to replace 12 with 500 ...

Do Loop :

This is another way to write a loop command that works the same way as **While Wend** (as long as the condition is true, the instructions contained within the While command will loop) :

```
Sub do_while_loop()

    Do While [condition]
        'Instructions
    Loop

End Sub
```

In this case, the conditions can also be placed at the end of the **Do Loop** loop, which means that the instructions will definitely be executed at least once :

```
Sub do_while_loop()

    Do
        'Instructions
    Loop While [condition]

End Sub
```

Rather than repeating the loop as long as the condition is true, it is also possible to exit the loop when the condition is true by replacing **While** with **Until** :

```
Sub do_while_loop()

    Do Until [condition]
        'Instructions
    Loop

End Sub
```

For :

```
Sub for_loop()

    For i = 1 To 5
        'Instructions
    Next

End Sub
```

The **For** loop will be repeated here 5 times.

At each repetition of the loop, the variable i is automatically incremented by 1 :

```
Sub for_loop()

    For i = 1 To 5
        MsgBox i
    Next

End Sub
```

Early exit from a loop :

It's possible to exit a **For** loop early by using the following instruction :

```
Exit For 'Exit a For loop
```

Here is an example of this :

```
Sub for_loop()
    Dim max_loops As Integer
    max_loops = Range("A1") 'In A1 : we have defined a limit to the number of repetitions

    For i = 1 To 7 'Number of loops expected : 7
       If i > max_loops Then 'If A1 is empty or contains a number < 7, decrease the number of
loops
           Exit For 'If the condition is true, we exit the For loop
       End If

       MsgBox i
    Next

End Sub
```

The other **Exit** instructions :

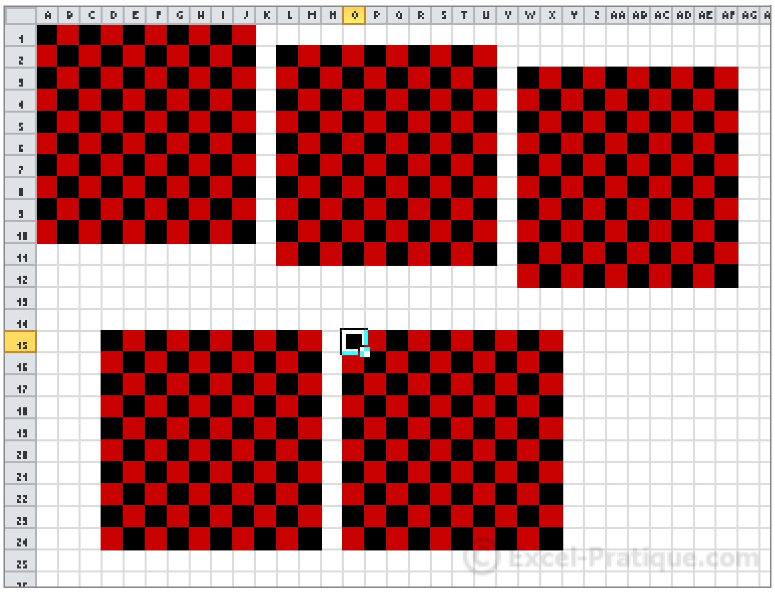
```
Exit Do 'Exit a Do Loop loop
```

```
Exit Sub 'Exit a procedure
```

```
Exit Function 'Exit a function
```

Exercise :

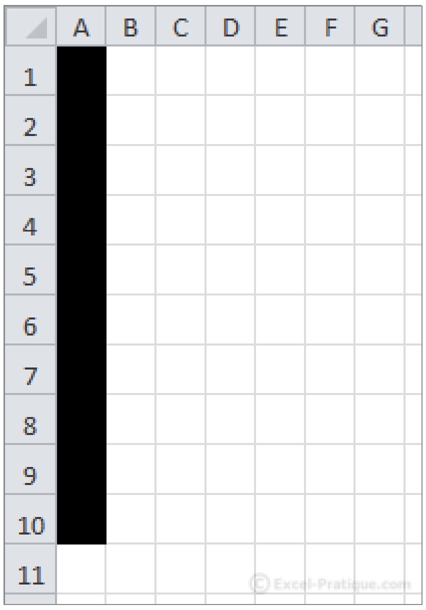
To practice what we have just learned, we'll go through the step-by-step process of creating a macro to add background colors to a 10x10 checkerboard of cells (in red and black) starting from the currently selected cell. See below :



Here's the first step of the exercise :

```
Sub loops_exercise()

    Const NB_CELLS As Integer = 10 'Number of cells to which we want to add background colors

    '...

End Sub
```

Let's start out by adding a **For** loop to add black backgrounds to the cells in column A (The NB_CELLS constant being 10). See below:



Take a moment to create this loop on your own before you look at the solution ...

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

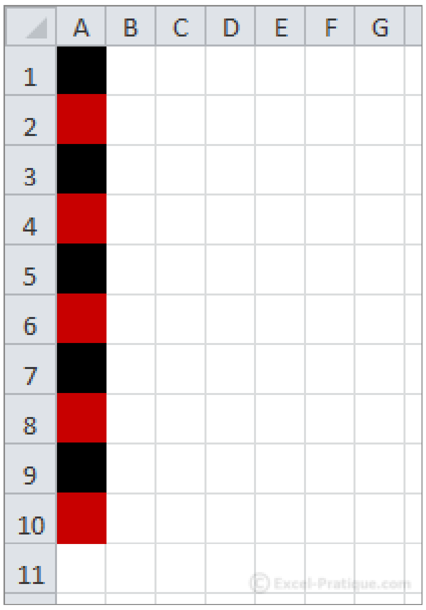
.

.

The solution :

```
Sub loops_exercise()

    Const NB_CELLS As Integer = 10 'Number of cells to which we want to add background colors

    For r = 1 To NB_CELLS 'r => row number

        Cells(r, 1).Interior.Color = RGB(0, 0, 0) 'Black

    Next

End Sub
```

The next step is making every other cell's background red with an **If** instruction (based on whether the row numbers are even or odd). See below :



	A	B	C	D	E	F	G
1	⬛						
2	🟥						
3	⬛						
4	🟥						
5	⬛						
6	🟥						
7	⬛						
8	🟥						
9	⬛						
10	🟥						
11							

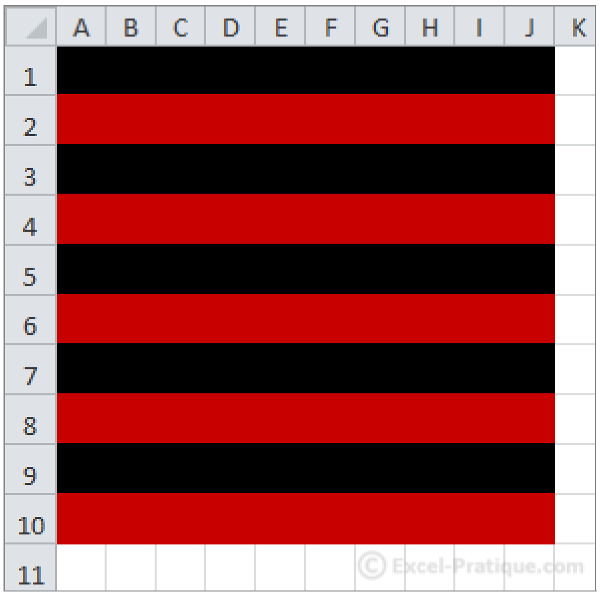
The solution :

```
Sub loops_exercise()

    Const NB_CELLS As Integer = 10 'Number of cells to which we want to add background colors

    For r = 1 To NB_CELLS 'r => row number

        If r Mod 2 = 0 Then 'Mod => is the remainder from division
            Cells(r, 1).Interior.Color = RGB(200, 0, 0) 'Red
        Else
            Cells(r, 1).Interior.Color = RGB(0, 0, 0) 'Black
        End If

    Next

End Sub
```

The condition **If r Mod 2 = 0** means : if the remained when we divide **r** by 2 equals 0 ...

Only row numbers that are even will have a remainder of 0 when they are divided by 2.

Now create a loop that executes the loop we already have for the 10 columns. See below :



.
.
.
.
.
.
.
.
.
.
.
.
.
.

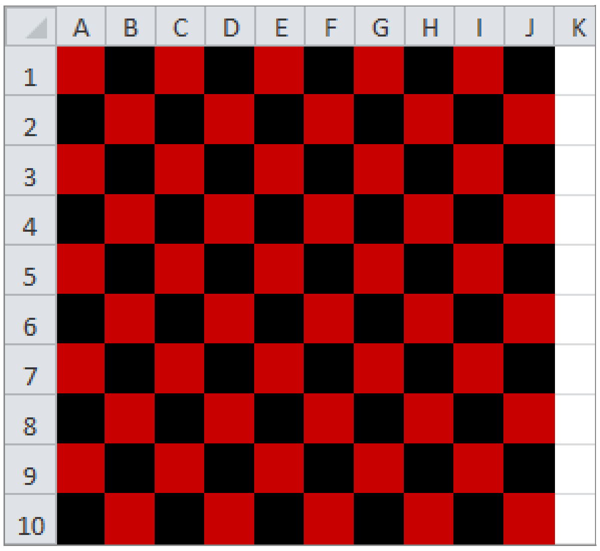
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.

The solution :

```
Sub loops_exercise()

    Const NB_CELLS As Integer = 10 '10x10 checkerboard of cells

    For r = 1 To NB_CELLS 'r => row number

        For c = 1 To NB_CELLS 'c => column number

           If r Mod 2 = 0 Then
               Cells(r, c).Interior.Color = RGB(200, 0, 0) 'Red
           Else
               Cells(r, c).Interior.Color = RGB(0, 0, 0) 'Black
           End If

        Next
    Next

End Sub
```

Now the second loop is nested within the first one.

To achieve this result ...



	A	B	C	D	E	F	G	H	I	J	K
1	🟥	⬛	🟥	⬛	🟥	⬛	🟥	⬛	🟥	⬛	
2	⬛	🟥	⬛	🟥	⬛	🟥	⬛	🟥	⬛	🟥	
3	🟥	⬛	🟥	⬛	🟥	⬛	🟥	⬛	🟥	⬛	
4	⬛	🟥	⬛	🟥	⬛	🟥	⬛	🟥	⬛	🟥	
5	🟥	⬛	🟥	⬛	🟥	⬛	🟥	⬛	🟥	⬛	
6	⬛	🟥	⬛	🟥	⬛	🟥	⬛	🟥	⬛	🟥	
7	🟥	⬛	🟥	⬛	🟥	⬛	🟥	⬛	🟥	⬛	
8	⬛	🟥	⬛	🟥	⬛	🟥	⬛	🟥	⬛	🟥	
9	🟥	⬛	🟥	⬛	🟥	⬛	🟥	⬛	🟥	⬛	
10	⬛	🟥	⬛	🟥	⬛	🟥	⬛	🟥	⬛	🟥	

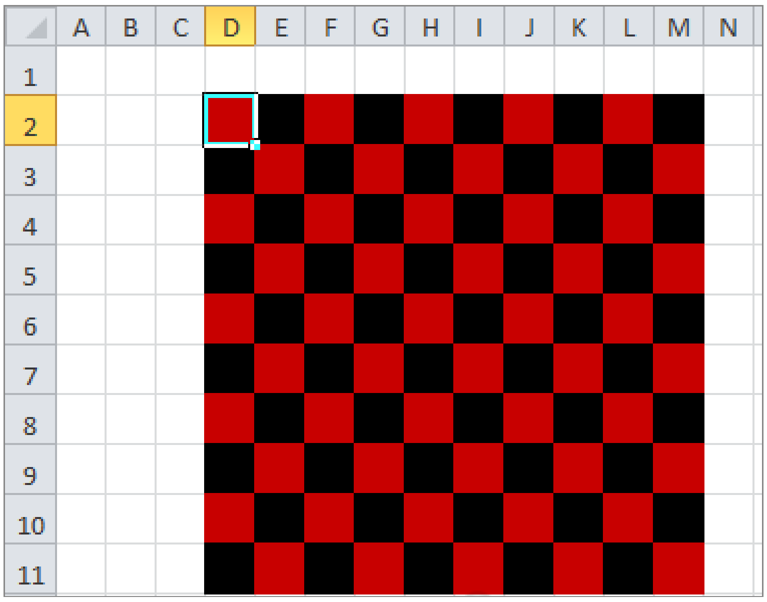
Replace :

```
If r Mod 2 = 0 Then
```

With :

```
If (r + c) Mod 2 = 0 Then
```

All that's left to do is to edit the code so that the checkerboard is created starting from the currently selected cell (rather than A1). See below :



	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1														
2				🟥	⬛	🟥	⬛	🟥	⬛	🟥	⬛	🟥	⬛	
3				⬛	🟥	⬛	🟥	⬛	🟥	⬛	🟥	⬛	🟥	
4				🟥	⬛	🟥	⬛	🟥	⬛	🟥	⬛	🟥	⬛	
5				⬛	🟥	⬛	🟥	⬛	🟥	⬛	🟥	⬛	🟥	
6				🟥	⬛	🟥	⬛	🟥	⬛	🟥	⬛	🟥	⬛	
7				⬛	🟥	⬛	🟥	⬛	🟥	⬛	🟥	⬛	🟥	
8				🟥	⬛	🟥	⬛	🟥	⬛	🟥	⬛	🟥	⬛	
9				⬛	🟥	⬛	🟥	⬛	🟥	⬛	🟥	⬛	🟥	
10				🟥	⬛	🟥	⬛	🟥	⬛	🟥	⬛	🟥	⬛	
11				⬛	🟥	⬛	🟥	⬛	🟥	⬛	🟥	⬛	🟥	

.
.
.
.
.
.
.

.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.

The solution :

```
Sub loops_exercise()

   Const NB_CELLS As Integer = 10 '10x10 checkerboard of cells
   Dim offset_row As Integer, offset_col As Integer ' => adding 2 variables

   'Shift (rows) starting from the first cell = the row number of the active cell - 1
   offset_row = ActiveCell.Row - 1
   'Shift (columns) starting from the first cell = the column number of the active cell - 1
   offset_col = ActiveCell.Column - 1

   For r = 1 To NB_CELLS 'Row number

        For c = 1 To NB_CELLS 'Column number

            If (r + c) Mod 2 = 0 Then
            'Cells(row number + number of rows to shift, column number + number of columns to
shift)
                Cells(r + offset_row, c + offset_col).Interior.Color = RGB(200, 0, 0) 'Red
            Else
                Cells(r + offset_row, c + offset_col).Interior.Color = RGB(0, 0, 0) 'Black
            End If

        Next
   Next

End Sub
```