# Excel-VBA programming

## Creare un nuovo foglio all’interno della cartella

Function CreaNuovoFoglio() As Worksheet

Dim WS As Worksheet

Set WS = Sheets.Add

WS.Name = “Nome del Foglio che sto aggiungendo”

Set CreaNuovoFoglio = WS

End Function

## Duplicare un foglio e posizionarvici

Function DuplicaModello()

Dim WS As Worksheet

Sheets(1).Copy After:=Sheets(Sheets.Count)

Set WS = ActiveSheet

WS.Name = "Nome del foglio che ho duplicato"

End Function

### Cancellare un range di celle

Range("K8:K500").ClearContents

## Formatting Cells Number

### **General**

Range("A1").NumberFormat = "General"

### **Number**

Range("A1").NumberFormat = "0.00"

### **Currency**

Range("A1").NumberFormat = "$#,##0.00"

### **Accounting**

Range("A1").NumberFormat = "\_($\* #,##0.00\_);\_($\* (#,##0.00);\_($\* ""-""??\_);\_(@\_)"

### **Date**

Range("A1").NumberFormat = "yyyy-mm-dd;@"

### **Time**

Range("A1").NumberFormat = "h:mm:ss AM/PM;@"

### **Percentage**

Range("A1").NumberFormat = "0.00%"

### **Fraction**

Range("A1").NumberFormat = "# ?/?"

### **Scientific**

Range("A1").NumberFormat = "0.00E+00"

### **Text**

Range("A1").NumberFormat = "@"

### **Special**

Range("A1").NumberFormat = "00000"

### **Custom**

Range("A1").NumberFormat = "$#,##0.00\_);[Red]($#,##0.00)"

## Formatting Cells Alignment

### **Text Alignment**

#### **Horizontal**

The value of this property can be set to one of the constants: **xlGeneral**, **xlCenter**, **xlDistributed**, **xlJustify**, **xlLeft**, **xlRight**.

The following code sets the horizontal alignment of cell A1 to center.

Range("A1").HorizontalAlignment = xlCenter

#### **Vertical**

The value of this property can be set to one of the constants: **xlBottom**, **xlCenter**, **xlDistributed**, **xlJustify**, **xlTop**.

The following code sets the vertical alignment of cell A1 to bottom.

Range("A1").VerticalAlignment = xlBottom

### **Text Control**

#### **Wrap Text**

This example formats cell A1 so that the text wraps within the cell.

Range("A1").WrapText = True

#### **Shrink To Fit**

This example causes text in row one to automatically shrink to fit in the available column width.

Rows(1).ShrinkToFit = True

#### **Merge Cells**

This example merge range A1:A4 to a large one.

Range("A1:A4").MergeCells = True

### **Right-to-left**

#### **Text direction**

The value of this property can be set to one of the constants: **xlRTL** (right-to-left), **xlLTR** (left-to-right), or **xlContext** (context).

The following code example sets the reading order of cell A1 to xlRTL (right-to-left).

Range("A1").ReadingOrder = xlRTL

### **Orientation**

The value of this property can be set to an integer value from –90 to 90 degrees or to one of the following constants: **xlDownward**, **xlHorizontal**, **xlUpward**, **xlVertical**.

The following code example sets the orientation of cell A1 to xlHorizontal.

Range("A1").Orientation = xlHorizontal

## **Font**

### **Font Name**

The value of this property can be set to one of the fonts: **Calibri**, **Times new Roman**, **Arial**...

The following code sets the font name of range A1:A5 to Calibri.

Range("A1:A5").Font.Name = "Calibri"

### **Font Style**

The value of this property can be set to one of the constants: **Regular**, **Bold**, **Italic**, **Bold Italic**.

The following code sets the font style of range A1:A5 to Italic.

Range("A1:A5").Font.FontStyle = "Italic"

### **Font Size**

The value of this property can be set to an integer value from 1 to 409.

The following code sets the font size of cell A1 to 14.

Range("A1").Font.Size = 14

### **Underline**

The value of this property can be set to one of the constants: **xlUnderlineStyleNone**, **xlUnderlineStyleSingle**, **xlUnderlineStyleDouble**, **xlUnderlineStyleSingleAccounting**, **xlUnderlineStyleDoubleAccounting**.

The following code sets the font of cell A1 to xlUnderlineStyleDouble (double underline).

Range("A1").Font.Underline = xlUnderlineStyleDouble

### **Font Color**

The value of this property can be set to one of the standard colors: **vbBlack**, **vbRed**, **vbGreen**, **vbYellow**, **vbBlue**, **vbMagenta**, **vbCyan**, **vbWhite** or an integer value from 0 to 16,581,375.

To assist you with specifying the color of anything, the VBA is equipped with a function named RGB. Its syntax is:

Function RGB(RedValue As Byte, GreenValue As Byte, BlueValue As Byte) As long

This function takes three arguments and each must hold a value between 0 and 255. The first argument represents the ratio of red of the color. The second argument represents the green ratio of the color. The last argument represents the blue of the color. After the function has been called, it produces a number whose maximum value can be 255 \* 255 \* 255 = 16,581,375, which represents a color.

The following code sets the font color of cell A1 to vbBlack (Black).

Range("A1").Font.Color = vbBlack

The following code sets the font color of cell A1 to 0 (Black).

Range("A1").Font.Color = 0

The following code sets the font color of cell A1 to RGB(0, 0, 0) (Black).

Range("A1").Font.Color = RGB(0, 0, 0)

### **Font Effects**

#### **Strikethrough**

True if the font is struck through with a horizontal line.

The following code sets the font of cell A1 to strikethrough.

Range("A1").Font.Strikethrough = True

#### **Subscript**

True if the font is formatted as subscript. False by default.

The following code sets the font of cell A1 to Subscript.

Range("A1").Font.Subscript = True

#### **Superscript**

True if the font is formatted as superscript; False by default.

The following code sets the font of cell A1 to Superscript.

Range("A1").Font.Superscript = True

## **Border**

### **Border Index**

Using VBA you can choose to create borders for the different edges of a range of cells:

1. **xlDiagonalDown** (Border running from the upper left-hand corner to the lower right of each cell in the range).
2. **xlDiagonalUp** (Border running from the lower left-hand corner to the upper right of each cell in the range).
3. **xlEdgeBottom** (Border at the bottom of the range).
4. **xlEdgeLeft** (Border at the left-hand edge of the range).
5. **xlEdgeRight** (Border at the right-hand edge of the range).
6. **xlEdgeTop** (Border at the top of the range).
7. **xlInsideHorizontal** (Horizontal borders for all cells in the range except borders on the outside of the range).
8. **xlInsideVertical** (Vertical borders for all the cells in the range except borders on the outside of the range).

### **Line Style**

The value of this property can be set to one of the constants: **xlContinuous** (Continuous line), **xlDash** (Dashed line), **xlDashDot** (Alternating dashes and dots), **xlDashDotDot** (Dash followed by two dots), **xlDot** (Dotted line), **xlDouble** (Double line), **xlLineStyleNone** (No line), **xlSlantDashDot** (Slanted dashes).

The following code example sets the border on the bottom edge of cell A1 with continuous line.

Range("A1").Borders(xlEdgeBottom).LineStyle = xlContinuous

The following code example removes the border on the bottom edge of cell A1.

Range("A1").Borders(xlEdgeBottom).LineStyle = xlNone

### **Line Thickness**

The value of this property can be set to one of the constants: **xlHairline** (Hairline, thinnest border), **xlMedium** (Medium), **xlThick** (Thick, widest border), **xlThin** (Thin).

The following code example sets the thickness of the border created to xlThin (Thin).

Range("A1").Borders(xlEdgeBottom).Weight = xlThin

### **Line Color**

The value of this property can be set to one of the standard colors: **vbBlack**, **vbRed**, **vbGreen**, **vbYellow**, **vbBlue**, **vbMagenta**, **vbCyan**, **vbWhite** or an integer value from 0 to 16,581,375.

The following code example sets the color of the border on the bottom edge to green.

Range("A1").Borders(xlEdgeBottom).Color = vbGreen

You can also use the RGB function to create a color value.

The following example sets the color of the bottom border of cell A1 with RGB fuction.

Range("A1").Borders(xlEdgeBottom).Color = RGB(255, 0, 0)

## **Fill**

### **Pattern Style**

The value of this property can be set to one of the constants:

1. **xlPatternAutomatic** (Excel controls the pattern.)
2. **xlPatternChecker** (Checkerboard.)
3. **xlPatternCrissCross** (Criss-cross lines.)
4. **xlPatternDown** (Dark diagonal lines running from the upper left to the lower right.)
5. **xlPatternGray16** (16% gray.)
6. **xlPatternGray25** (25% gray.)
7. **xlPatternGray50** (50% gray.)
8. **xlPatternGray75** (75% gray.)
9. **xlPatternGray8** (8% gray.)
10. **xlPatternGrid** (Grid.)
11. **xlPatternHorizontal** (Dark horizontal lines.)
12. **xlPatternLightDown** (Light diagonal lines running from the upper left to the lower right.)
13. **xlPatternLightHorizontal** (Light horizontal lines.)
14. **xlPatternLightUp** (Light diagonal lines running from the lower left to the upper right.)
15. **xlPatternLightVertical** (Light vertical bars.)
16. **xlPatternNone** (No pattern.)
17. **xlPatternSemiGray75** (75% dark moiré.)
18. **xlPatternSolid** (Solid color.)
19. **xlPatternUp** (Dark diagonal lines running from the lower left to the upper right.)

## **Protection**

### **Locking Cells**

This property returns **True** if the object is locked, **False** if the object can be modified when the sheet is protected, or **Null** if the specified range contains both locked and unlocked cells.

The following code example unlocks cells A1:B22 on Sheet1 so that they can be modified when the sheet is protected.

Worksheets("Sheet1").Range("A1:B22").Locked = False

Worksheets("Sheet1").Protect

### **Hiding Formulas**

This property returns **True** if the formula will be hidden when the worksheet is protected, **Null** if the specified range contains some cells with FormulaHidden equal to **True** and some cells with FormulaHidden equal to **False**.

Don’t confuse this property with the Hidden property. The formula will not be hidden if the workbook is protected and the worksheet is not, but only if the worksheet is protected.

The following code example hides the formulas in cells A1 and C1 on Sheet1 when the worksheet is protected.

Worksheets("Sheet1").Range("A1:C1").FormulaHidden = True