



# La Gerarchia del linguaggio VBA

*Corso di formazione su Excel VBA base*



[www.excel8020.it](http://www.excel8020.it)  
Professional Excel and Microsoft BI Hub

Microsoft®  
**Visual Basic®**  
for Applications



## Le principali componenti del linguaggio VBA: “Classi, Oggetti, Properties, Methods e Functions”

La comprensione delle componenti del linguaggio VBA è molto complessa. Ai fini del corso base questa trattazione non vuole essere esaustiva, l'obiettivo è consolidare le basi per comprendere al meglio le sessioni successive e dare spazio ad eventuali successivi approfondimenti

La gerarchia del linguaggio VBA è navigabile attraverso l'“Object Browser”

**VBA Editor → Visualizza → Visualizzatore Oggetti**



## Classi VBA:

**“Le Classi VBA sono la definizione astratta di tutti i possibili oggetti istanziabili, Property o Methods utilizzabili”**

### Librerie VBA

Possono essere istanziati differenti librerie (Excel, VBA, Office..) ognuna delle quali definisce le proprie “Classi”

### Classi-Membri

Questa è una prima suddivisione gerarchica: ad ogni Classe appartengono differenti “Membri” che la definiscono completamente

### Properties e Methods

In genere i membri di una certa Classe si suddividono in “Properties” e “Methods”



## Classi VBA:

**“Le Classi VBA sono la definizione astratta di tutti i possibili oggetti istanziabili , Property o Methods utilizzabili”**

### Oggetti

Un oggetto è l'istanza di una certa “Classe”. L' Oggetto quindi appartiene ad una certa Classe che ne definisce la struttura

### Classi Personalizzate

E' possibile definire “Classi personalizzate” a seconda delle esigenze di progetto. Le “Classi Personalizzate” sono oggetto di corsi avanzati



## Classi VBA:

**“Le Classi VBA sono la definizione astratta di tutti i possibili Oggetti istanziabili , Property o Methods utilizzabili”**

Ad ogni Oggetto istanziabile, ad ogni Property o Method utilizzabile corrisponde una determinata Classe “astratta” tipicamente di tipo “Module” ma non solo, ognuna caratterizzata da propri “Membri”



**VBA Properties:**  
“attributi, caratteristiche o qualità con le quali si può descrivere o modificare un oggetto”

## Properties Read-Only

Possono essere solo lette per determinare le qualità di un oggetto ed eventualmente compiere una certa azione (tramite es. un if statement)

## Property Read-Write

Possono essere anche modificate per cambiare le qualità o caratteristiche di un oggetto

Si accede alle properties mediante

**<Object name>.<Property name>**



# Excel VBA: la gerarchia del linguaggio

- Esempio di Read/Write Properties:
  - **Range("C5").value** *(Il valore di una cella è leggibile e modificabile)*
  - oppure **Range("C5").Font.Size** *( la dimensione di un font è leggibile e modificabile)*
  
- Esempio di Read-Only Properties:
  - **Range("A1:C3").count** *(ci sono 9 celle in questo range)*
  - **Worksheets("Sheet1").Range("F3").Column** *(la colonna F è la sesta)*



- Alcune Properties sono utilizzate per “ritornare oggetti”, cioè per accedere ad alcuni oggetti.

Esempio:

- ✓ La Property **Range.Interior** (ReadOnly) può essere utilizzata per accedere all'oggetto generico **Interior** ( specificato per. es. da **Range(“C5”).Interior** e modificarne il valore tramite **ColorIndex**.
- ✓ Come vedremo nel seguito **ColorIndex** è una property al cui valore (tra 1 e 56) corrisponde un determinato colore





```
If ThisWorkbook.Worksheets("Foglio1").Range("C5").Value = 12 Then  
    ThisWorkbook.Worksheets("Foglio2").Range("G5").Interior.ColorIndex = 24  
End If
```

- ThisWorkbook:*** rappresenta, tra tutti i file Excel, quello correntemente aperto e attivo
- Worksheets("Foglio1"):*** rappresenta, tra tutti i fogli ("Sheets") dei possibili file Excel (Workbooks) quello chiamato "Foglio1"
- Range("C5"):*** rappresenta la cella "C5" ovvero l'oggetto "Range("C5")
- Value:*** property Read/Write che rappresenta il valore della cella
- Interior:*** oggetto che rappresenta l'interno della cella : Range("C5").Interior è l'int. della cella
- ColorIndex:*** Property che rappresenta il colore secondo la codifica "ColorIndex"
- =:*** Rappresenta l'azione di "assegnazione"
- 24:*** Rappresenta il codice colore nella codifica "ColorIndex"



## Property indicizzate:

“ Sono delle properties particolari che permettono di cambiare le caratteristiche complesse di un certo oggetto, caratteristiche che possono essere modificate accedendo a dei valori preimpostati”

Esempio: l'oggetto **Range.Borders**

E' possibile accedere a questo oggetto (per modificare i bordi di una cella) attraverso la property indicizzata `Range.Borders()`.

Attraverso la parentesi vengono specificati alcuni valori principali come **`xlEdgeBottom`**, **`xlEdgeTop`**, **`xlEdgeLeft`** e **`xlEdgeRight`** che specificano quale bordo si intende modificare.

Il bordo è anche caratterizzato da un'ulteriore property: **`LineStyle`** che caratterizza la linea del bordo stesso (**`xlContinuous`**, **`xlDash`**, **`xlDot...`**)



# Excel VBA: la gerarchia del linguaggio

*Le property indicizzate: l'oggetto Range.Borders*

- **Esempio :**

**Range("C2").Borders(xlEdgeLeft).LineStyle = xlContinuous**

**Range("C2").Borders** : oggetto "bordo"

**Range("C2").Borders(xlEdgeLeft)**: Property indicizzata che definisce il bordo sx

**Range("C2").Borders(xlEdgeLeft).LineStyle**: property Read/Write, ma "writable" attraverso un set di valori preimpostati (vedere nota slide successiva)

**Range("C2").Borders(xlEdgeLeft).LineStyle = xlContinuous**: definisce il bordo sinistro della cella C2 a linea continua



# Excel VBA: la gerarchia del linguaggio

## ▪ Nota

`Range("C2").Borders(xlEdgeLeft).LineStyle = xlContinuous`

Quali valori può assumere **"LineStyle"**?

→ *Intellisense da informazioni su come accedere al bordo della cella (**xlEdgeLeft**, **xlEdgeRight**...) ma non su quali valori può assumere **"LineStyle"**.*

→ *Per accedere a queste informazioni digitare **"xlLineStyle"** nella finestra di ricerca dell'Object Browser, si accederà alla classe **"XlLineStyle"** ( **Classe Built-In Keywords**) i cui membri sono i valori utilizzabili.*



**VBA Methods:**  
“l’insieme delle azioni che possono essere compiute con un oggetto ( o dall’oggetto o sull’oggetto)”

## Method action1

Utilizzare l’oggetto per un qualche scopo definito

## Method action2

Modificare le Properties dell’oggetto

Si accede ai Methods mediante

**<Object name>.<Object Method>**



# Excel VBA: la gerarchia del linguaggio

- I Methods possono avere “argomenti” ( o parametri)
  - Solitamente i parametri definiscono l'azione che si intende compiere su un oggetto
  - I parametri possono essere obbligatori o opzionali
  
- Ci sono due modalità per accedere ai parametri di un Method:
  - a. Per posizione: specifica del parametro in base alla posizione all'interno della chiamata
  - b. Per riferimento: specifica del parametro in base al nome



## Per Posizione

<Object Name>.<Object Method>, par1, par2,.....par N

- Qualora un parametro non venisse specificato, è necessario comunque inserire una “,” nella posizione opportuna (placeholder)

## Per Riferimento

<Object Method> par1Name := par1Value, par2Name := par2Value ...

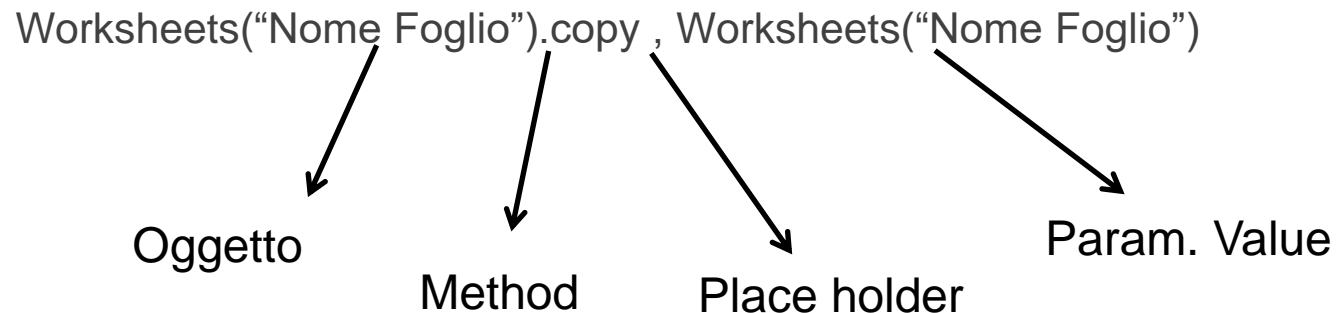
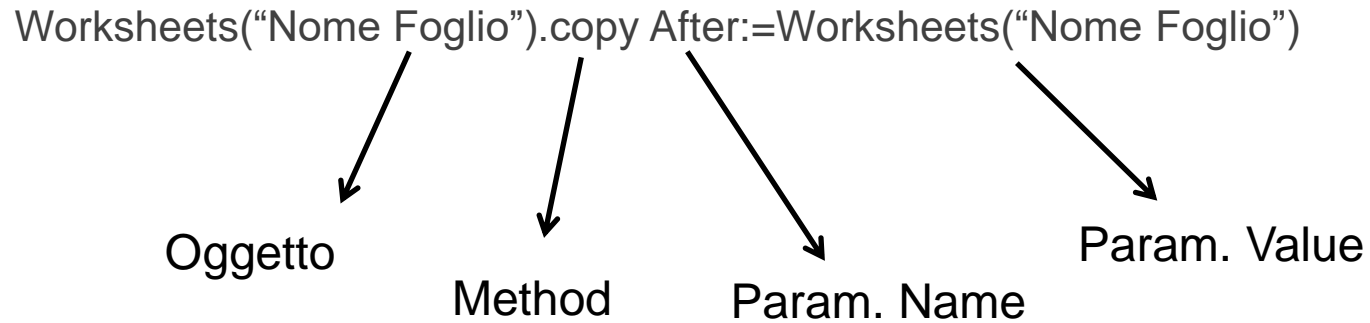
- Se un parametro non viene specificato, non è necessario nessun placeholder
- Accedere con questa modalità in genere garantisce una maggiore leggibilità del codice sviluppato



# Excel VBA: la gerarchia del linguaggio

VBA Methods: esempio

sintassi: `copy([Before],[After])`





## Ricapitolando

- Gli Oggetti sono il fondamento del linguaggio VBA, ogni cosa in VBA riguarda la manipolazione di oggetti
  - ✓ Un oggetto è parte di una “Collection” di oggetti simili, tutti dello stesso tipo. Le collection sono oggetti a loro volta.
  - ✓ Una property è un attributo di un oggetto e ne rappresenta le caratteristiche. Per cambiare le caratteristiche di un oggetto se ne modificano li valori delle Properties
  - ✓ Un Method rappresenta un’azione che possiamo compiere con un oggetto o su di un oggetto, e spesso i Methods hanno argomenti che specificano le modalità proprie di una certa azione. (es: `ActiveSheet.PrintOut`)
  - ✓ Spesso dunque i Method sono **azioni** e le Properties **qualità**. Utilizzare un Method significa fare qualcosa con l’oggetto, utilizzarne una property significa aver letto o modificato una caratteristica dell’oggetto.



## VBA Functions e VBA Routines: cenni

### VBA Routines

Un insieme di righe di codice racchiuso tra le Keywords  
Sub <nome subroutine> ... End Sub

### VBA Functions

Un insieme di righe di codice racchiuso tra le Keywords  
Func <nome function> ... End Function

Le “Functions” non sono oggetto di questo corso



## VBA Functions vs VBA Routines:

### Routines

- ✓ Non possono ritornare valori
- ✓ Possono essere richiamate in seguito ad eventi

### Functions

- ✓ Possono ritornare valori
- ✓ Possono essere chiamate solo da un'altra Sub/Function

Excel 2016 standard ha circa 400 “Built-in Functions”, con il VBA il set cresce fino a circa 600 e oltre



## Come accedere al valore di ritorno di una funzione ?

- Ai fini di questo corso è importante capire che ci sono funzioni che ritornano valori e altre che possono ritornare valori o eseguire azioni.

→ Se siamo interessati al valore di ritorno di una funzione, o se una funzione ritorna solo valori (come le funzioni matematiche) allora è necessario assegnare il risultato ad una variabile e racchiudere gli argomenti in parentesi tonde.

**var =function(arg1,arg2,...,argN)**

→ Se invece non siamo interessati al valore di una funzione ma solo ad una azione specifica eseguita dalla funzione, allora NON si usano le parentesi, non si definisce la variabile e i parametri vengono listati per posizione dopo il nome della funzione

**function arg1,arg2,...,argN**



## Come accedere al valore di ritorno di una funzione ?

- Esempi

La funzione matematica di arrotondamento “**Round**”

**Round(Number,[NumDigitsAfterDecimal])**

arrotonda un numero al numero decimali indicati

***Num = Round(1.55,1)***

In questo caso è stata definita implicitamente una variabile Num e assegnata al risultato della funzione Round.

Num vale 1,6

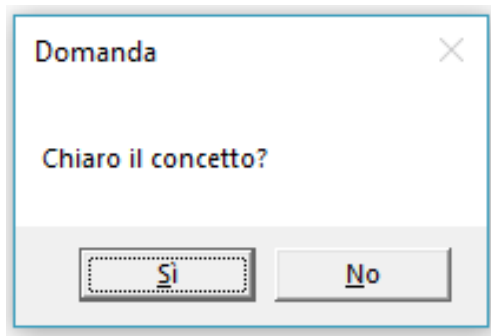


## Come accedere al valore di ritorno di una funzione ?

- Esempi

**MsgBox** è una funzione di interfaccia che può semplicemente eseguire un azione (inoltrando un semplice messaggio all'utente tramite un icona) oppure ritornare un valore a seconda di un tasto di scelta

***Risposta = MsgBox("E' chiaro il concetto?", vbYesNo, "Domanda")***



La funzione ritorna "6" o "7" a seconda del tasto cliccato.

"Risposta" vale dunque "6" o "7" e a seguito di ciò si possono intraprendere differenti azioni

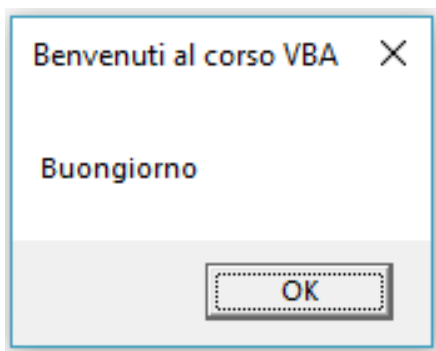


## Come accedere al valore di ritorno di una funzione ?

- Esempi

**MsgBox** può anche inoltrare un semplice messaggio, e quindi l'utente non è interessato ad alcun valore di ritorno

*MsgBox "Buongiorno", , "Benvenuti al corso VBA"*



I parametri vengono listati per posizione, è necessario un “placeholder” in seconda posizione, non siamo interessati ad un valore di ritorno e non viene istanziata nessuna variabile

