Common VBA Statements for Excel VBA Programming

The following table provides a list of commonly used VBA statements that you might use when creating macros for Excel. For more information on a particular statement, consult Excel's Help system.

Statement	What It Does
AppActivate	Activates an application window
Beep	Sounds a tone via the computer's speaker
Call	Transfers control to another procedure
ChDir	Changes the current directory
ChDrive	Changes the current drive
Close	Closes a text file
Const	Declares a constant value
Date	Sets the current system date
Declare	Declares a reference to an external procedure in a Dynamic Link Library (DLL)
DeleteSetting	Deletes a section or key setting from an application's entry in the Windows Registry
Dim	Declares variables and (optionally) their data types
Do-Loop	Loops through a set of instructions
End	Used by itself, exits the program; also used to end a block of statements that begin with If, With, Sub, Function, Property, Type, or Select
Erase	Re-initializes an array
Error	Simulates a specific error condition
Exit Do	Exits a block of Do-Loop code
Exit For	Exits a block of For-Next code
Exit Function	Exits a Function procedure
Exit Property	Exits a property procedure
Exit Sub	Exits a subroutine procedure
FileCopy	Copies a file
For Each-Next	Loops through a set of instructions for each member of a series
For-Next	Loops through a set of instructions a specific number of times
Function	Declares the name and arguments for a Function procedure
Get	Reads data from a text file
GoSub...Return	Branches to and returns from a procedure
GoTo	Branches to a specified statement within a procedure
If-Then-Else	Processes statements conditionally
Input #	Reads data from a sequential text file
Kill	Deletes a file from a disk
Let	Assigns the value of an expression to a variable or property
Line Input #	Reads a line of data from a sequential text file
Load	Loads an object but doesn't show it
Lock...Unlock	Controls access to a text file
Mid	Replaces characters in a string with other characters
MkDir	Creates a new directory
Name	Renames a file or directory
On Error	Gives specific instructions for what to do in the case of an error
On...GoSub	Branches, based on a condition
On...GoTo	Branches, based on a condition
Open	Opens a text file
Option Base	Changes the default lower limit for arrays
Option Compare	Declares the default comparison mode when comparing strings
Option Explicit	Forces declaration of all variables in a module
Option Private	Indicates that an entire module is Private
Print #	Writes data to a sequential file

Private	Declares a local array or variable
Property Get	Declares the name and arguments of a Property Get procedure
Property Let	Declares the name and arguments of a Property Let procedure
Property Set	Declares the name and arguments of a Property Set procedure
Public	Declares a public array or variable
Put	Writes a variable to a text file
RaiseEvent	Fires a user-defined event
Randomize	Initializes the random number generator
ReDim	Changes the dimensions of an array
Rem	Specifies a line of comments (same as an apostrophe [''])
Reset	Closes all open text files
Resume	Resumes execution when an error-handling routine finishes
RmDir	Removes an empty directory
SaveSetting	Saves or creates an application entry in the Windows Registry
Seek	Sets the position for the next access in a text file
Select Case	Processes statements conditionally
SendKeys	Sends keystrokes to the active window
Set	Assigns an object reference to a variable or property
SetAttr	Changes attribute information for a file
Static	Declares variables at the procedure level so that the variables retain their values as long as the code is running
Stop	Pauses the program
Sub	Declares the name and arguments of a Sub procedure
Time	Sets the system time
Type	Defines a custom data type
Unload	Removes an object from memory
While...Wend	Loops through a set of instructions as long as a certain condition remains true
Width #	Sets the output line width of a text file
With	Sets a series of properties for an object
Write #	Writes data to a sequential text file

VBA Functions for Excel VBA Programming

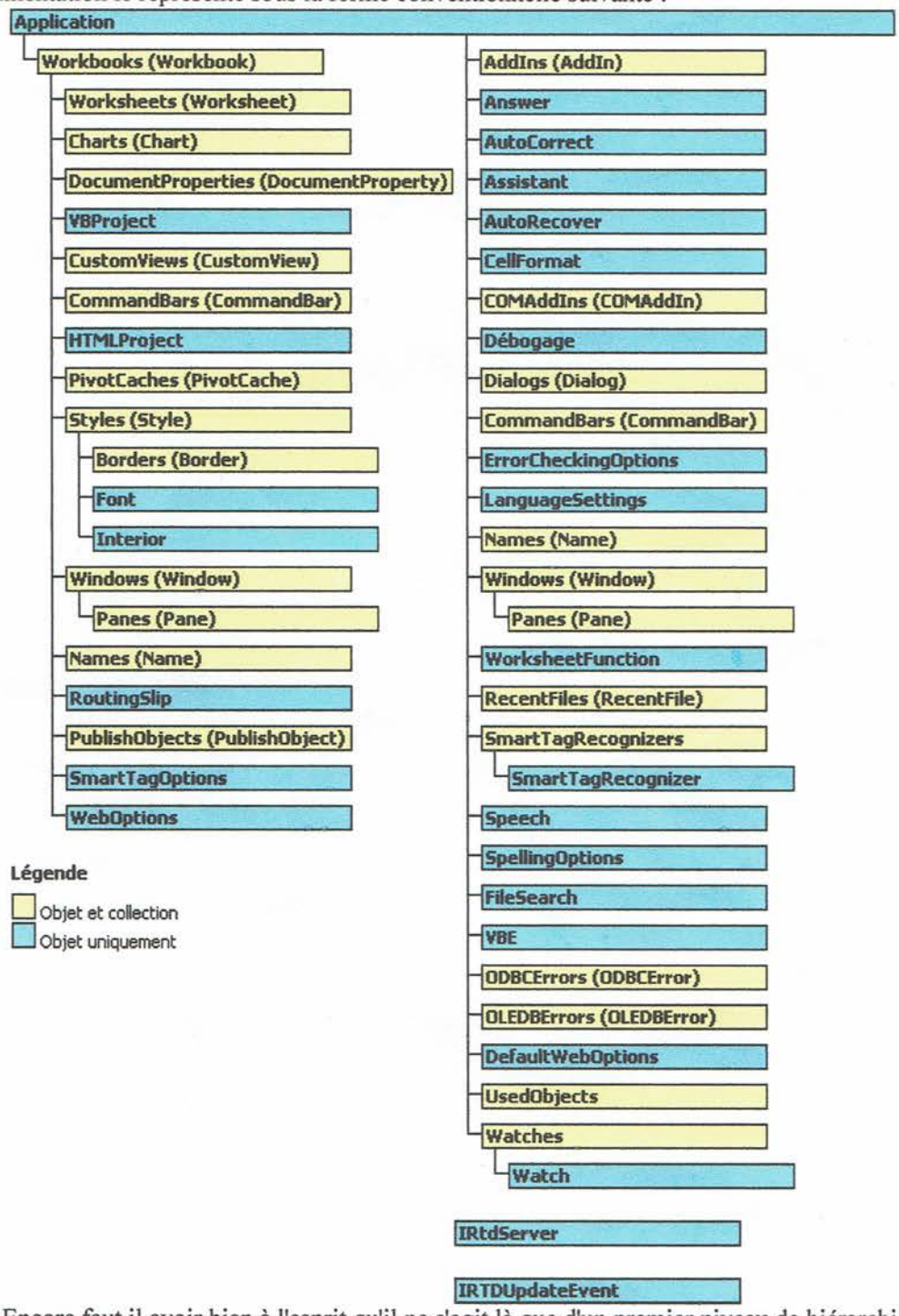
The VBA language contains a number of functions that you can use to build code in Excel. The following table provides descriptions of the most useful functions. When writing code, type the word **VBA** followed by a period, and you'll see a drop-drop list of these functions. See Excel's Help system for more details.

Function	What It Does
Abs	Returns the absolute value of a number
Array	Returns a variant that contains an array
Asc	Converts the first character of string to its ASCII value
Atn	Returns the arctangent of a number
CBool	Converts an expression to boolean
CByte	Converts an expression to byte data type
CCur	Converts an expression to currency data type
CDate	Converts an expression to date data type
CDbl	Converts an expression to double data type
CDec	Converts an expression to decimal data type
Choose	Selects and returns a value from a list of arguments
Chr	Converts an ANSI value to a string
CInt	Converts an expression to integer data type
CLng	Converts an expression to long data type
Cos	Returns the cosine of a number
CreateObject	Creates an OLE Automation object
CSng	Converts an expression to single data type
CStr	Converts an expression to string data type
CurDir	Returns the current path
CVar	Converts an expression to variant data type
CVDate	Converts an expression to date data type
CVErr	Returns a user-defined error number
Date	Returns the current system date
DateAdd	Returns a date with a specific date interval added to it
DateDiff	Returns a date with a specific date interval subtracted from it
DatePart	Returns an integer containing a specific part of a date
DateSerial	Converts a date to a serial number
DateValue	Converts a string to date
Day	Returns the day of the month of a date
Dir	Returns the name of a file or directory that matches a pattern
DoEvents	Yields execution so the operating system can process other events
EOF	Returns True if the end of a text file has been reached
Error	Returns the error message that corresponds to an error number
Exp	Returns the base of the natural logarithms (*e*) raised to a power
FileAttr	Returns the file mode for a text file
FileDateTime	Returns the date and time when a file was last modified
FileLen	Returns the number of bytes in a file
Fix	Returns the integer portion of a number
Format	Displays an expression in a particular format
Format Currency	Returns a number as a string, formatted as currency
FormatDateTime	Returns a number as a string, formatted as a date and/or time
Format Number	Returns a number as a formatted string
Format Percent	Returns a number as a string, formatted as a percentage
FreeFile	Returns the next file number available for use by the Open statement
GetAll	Returns a list of key settings and their values (originally created with SaveSetting) from an application's entry in the Windows registry
GetAttr	Returns a code representing a file attribute
GetObject	Retrieves an OLE Automation object from a file
GetSetting	Returns a key setting value from an application's entry in the Windows registry
Hex	Converts from decimal to hexadecimal
Hour	Returns the hour of a time
IIf	Returns one of two parts, depending on the evaluation of an expression
Input	Returns a specific number of characters from an open text file
InputBox	Displays a box to prompt a user for input
InStr	Returns the position of a string within another string
InStrRev	Returns the position of a string within another string, beginning at the back end of the string
Int	Returns the integer portion of a number

IsArray	Returns True if a variable is an array
IsDate	Returns True if a variable is a date
IsEmpty	Returns True if a variable has been initialized
IsError	Returns True if an expression is an error value
IsMissing	Returns True if an optional argument was not passed to a Procedure
IsNull	Returns True if an expression contains no valid data
IsNumeric	Returns True if an expression can be evaluated as a number
IsObject	Returns True if an expression references an OLE Automation object
Join	Returns a string created by joining a number of substrings contained in an array
LBound	Returns the lower bound of an array
LCase	Returns a string converted to lowercase
Left	Returns a specified number of characters from the left of a string
Len	Returns the length of a string, in characters
Loc	Returns the current read or write position of a text file
LOF	Returns the number of bytes in an open text file
Log	Returns the natural logarithm of a number
LTrim	Returns a copy of a string with no leading spaces
Mid	Returns a specified number of characters from a string
MidB	Returns a specified number of bytes from a string
Minute	Returns the minute of a time
Month	Returns the month of a date
MonthName	Returns a string indicating the specified month
MsgBox	Displays a modal message box
Now	Returns the current system date and time
Oct	Converts from decimal to octal
Replace	Returns a string in which one substring is replaced with another
RGB	Returns a number representing an RGB color value
Space	Returns a string with a specified number of spaces
Split	Returns an array consisting of a number of substrings
Sqr	Returns the square root of a number
Str	Returns a string representation of a number
Right	Returns a specified number of characters from the right of a string
Rnd	Returns a random number between 0 and 1
Round	Rounds a number to a specific number of decimal places
RTrim	Returns a copy of a string with no trailing spaces
Second	Returns the second of a time
Seek	Returns the current position in a text file
Sgn	Returns an integer that indicates the sign of a number
Shell	Runs an executable program
Sin	Returns the sin of a number
StrComp	Returns a value indicating the result of a string comparison
StrConv	Returns a string variant converted as specified
String	Returns a repeating character or string
StrReverse	Reverses the character order of a string
Switch	Evaluates a list of expressions and returns a value associated with the first expression in the list that is True
Tab	Positions output in an output stream
Tan	Returns the tangent of a number
Time	Returns the current system time
Timer	Returns the number of seconds since midnight
TimeSerial	Returns the time for a specified hour, minute, and second
TimeValue	Converts a string to a time serial number
Trim	Returns a string containing a copy of a specified string without leading spaces and trailing spaces
TypeName	Returns a string that describes the data type of a variable
UBound	Returns the upper bound of an array
UCase	Converts a string to uppercase
Val	Returns the numbers contained in a string
VarType	Returns a value indicating the subtype of a variable
Weekday	Returns a number representing a day of the week
Weekday Name	Returns a string indicating the specified weekday
Year	Returns the year of a date

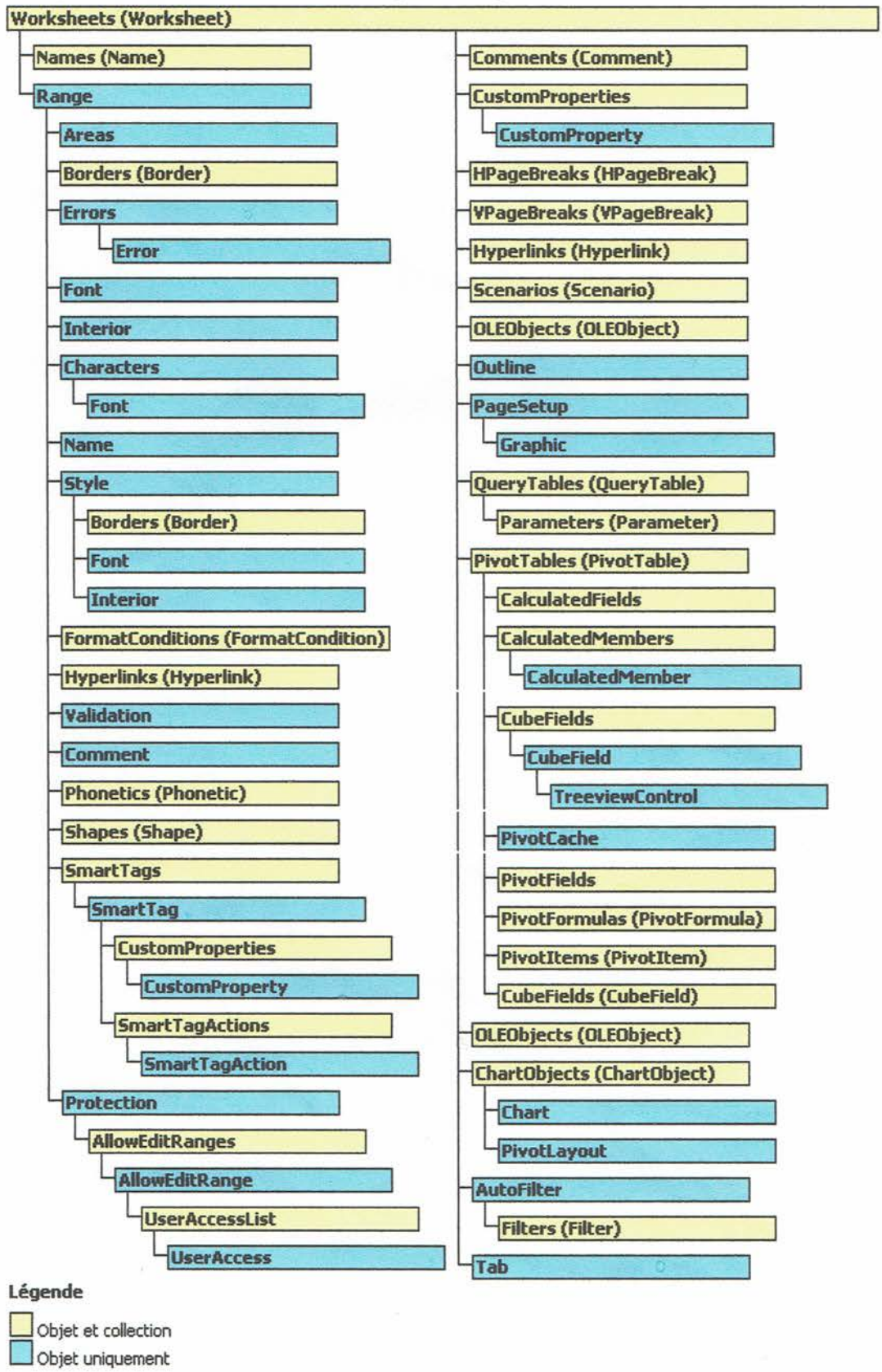
Présentation

Le modèle objet Excel est un monstre énorme lorsqu'on le considère dans son ensemble. La documentation le représente sous la forme conventionnelle suivante :

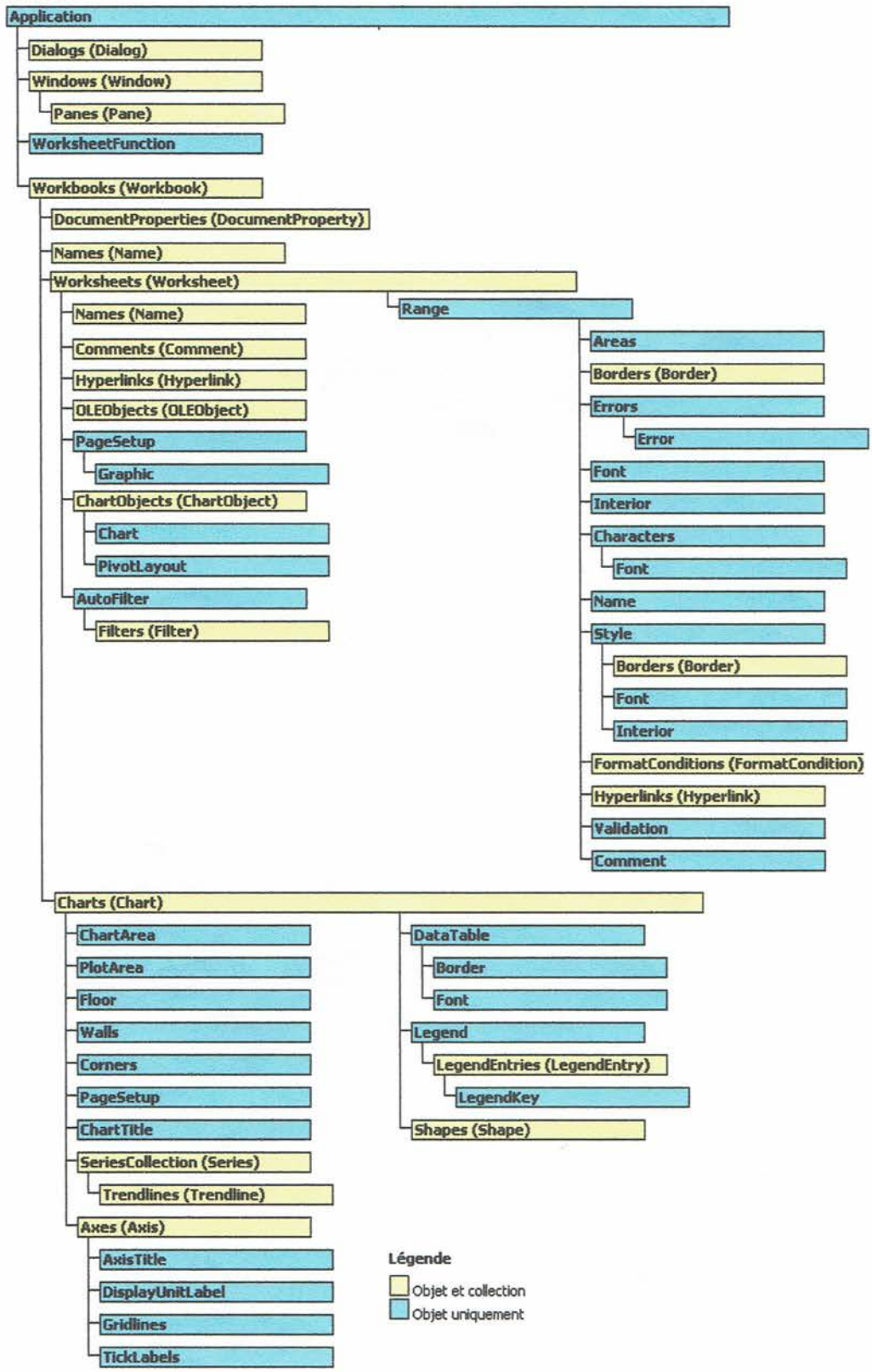


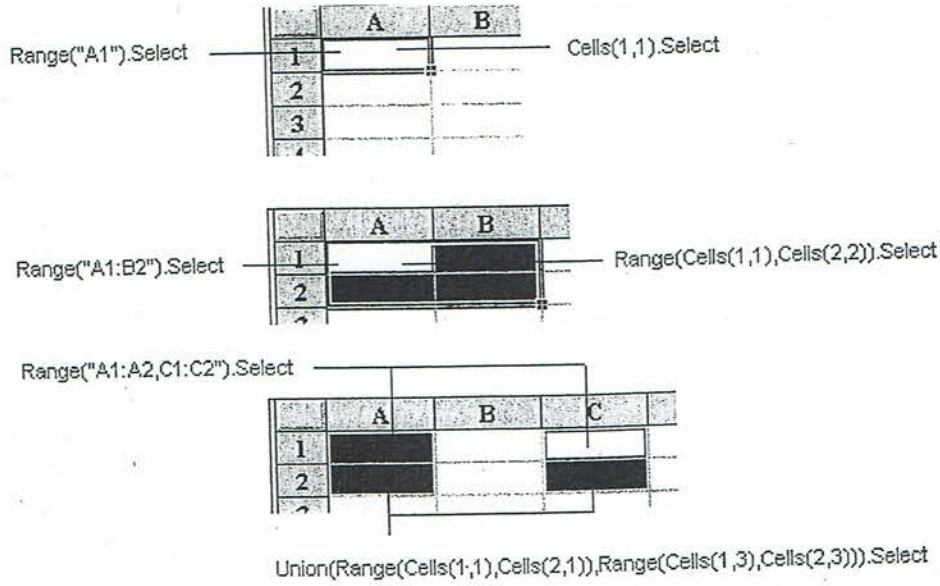
Encore faut il avoir bien à l'esprit qu'il ne s'agit là que d'un premier niveau de hiérarchie.

La simple collection WorkSheets se décompose telle que :



Il n'est pas question ici de détailler l'ensemble de ce modèle objet. Nous allons nous cantonner dans l'étude des principaux objets utilisés dans des macros classiques. Le modèle objet vu dans ce cours sera :

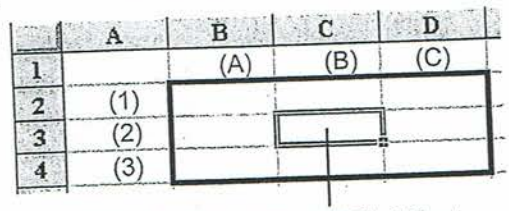




MULTIPLE SELECTIONS USING THE CELLS METHOD MUST ALSO USE THE UNION METHOD.

Virtual Worksheets

A Range object is a *virtual worksheet*. Cells can be referred to relative to the Range object, which need not necessarily coincide with the real worksheet.

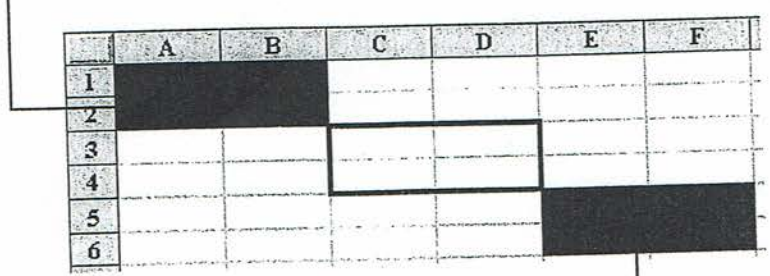


Range("B2:D4").Range("B2").Select or
Range("B2:D4").Cells(2,2).Select or
Range("B2:D4").Cells(5).Select

In the example above, Range("B2:D4").Range("B2").Select selects B2 on the virtual worksheet, which actually corresponds to C3 on the real worksheet. However, when referring to a range within a virtual worksheet it is often better to refer to the range using the Cells method:

e.g. Range("B2:D4").Cells(2,2).Select selects the cell in the second row and second column of the range.

Range Contents



Range("C3:D4").Offset(2,2).Select

Variabels Variant.

Variables can be declared as any one of the following data types:

Byte data type
A data type used to hold positive integer numbers ranging from 0 to 255. Byte variables are stored as single, unsigned 8-bit (1-byte) numbers.
Boolean data type
A data type with only two possible values, True (-1) or False (0). Boolean variables are stored as 16-bit (2-byte) numbers.
Integer data type
A data type that holds integer variables stored as 2-byte whole numbers in the range -32,768 to 32,767. The Integer data type is also used to represent enumerated values. The percent sign (%) type-declaration character represents an Integer in Visual Basic.
Long data type
A 4-byte integer ranging in value from -2,147,483,648 to 2,147,483,647. The ampersand (&) type-declaration character represents a Long in Visual Basic.
Currency data type
A data type with a range of -922,337,203,685,477.5808 to 922,337,203,685,477.5807. Use this data type for calculations involving money and for fixed-point calculations where accuracy is particularly important. The at sign (@) type-declaration character represents Currency in Visual Basic.
Single data type
A data type that stores single-precision floating-point variables as 32-bit (2-byte) floating-point numbers, ranging in value from -3.402823E38 to -1.401298E-45 for negative values, and 1.401298E-45 to 3.402823E38 for positive values. The exclamation point (!) type-declaration character represents a Single in Visual Basic.
Double data type
A data type that holds double-precision floating-point numbers as 64-bit numbers in the range -1.79769313486232E308 to -4.94065645841247E-324 for negative values; 4.94065645841247E-324 to 1.79769313486232E308 for positive values. The number sign (#) type-declaration character represents the Double in Visual Basic.
Date data type
A data type used to store dates and times as a real number. Date variables are stored as 64-bit (8-byte) numbers. The value to the left of the decimal represents a date, and the value to the right of the decimal represents a time.
String data type
A data type consisting of a sequence of contiguous characters that represent the characters themselves rather than their numeric values. A String can include letters, numbers, spaces, and punctuation. The String data type can store fixed-length strings ranging in length from 0 to approximately 63K characters and dynamic strings ranging in length from 0 to approximately 2 billion characters. The dollar sign ($) type-declaration character represents a String in Visual Basic.
Object data type
A data type that represents any Object reference. Object variables are stored as 32-bit (4-byte) addresses that refer to objects. Variant data type A special data type that can contain numeric, string, or date data as well as the special values Empty and Null. The Variant data type has a numeric storage size of 16 bytes and can contain data up to the range of a Decimal, or a character storage size of 22 bytes (plus string length), and can store any character text. The VarType function defines how the data in a Variant is treated. All variables become Variant data types if not explicitly declared as some other data type.

There are some words you should (must) not use to name your variables. Those words are reserved for the VBA internal use. Therefore, those words are called keywords. Some of them are:

And (Bitwise)	And (Condition)	As	Boolean	ByRef	Byte
ByVal	Call	Case	CBool	CByte	CDate
CDbl	CInt	CLng	Const	CSng	CStr
Date	Dim	Do	Double	Each	Else
ElseIf	End	EndIf	Error	False	For
Function	Get	GoTo	If	Integer	Let
Lib	Long	Loop	Me	Mid	Mod
New	Next	Not	Nothing	Option	Or (Bitwise)
Or (Condition)	Private	Public	ReDim	REM	Resume
Select	Set	Single	Static	Step	String
Sub	Then	To	True	Until	vbCrLf
vbTab	With	While	Xor		

The Buttons of a Message Box

The *Buttons* argument specifies what button(s) should display on the message box. There are different kinds of buttons available and the VBA language. Each button uses a constant integer as follows:

Constant	Numeric Value	Display
vbOKOnly	0	OK
vbOKCancel	1	OK Cancel
vbAbortRetryIgnore	2	Abort Retry Ignore
vbYesNoCancel	3	Yes No Cancel
vbYesNo	4	Yes No
vbRetryCancel	5	Retry Cancel

The Returned Value of a Message Box

The **MsgBox()** function can be used to return a value. This value corresponds to the button the user clicked on the message box. Depending on the buttons the message box is displaying, after the user has clicked, the **MsgBox()** function can return a value. The value can be a member of the **MsgBoxResult** enumeration or a constant numeric value recognized by the Visual Basic language. The value returned can be one of the following values:

If the user click	The function returns	Numeric Value
OK	vbOK	1
Cancel	vbCancel	2
Abort	vbAbort	3
Retry	vbRetry	4
Ignore	vbIgnore	5
Yes	vbYes	6
No	vbNo	7

VBA Workbooks, Worksheets and Ranges

ACCESS A WORKBOOK

The current workbook	ThisWorkbook
The active workbook	ActiveWorkbook
Any open workbook	Workbooks("example.xlsx")
Open a workbook	Workbooks.Open "C:\docs\book.xlsx"
Using a variable	Dim wk As Workbook Set wk = Workbooks("example.xlsx")

ACCESS A WORKSHEET

In current workbook	ThisWorkbook.Worksheets("name")
In current workbook	Use the code name e.g. Sheet1
In a given workbook	wk.Worksheets("name")
The active worksheet	ActiveSheet
Worksheet variable	Dim sh As Worksheet Set sh = wk.Worksheets("name")

USING RANGE

Access Range	<worksheet>.Range
Read value from cell	Total = Range("A1") .Value
Write value to cell	Range("A1").Value = 5
Assign one cell to another	Range("A1").Value = Range("B1") .Value
Assign multiple cells	Range("A1:C4").Value = Range("F1:H4").Value
Read from range to array	Dim arr As Variant arr = Range("A1:B3").Value
Write from array to range	Range("A1:B3").Value = arr
Assign and transpose	Range("E1:H1").Value = WorksheetFunction.Transpose

USING VARIABLES WITH RANGES

Row is variable	Range("A" & i) **Or** Cells(i,1)
Column is variable	Cells(1,i)
Multiple cells	Range("A1:A" & i) **Or** Range(Cells(1,1),Cells(i,1))

OFFSET PROPERTIES

Offset	Moves range by rows and colums
Get cell to right	Range("B2").Offset(0,1) ' C2
Get cell to left	Range("B2").Offset(0,-1) ' A2
Get cell above	Range("B2").Offset(-1,0) ' B1
Get cell below	Range("B2").Offset(1,0) ' B3
Multiple cells	Range("A1:B2").Offset(3,3) ' D4:E5

USING CURRENT REGION

CurrentRegion	Gets the entire range of data
Cells A1:D5 have data. Get entire range	Range("A1") .CurrentRegion **Or** Range("B2") .CurrentRegion **Or** Range("D5") .CurrentRegion **etc.**

ROWS AND COLUMNS

Last row	Cells(Rows.Count, 1).End(xlUp).row
Last column	Cells(1,Columns.Count).End(xlToLeft).Column
Cell count	Range("A1") .Count
Row count	Range("A1:D5") .Rows.Count
Column count	Range("A1:D5") .Columns.Count
Get a row	<worksheet>.Rows(3)
Get a column	<worksheet>.Columns(2)

READING THROUGH RANGES

Every cell in range	Dim rg As Range For Each rg In Range("A1:A10") Debug.Print rg Next rg

Read by row	Dim row As Range
	For Each row In Range("A1:A5").Rows
	  ' Print cell in third column of row
	  Debug.Print row.Cells(1,3)
	Next row

READING THROUGH OTHER ITEMS

All Open workbooks	Dim wk As Workbook For Each wk In Workbooks Debug.Print wk.Name Next wk
All worksheets in a workbook	Set wk = Workbooks("Test.xlsx") Dim sh As Worksheet For Each sh In wk.Worksheets Debug.Print sh.Name Next sh

USEFUL PROPERTIES FOR TESTING

Worksheet of a range	Range("A1") .Parent
Workbook of a worksheet	Worksheets(1).Parent
Workbook of a range	Range("A1").Parent.Parent
Workbook name	Workbooks(1).Name
Workbook path	Workbooks(1).Path
Workbook path + name	Workbooks(1).FullName
Current user	Application.UserName

COPYING RANGES

Everything	Range("A1:A5").Copy Range("C1:C5")
Paste Special	Range("A1:A5").Copy Range("C1:C5").PasteSpecial xlPasteFormats Range("C1:C5").PasteSpecial xlPasteValues

VBA Arrays, Collections and Dictionaries

STATIC ARRAY

Declare	Dim arr(0 To 5) As Long Dim arr(5) As Long
Reset all values	Erase arr

DYNAMIC ARRAY

Declare	Dim arr() As Long
Set size	ReDim arr(1 To 10)
Increase size of existing array	ReDim Preserve arr(1 To 10)
Set size to zero	Erase arr

DYNAMIC AND STATIC ARRAY

Assign a value	arr(1) = 56
Go through all items(For)	For i = LBound(arr) To UBound(arr) Debug.Print arr(i) Next i
Go through all items(For Each)	For Each v In arr Debug.Print v Next v

ARRAYS AND RANGES

Create variant array	Dim arr() As Variant
Read cell values to array	arr = Range("A1:Z2").Value
Write array values to Range	Range("A3:Z4").Value = arr

ARRAYS AND SUBS/FUNCTIONS

Use array parameter	Sub UseArray(ByRef arr()) As Long)
Pass to procedure	Dim arr(0 To 2) As Long UseArray arr

Use as return value	Function GetArray() As Long()
	  Dim arr(0 To 2) As Long
	  GetArray = arr
	End Sub

Return from procedure	Dim arr() As Long arr = GetArray

TWO DIMENSIONAL ARRAY

Create	Dim arr(0 To 2,0 To 4) As Long
Assign value	arr(0,0) = 45 arr(2,4) = 67

Read through array	Dim i As Long, j as Long
	For i = LBound(arr) To UBound(arr)
	  For j = LBound(arr, 2) To UBound(arr, 2)
	  Debug.Print arr(i, j)
	  Next j
	Next i

COLLECTION

Declare and create	Dim coll As New Collection
Declare and create in two steps	Dim coll As Collection Set coll = New Collection
Add item	coll.Add "Apple" coll.Add 55
Access item	Range("A1") = coll(1)
Remove item at index two	Coll.Remove 2
Go through all items (For)	For i = 1 To coll.Count Debug.Print coll(i) Next i
Go through all items(For Each)	For Each v In coll Debug.Print v Next v

DICTIONARY

Early binding reference	"Microsoft Scripting Runtime" (Add using Tools->References)
Declare(early binding)	Dim dict As Scripting.Dictionary
Create(early binding)	Set dict = New Scripting.Dictionary
Declare(late binding)	Dim dict As Object
Create(late binding)	Set dict = CreateObject("Scripting.Dictionary")
Add item. Key must not already exist.	dict.Add <key>,<value> dict.Add "Apples",50
Silent Add. Updates value if key exists.	dict(<key>)= value dict("Orange") = 67
Access item	value = dict("Apple")
Check if item exists	If dict.Exists("Apple") Then
Remove item	dict.Remove "Apple"
Remove all items	dict.RemoveAll
Go through all items	Dim key As Variant For Each key In dict.Keys Debug.Print key, dict(key) Next
Number of items	dict.Count
Make key non case sensitive. Dictionary must be empty.	dict.CompareMode = TextCompare