

Aide-mémoire VBA

Enregistrer le fichier

Classeur Excel prenant en charge les macros XLSM

Débloquer le fichier (PC)

Avant d'ouvrir, dans l'explorateur de fichier:

1. Clic droit sur le fichier téléchargé
2. Choisir propriété
3. Cocher "débloquer" en bas

Enregistrer une macro

Menu Affichage / Macros / Enregistrer une macro

Possibilité d'enregistrer en mode "relatif" (par défaut, enregistrement en relatif)

Afficher l'onglet Développeur

Fichier > Options > Personnaliser le Ruban > cocher Développeur (dans la colonne de droite)

A chaque ouverture du fichier, les macros sont enregistrées dans un nouveau module.

Détail des macros

Nom de macro

Lettres, chiffres, et soulignement. La plupart des autres caractères sont interdits

Ambiguïté : On ne peut pas avoir deux macros avec le même nom

Lieux d'enregistrement

- Dans ce fichier
- Dans le fichier de macro personnel **xlsb** ou macro complémentaires **xlam**

Raccourcis

On peut ensuite changer le raccourci dans



Affichage ou Développeur > Macro > Afficher les macros > options

Déclencher une macro

- Directement dans l'éditeur avec le bouton exécuter
- En ajoutant un bouton dans une feuille excel Clic droit > affecter une macro
- A partir d'un évènement (ouverture du classeur, changement de feuille, avant fermeture...)
- A l'aide d'un formulaire personnalisé

Parties de l'éditeur

Raccourci pour ouvrir l'éditeur: Alt+F11 (ou Alt+Fn+F11)

- Explorateur de projet
- Fenêtre de propriétés
- Explorateur d'objet (contenant les propriétés, les méthodes , les événements )
- Fenêtre d'exécution (affiche les Debug.Print)
- Fenêtre Variables locales
- Fenêtre espion (utile en mode débogage ou avec les points d'Arrêt)

Si vous n'avez pas enregistré de macro, vous devez commencer par insérer un module: Insertion > Module

On peut protéger le code avec un mot de passe: Outil > Propriétés de VBAProject > Protection

Commentaires

Les commentaires commencent par une apostrophe. On peut faire un commentaire après du code.

'Ceci est un commentaire

Couper les lignes

Les longues lignes peuvent être coupées pour une meilleure lecture avec: **espace+souligné+enter**

MsgBox("N'oubliez pas de taper un espace avant le souligné " & nom _

& " et de taper ENTER juste après.")

Sub

Une macro commence par **Sub** et finit par **End Sub**

Sub est l'abréviation de sub routine (sous routine): L'application est le programme (la routine) et la macro est un sous-programme, un programme dans le programme, donc un sous-routine.

Private: La macro n'apparaît pas dans la liste des macros. Elle ne peut être lancée que par une autre macro.

Écrire dans une cellule

Range("A2") = "Bonjour"

Texte: Cells(2,1) = "Bonjour"

Formule: Range("D3") = "=D1+D2"

Formule relative: Range("D3").FormulaR1C1 = "=R[-2]C+R[-1]C"

Lire dans une cellule

Valeur (version abrégée): Range("A2") ou Cells(2,1)

Valeur: Range("A2").value

Formule: Range("D3").Formula ou Formula2

Formule relative: Range("D3").FormulaR1C1 ou Formula2R1C1

En se décalant: ActiveCell.Offset(1,2).value

Conditions

Alternatives avec IF

```
If Range("A1")>10 Then
    Range("A2") = "Moyen"
Elseif Range("A2") > 5 Then
    Range("A2") = "Petit"
Else
    Range("A2") = "Grand"
End If
```

Il existe une syntaxe sur une ligne sans End IF:

```
If Range("A1")>10 Then Range("A2")="Moyen" Else Range("A2")="Grand"
```

Ligne unique avec plusieurs instructions, séparée par des deux points:

```
If A > 10 Then A = A + 1 : B = B + A : C = C + B
```

Alternatives avec Select Case

```
Select Case Range("D3")
Case 1
    Bonus = salary * 0.1
Case 2, 3
    Bonus = salary * 0.09
Case 4 To 6
    Bonus = salary * 0.07
Case Is > 8
    Bonus = 100
Case Else
    Bonus = 0
End Select
```

Déclarer des variables

Référence: [résumé des types de données](#) (la trad fr est incomplète)

Pour obliger la déclaration des variables on peut ajouter **Option Explicit** en haut du module.

Texte: `Dim s as String`

Entier entre -32768 et 32767: `Dim i as Integer`

Grands nombres: `Dim n as Long` ou `Dim n as Double`

Deux valeurs (Vrai / Faux): `Dim b as Boolean`

Plusieurs types : `Dim v as Variant`

Static : Tant que le classeur reste ouvert, la variable sera initialisée à sa dernière valeur à chaque fois que la macro sera exécutée

Public: La portée de la variable est globale à tous les modules standard du projet VBA

Tester une valeur

`IsArray, IsDate, IsEmpty, IsError, IsMissing, IsNull, IsNumeric, IsObject`

Formules fréquentes

Extrait des caractères: `Left("Bonjour",3)our` ou `Right("Bonjour", 2)Bo` ou `Mid("Bonjour",3,2)o`

Majuscule Minuscule: `UCase("Bonjour")BONJOUR` ou `LCase("Bonjour")bonjour`

Coupe en plusieurs parties (voir Array): `Split("Dossier A/Sous-dossier B/Fichier C", "/")`

Cherche une expression: `InStr(ActiveCell.value, "p")`

Cherche une expression en partant de la droite: `InStrRev(ActiveCell.value, "p")`

Remplace toutes les occurrences d'un texte: `replace(ActiveCellvalue, "p", "P")`

Si les formules de VBA sont insuffisantes, on peut utiliser les formules de Excel (avec leur nom anglais):

`Application.WorksheetFunction.CountA(Range("A:A"))`

Nb Count; **Nbval** CountA; **Nb.si.ens** Countifs; **Somme.si.Ens** Sumifs; **RechercheV** Vlookup

Messages et demandes

`MsgBox "Bonjour"`

`Réponse = MsgBox("Bonjour", vbOkCancel)`

`Réponse = InputBox("Quel est votre âge?")`

`MsgBox "Tu as:" & VbCrLf & réponse & " ans"`

L'inputBox normale du VBA ne permet pas de choisir une plage, par contre on peut le faire avec l'[Application.InputBox](#) qui est spécifique à Excel, en utilisant le type 8.

Position = Application.InputBox("Choisissez une cellule", Type:=8)

Les objets dans Excel

Classeur: Workbook

Compter les classeurs ouverts: Workbooks.count

Le nom du premier classeur: Workbooks(1).name

Le classeur où se trouve la macro en cours: ThisWorkbook

Le classeur actif: ActiveWorkbook

Plage: Range

ClearContents ou ClearFormats ou Clear.Comments

Selection:

Range("A1:B5").Select

Selection.count

Savoir le type d'objet sélection: TypeName(Selection)

Cellules: cells

Lignes: Rows

Rows(4).Insert

Range("b4").EntireRow.Insert

Rows("4:6").Insert

Colonne: Columns

Selection.EntireColumn.Autofit

Feuilles: Sheets

La feuille active: ActiveSheet

Compter le nombre de feuille: Sheets.Count

Le numéro d'après le nom: Sheets("Mars 2023").index

Sheets.Add

Sheets.Add After:=Sheets("Input")

Sheets.Add.Name = "NewSheet"

Sheets.Add(After:=Sheets("Input")).Name = "NewSheet"

Sheets.Add After:=Sheets(Sheets.Count)

Formes ou images: Shapes

Graphiques: Charts

Tableaux: ListObjects

Tableaux croisés dynamiques (TCD): PivotTables

Actualiser tout: ActiveWorkbook.RefreshAll

Copier Coller: Copy Paste

```
Range("A1").Copy Range("B1")  
Range("A1:A3").Copy Range("B1:B3")
```

'Couper Coller une ligne

```
Range("1:1").Cut Range("2:2")
```

'Collage spécial

```
Range("A1").Copy  
Range("B1").PasteSpecial Paste:=xlPasteFormulas
```

Plusieurs propriétés avec With

```
With Range("A1")  
    .Font.Bold = True  
    .Font.ThemeColor = xlThemeColorAccent4  
    .Interior.ThemeColor = xlThemeColorAccent1  
End With
```

Boucles

Rappel: en cas de boucle infinie, utiliser Ctrl+Pause (en anglais Ctrl+Break)

Boucle For

```
For each c in selection  
Next c
```

```
For i= 1 to 10  
    Debug.Print i  
Next i
```

```
For i = 10 to 1 Step -2  
Next i
```

On peut sortir d'une boucle for avec **exit for**.

On peut soit faire un point d'Arrêt soit mettre **Stop** dans le code pour suivre l'exécution d'une boucle.

Boucle Do While

```
Sub répéter()  
    ' la boucle fait pause tous les multiples de 100  
    Dim i As Integer  
    i = 2  
    Do While i < 500  
        Cells(i, 1) = Cells(i - 1, 1) + 2  
        i = i + 1  
        ' mod est le modulo: le reste de la division  
        If i Mod 100 = 0 Then  
            Stop  
        End If  
    Loop  
End Sub
```



Vidéo YouTube Boucles

On peut sortir d'une boucle Do avec **Exit Do**.

Il y a aussi une boucle **While Wend**.

Fonctions

```
Function tva(ht)  
    Tva = ht * 0.2  
End Function
```

Set

Permet d'affecter un objet à une variable (d'habitude on affecte des valeurs aux variable).

```
Dim wb As Workbook  
Dim ws As Worksheet  
Set wb = Workbook("ma première macro.xlsm")  
Set ws = wb.sheets("comptabilité mensuelle")
```

Les arrays

Les arrays sont variables contenant plusieurs valeurs. Par défaut les éléments sont comptés (indexés) en commençant à 0

```
Sub découper()  
    Dim monArray() As String  
    monArray = Split("Les sanglots longs des violons", " ")  
    MsgBox UBound(monArray) affiche 4 car on compte 0,1,2,3,4 mots  
End Sub  
  
Dim MaSem  
MaSem = Array("Lundi", "Mardi", "Mercredi", "Jeudi", "Vendredi")  
MsgBox MaSem(1) affiche "Mardi" et non "Lundi"!!!
```

On peut ajouter **Option Base 1** En haut du module pour indiquer que l'on comptera à partir de 1.

Option Base 1

Dim MaSem

MaSem = Array("Lundi", "Mardi", "Mercredi", "Jeudi", "Vendredi")

MsgBox MaSem(1)^{affiche "Lundi"}

Redim

On peut agrandir un array en conservant les éléments précédemment ajoutés.

Sub agrandirArray()

Dim A As Variant, B As Long, i As Long

A = Array(10, 20, 30) ' A est une liste de 3 éléments indexés par défaut de 0 à 2

B = A(2) ' B est maintenant égal à 30

Re Dim Preserve A(4) ' On étend la taille de A à 5 éléments

A(4) = 40 ' On définit la valeur du cinquième élément

For i = LBound(A) To UBound(A)

Debug.Print "A(" & i & ") = " & A(i)

Next i

End Sub

Déclaration d'un tableau multidimensionnel

Dim tableau(5,3) as Integer 'tableau à 2 dimensions de 5 sur 3 entiers

tableau(3,1)=10

Le système de fichier

Afficher le séparateur de fichier pour le système en cours: Application.PathSeparator

Le répertoire actuel: CurDir

Créer un dossier: MkDir

Supprimer un répertoire: RmDir

Changer de dossier: ChDir

Monter d'un dossier: ChDir ".."

Changer de lecteur: ChDrive


```

Sub choisirDossier()
'variable pour le chemin du dossier
Dim cDossier As String

'Ouvrir le sélecteur de fichier
With Application.FileDialog(msoFileDialogFolderPicker)
.Show
    cDossier = .SelectedItems(1)
End With

' on écrit le chemin en A1
Range("A1") = cDossier
End Sub

```

```

Sub LoopThroughFiles()
Dim oFSO As Object 'FSO = File System Object
Dim oFolder As Object
Dim oFile As Object
Dim i As Integer
Set oFSO = CreateObject("Scripting.FileSystemObject")
Set oFolder = oFSO.GetFolder("C:\Documents\")
For Each oFile In oFolder.Files
    Cells(i + 1, 1) = oFile.Name
    i = i + 1
Next oFile
End Sub

```

Erreurs

Erreurs fréquentes

Nom ambigu: deux macros ont le même nom

Propriété non gérée par ce objet: Votre action concerne une cellule, mais c'est un graphique qui est sélectionné.

Incompatibilité de type: Vous faites une opération mathématique sur du texte

On Error Resume Next

On peut dire au VBA de ne pas tenir compte des erreurs:

```

Sub Clear_All_Filters_Range()
'On utilise ShowAllData pour supprimer les filtres
'de la feuille. Mais une erreur apparaît s'ils sont déjà enlevés
'(pour les tables utiliser: maTable.AutoFilter.ShowAllData)
On Error Resume Next
ActiveSheet.ShowAllData
On Error GoTo 0
End Sub

```

Goto

Il est possible de sauter des parties de code avec Goto:

```
Sub GotoDemo()  
    Dim Nombre, MonMessage  
    Nombre = 1  
    ' On oriente en fonction du nombre  
    If Nombre = 1 Then GoTo Ligne1 Else GoTo Ligne2  
  
    Ligne1:  
        MonMessage = "Nombre égale 1"  
        GoTo DernièreLigne ' On va à DernièreLigne  
  
    Ligne2:  
        ' Cette partie n'est jamais exécutée  
        MonMessage = "Nombre égale 2"  
  
    DernièreLigne:  
        Debug.Print MonMessage ' Écrit "Nombre égale 1" dans  
        ' la fenêtre d'exécution.  
End Sub
```

Attendre

'attend une certaine heure aujourd'hui

```
Application.Wait "18:23:00"
```

'indique si 10 secondes se sont écoulées

```
If Application.Wait(Now + TimeValue("0:00:10")) Then  
    MsgBox "Time expired"  
End If
```

UserForm



Youtube UserForm