

COURS VBA GRATUIT

MESETUDESDECAS.FR | BIBLIOTHÈQUE VBA

Le VBA Excel, une fiche par notion.

21 fiches de cours pour passer de la première macro enregistrée au formulaire complet : définitions, syntaxe, exemples prêts à copier et pièges à éviter.

Version	V1.2
Dernière mise à jour	29/10/2026
Auteur	Clément Delobel, formateur Excel et VBA



Toutes les fiches en ligne
mesetudesdecas.fr/vba/

SOMMAIRE

Les 21 fiches

01 BASES

L'enregistreur de macros

02 BASES

Sub et Function

03 BASES

Variables et Dim

04 BASES

MsgBox et InputBox

05 CONDITIONS

If Then Else

06 CONDITIONS

Select Case

07 BOUCLES

Boucle For Next

08 BOUCLES

Boucle For Each

09 BOUCLES

Do While et Do Until

10 OBJETS EXCEL

Range et Cells

11 OBJETS EXCEL

Trouver la dernière ligne

12 OBJETS EXCEL

Feuilles et classeurs

13 OBJETS EXCEL

With End With

14 FORMULAIRES

Créer un premier UserForm

15 FORMULAIRES

Les contrôles d'un formulaire

16 DÉBOGAGE

Déboguer pas à pas (F8)

17 DÉBOGAGE

On Error et gestion d'erreurs

18 ERREURS

Erreur 1004

19 ERREURS

Erreur 91

20 ERREURS

Erreur 13

21 ERREURS

Erreur 9

FICHE DE COURS GRATUITE

VBA EXCEL | BASES

L'enregistreur de macros

L'enregistreur de macros transforme vos clics en code VBA. C'est la porte d'entrée idéale : on fait la manipulation une fois, Excel écrit le code, et on apprend en le relisant. Il a aussi ses limites, qu'il faut connaître pour passer à la suite.

Version	V1.2
Dernière mise à jour	29/10/2026
Niveau	1 / 3, Découverte
Durée de lecture	7 min
Auteur	Clément Delobel



Version en ligne
mesetudesdecas.fr/vba/enregistreur-de-macros.html

01 Définition

EN UNE PHRASE

L'enregistreur de macros **mémore vos actions dans Excel** et les traduit en instructions VBA, que vous pouvez ensuite rejouer d'un clic ou d'un raccourci clavier.

Une macro est simplement une procédure VBA, c'est-à-dire une suite d'instructions rangée entre `Sub` et `End Sub`. L'enregistreur l'écrit à votre place dans un module du classeur.

02 Afficher l'onglet Développeur

Tout se passe dans l'onglet **Développeur**, masqué par défaut.

- 1 Ouvrez **Fichier > Options > Personnaliser le ruban**.
- 2 Dans la liste de droite, cochez **Développeur**, puis validez.
- 3 L'onglet apparaît : il contient les boutons **Enregistrer une macro**, **Macros** et **Visual Basic** (raccourci `Alt + F11`).

03 Enregistrer une macro, étape par étape

- 1 Cliquez sur **Enregistrer une macro**.
- 2 Donnez un **nom** sans espace ni caractère spécial, par exemple `MiseEnFormeEntete`.
- 3 Choisissez si besoin un **raccourci** : préférez `Ctrl + Maj + lettre`, pour ne pas écraser un raccourci d'Excel comme `Ctrl + C`.
- 4 Choisissez où stocker la macro : **Ce classeur**, ou **Classeur de macros personnelles** pour la retrouver dans tous vos fichiers.
- 5 Faites vos manipulations, puis cliquez sur **Arrêter l'enregistrement**.

 ASTUCE**Références absolues ou relatives ?**

Par défaut, l'enregistreur note des adresses fixes : « la cellule B3 ». Activez **Utiliser les références relatives** avant d'enregistrer si la macro doit agir à partir de la cellule active, quelle qu'elle soit.

04 Relire le code produit

Ouvrez l'éditeur avec **Alt + F11** : la macro se trouve dans un module, en général **Module1**. Voici ce que produit l'enregistreur pour mettre en gras et colorer une ligne d'en-tête :

CODE BRUT DE L'ENREGISTREUR

```
Sub MiseEnFormeEntete()  
,  
,  
,  
,  
    Range("A1:D1").Select  
    Selection.Font.Bold = True  
    With Selection.Interior  
        .Pattern = xlSolid  
        .PatternColorIndex = xlAutomatic  
        .Color = 16247773  
        .TintAndShade = 0  
        .PatternTintAndShade = 0  
    End With  
    Range("A1").Select  
End Sub
```

Le code fonctionne, mais il est bavard : il sélectionne avant d'agir et recopie toutes les propriétés, même celles que vous n'avez pas touchées.

05 Nettoyer le code

Trois réflexes suffisent pour rendre un code enregistré plus court, plus rapide et plus fiable :

- supprimer les **.Select** et agir directement sur l'objet ;
- supprimer les propriétés inutiles ;
- regrouper les actions sur un même objet avec **With End With**.

LE MÊME CODE, NETTOYÉ

```
Sub MiseEnFormeEntete()  
    With Range("A1:D1")  
        .Font.Bold = True  
        .Interior.Color = RGB(221, 235, 247)  
    End With  
End Sub
```

06 Pièges à éviter

⚠ PIÈGE

Enregistrer en .xlsx

Un classeur `.xlsx` ne peut pas contenir de macros : Excel les supprime à l'enregistrement. Enregistrez au format **Classeur Excel prenant en charge les macros (.xlsm)**.

⚠ PIÈGE

Pas de retour arrière

Une action réalisée par une macro ne s'annule pas avec `Ctrl + Z`. Testez toujours une nouvelle macro sur une copie du fichier.

⚠ PIÈGE

Tout est enregistré

Un clic de trop, un défilement, une cellule sélectionnée par erreur : l'enregistreur note tout. Préparez votre manipulation avant de cliquer sur Enregistrer.

⚠ PIÈGE

Macros bloquées à l'ouverture

Un fichier .xlsm reçu par mail ou téléchargé peut avoir ses macros bloquées. Clic droit sur le fichier, **Propriétés**, puis cochez **Débloquer**.

07 Les limites de l'enregistreur

L'enregistreur ne sait pas écrire de condition, de boucle ni de variable : il rejoue toujours exactement la même chose. Dès qu'il faut que la macro **décide** (**If Then Else**) ou **répète** (**For Next**), il faut écrire le code soi-même. C'est tout l'objet des fiches suivantes.

FICHE DE COURS GRATUITE

VBA EXCEL | BASES

Sub et Function

Tout code VBA vit dans une procédure. Une Sub agit, une Function calcule et renvoie un résultat. Savoir choisir entre les deux, et les faire dialoguer, c'est la base de toute macro bien construite.

Version	V1.2
Dernière mise à jour	29/10/2026
Niveau	1 / 3, Découverte
Durée de lecture	7 min
Auteur	Clément Delobel



Version en ligne
mesetudesdecas.fr/vba/sub-et-function.html

01 Définition

EN UNE PHRASE

Une **Sub** est une procédure qui exécute des actions. Une **Function** est une procédure qui **renvoie une valeur**, utilisable dans le code ou directement dans une cellule Excel.

Sub

Met en forme, copie, supprime, envoie un message.
Elle apparaît dans la liste des macros (Alt + F8).

Function

Calcule et renvoie un résultat. Elle n'apparaît pas dans la liste des macros, mais dans les fonctions Excel.

02 Syntaxe

SYNTAXE GÉNÉRALE

```
Sub NomDeLaMacro()  
    ' instructions  
End Sub
```

```
Function NomDeLaFunction(argument As Type) As TypeRetour  
    ' calcul  
    NomDeLaFunction = resultat  
End Function
```

Élément	Rôle
Sub / Function	Mot-clé qui ouvre la procédure.
(argument As Type)	Données transmises à la procédure. Facultatif pour une Sub.
As TypeRetour	Type de la valeur renvoyée par une Function.

Élément	Rôle
NomDeLaFonction = ...	Affecter une valeur au nom de la fonction, c'est ce qui la renvoie.
End Sub / End Function	Fin de la procédure.

03 Créer une fonction personnalisée

Une Fonction écrite dans un **module standard** devient utilisable dans les cellules, comme SOMME ou RECHERCHEX.

UNE FONCTION TTC AVEC TAUX FACULTATIF

```
Function TTC(montantHT As Double, Optional taux As Double = 0.2) As Double
    TTC = montantHT * (1 + taux)
End Function
```

Dans la feuille, on écrit `=TTC(A2)` pour un taux de 20 %, ou `=TTC(A2; 0,055)` pour un taux de 5,5 %. Le mot-clé `Optional` rend l'argument facultatif et lui donne une valeur par défaut.

💡 ASTUCE

Retrouver ses fonctions

Vos fonctions personnalisées apparaissent dans l'assistant fonction (bouton fx), catégorie **Personnalisées**.

04 Appeler une procédure depuis une autre

Découper une macro en petites procédures la rend plus lisible. Une Sub peut en appeler une autre, et utiliser le résultat d'une Function.

UNE MACRO PRINCIPALE QUI DÉLÈGUE

```
Sub Traitement()
    PreparerFeuille
    Call MettreEnForme("A1:F1")
    MsgBox "Total TTC : " & TTC(Range("B10").Value)
End Sub

Sub PreparerFeuille()
    Range("A2:F500").ClearContents
End Sub

Sub MettreEnForme(adresse As String)
    Range(adresse).Font.Bold = True
End Sub
```

05 ByVal et ByRef

Par défaut, VBA transmet les arguments **ByRef** (par référence) : si la procédure appelée modifie l'argument, la variable d'origine est modifiée aussi. Avec **ByVal**, la procédure travaille sur une copie.

LA DIFFÉRENCE EN PRATIQUE

```
Sub Doubler(ByVal n As Long)
    n = n * 2          ' la variable d'origine ne change pas
End Sub
```

06 Pièges à éviter

 PIÈGE**Oublier d'affecter le résultat**

Une Function qui ne contient pas la ligne `NomDeLaFunction = ...` renvoie 0 ou une valeur vide, sans message d'erreur.

 PIÈGE**Une fonction de cellule ne modifie rien**

Appelée depuis une cellule, une Function ne peut que renvoyer une valeur : elle ne peut pas mettre en forme, écrire dans une autre cellule ou supprimer une ligne.

 PIÈGE**Le mauvais module**

Une fonction placée dans le module d'une feuille (Feuil1) ou dans ThisWorkbook n'est pas visible dans les cellules. Insérez un module standard : **Insertion > Module**.

 PIÈGE**Private cache la macro**

Une `Private Sub` n'apparaît pas dans la liste des macros. Pratique pour les sous-procédures, déroutant quand on l'ignore.

FICHE DE COURS GRATUITE

VBA EXCEL | BASES

Variables et Dim

Une variable est une boîte nommée dans laquelle la macro range une information pour s'en resservir : un total, un nom de client, une date, une plage de cellules. Bien les déclarer évite une bonne moitié des bugs.

Version	V1.2
Dernière mise à jour	29/10/2026
Niveau	1 / 3, Découverte
Durée de lecture	8 min
Auteur	Clément Delobel



Version en ligne
mesetudesdecas.fr/vba/variables-et-dim.html

01 Définition

EN UNE PHRASE

Une variable est un **espace mémoire nommé** qui stocke une valeur pendant l'exécution de la macro. On la déclare avec **Dim**, en précisant son **type**.

DÉCLARER ET UTILISER DES VARIABLES

```
Sub Exemple()
    Dim client As String
    Dim quantite As Long
    Dim prixUnitaire As Double

    client = "Dupont SARL"
    quantite = 12
    prixUnitaire = 14.5

    MsgBox client & " : " & quantite * prixUnitaire & " euros"
End Sub
```

02 Les types à connaître

Type	Contient	Exemple
Long	Nombre entier (jusqu'à environ 2,1 milliards)	numéro de ligne, quantité
Double	Nombre à virgule	prix, taux, montant
String	Texte	nom, référence, adresse
Boolean	True ou False	trouvé ou non

Type	Contient	Exemple
Date	Date et heure	échéance, date de commande
Variant	N'importe quoi (type par défaut)	à éviter sauf besoin réel
Range	Une cellule ou une plage	la cellule d'un total
Worksheet	Une feuille	la feuille Données

 ASTUCE**Long plutôt qu'Integer**

Integer s'arrête à 32 767, alors qu'une feuille compte plus d'un million de lignes. Pour un compteur ou un numéro de ligne, prenez toujours Long.

03 Les variables objet et Set

Une variable qui désigne un **objet** (cellule, feuille, classeur) se remplit avec **Set**. Sans Set, VBA déclenche l'**erreur 91**.

VARIABLES OBJET

```
Dim ws As Worksheet
Dim total As Range

Set ws = ThisWorkbook.Worksheets("Ventes")
Set total = ws.Range("F2")
total.Value = 1500
```

04 Option Explicit

Placée tout en haut du module, la ligne `Option Explicit` oblige à déclarer chaque variable. Une faute de frappe dans un nom devient alors une erreur signalée à la compilation, au lieu de créer silencieusement une nouvelle variable vide.

 ASTUCE**L'activer pour toujours**

Dans l'éditeur VBA : **Outils > Options**, cochez **Déclaration des variables obligatoire**. La ligne sera ajoutée automatiquement à chaque nouveau module.

05 Portée et constantes

- Déclarée **dans une procédure** : la variable n'existe que pendant cette procédure.
- Déclarée **en haut du module**, avec Dim ou Private : elle est partagée par toutes les procédures du module.
- Déclarée **Public** en haut d'un module standard : elle est visible partout dans le classeur.

Une valeur qui ne change jamais se déclare en constante : `Const TAUX_TVA As Double = 0.2`.

06 Pièges à éviter

⚠ PIÈGE

Dim a, b As Long

Seule la dernière variable est un Long : `a` est un Variant. Il faut écrire `Dim a As Long, b As Long`.

⚠ PIÈGE

Oublier Set

`ws = Worksheets("Ventes")` provoque une erreur. Pour un objet, c'est `Set ws = Worksheets("Ventes")`.

⚠ PIÈGE

Un texte dans un nombre

Ranger le contenu d'une cellule texte dans une variable Long déclenche l'**erreur 13**. Vérifiez avec `IsNumeric` avant de convertir.

⚠ PIÈGE

Des noms parlants

`x`, `a2`, `temp` ne disent rien six mois plus tard. Préférez `derniereLigne`, `totalHT`, `wsClients`.

FICHE DE COURS GRATUITE

VBA EXCEL | BASES

MsgBox et InputBox

Une macro a souvent besoin de parler à l'utilisateur : annoncer que le traitement est fini, demander une confirmation avant de supprimer, réclamer un mois ou un nom de client. MsgBox et InputBox sont les deux outils les plus simples pour cela.

Version	V1.2
Dernière mise à jour	29/10/2026
Niveau	1 / 3, Découverte
Durée de lecture	7 min
Auteur	Clément Delobel



Version en ligne
mesetudesdecas.fr/vba/
msgbox-et-inputbox.htm
1

01 Définition

EN UNE PHRASE

MsgBox affiche un message et peut proposer des boutons (Oui, Non, Annuler). **InputBox** affiche une zone de saisie et renvoie ce que l'utilisateur a tapé.

02 MsgBox : informer

SYNTAXE ET EXEMPLES

MsgBox message, [boutons], [titre]

MsgBox "Traitement terminé."

MsgBox "Fichier introuvable.", vbCritical, "Erreur"

MsgBox "Total : " & total & vbCrLf & "Lignes : " & nb

Le symbole `&` assemble les morceaux de texte, et `vbCrLf` passe à la ligne.

Constante	Effet
<code>vbInformation</code>	icône « i » bleue
<code>vbExclamation</code>	icône d'avertissement
<code>vbCritical</code>	icône d'erreur rouge
<code>vbQuestion</code>	icône point d'interrogation
<code>vbYesNo</code>	boutons Oui et Non
<code>vbOKCancel</code>	boutons OK et Annuler

03 MsgBox : demander une confirmation

Utilisée comme une fonction, MsgBox renvoie le bouton cliqué : `vbYes`, `vbNo`, `vbCancel` ...

CONFIRMER AVANT DE SUPPRIMER

```
Sub ViderDonnees()  
    If MsgBox("Effacer toutes les données ?", vbYesNo + vbQuestion, "Confirmation") = vbNo Then  
        Exit Sub  
    End If  
    Range("A2:F1000").ClearContents  
End Sub
```

04 InputBox : demander une saisie

DEMANDER UN NOM DE CLIENT

```
Sub Saluer()  
    Dim nom As String  
    nom = InputBox("Nom du client ?", "Recherche")  
  
    If nom = "" Then Exit Sub ' Annuler ou zone vide  
    MsgBox "Recherche de " & nom  
End Sub
```

05 Application.InputBox : contrôler le type

La version d'Excel, `Application.InputBox`, accepte un argument **Type** qui refuse les saisies incorrectes : 1 pour un nombre, 2 pour du texte, 8 pour une plage de cellules sélectionnée à la souris.

DEMANDER UN NOMBRE, PUIS UNE PLAGE

```
Dim quantite As Variant  
quantite = Application.InputBox("Quantité ?", Type:=1)  
If VarType(quantite) = vbBoolean Then Exit Sub ' Annuler renvoie False  
  
Dim zone As Range  
On Error Resume Next  
Set zone = Application.InputBox("Sélectionnez la plage", Type:=8)  
On Error GoTo 0  
If zone Is Nothing Then Exit Sub
```

06 Pièges à éviter

⚠ PIÈGE

InputBox renvoie du texte

"5" + "3" donne "53" en VBA, pas 8. Convertissez avec `CLng`, `CDbl` ou `Val` avant de calculer.

 PIÈGE**Annuler n'est pas une erreur**

Avec `InputBox`, `Annuler` renvoie une chaîne vide ; avec `Application.InputBox`, il renvoie `False`. Testez toujours ce cas, sinon la macro continue avec une valeur vide.

 PIÈGE**Trop de MsgBox**

Un message à chaque ligne d'une boucle bloque l'utilisateur. Gardez `MsgBox` pour le début, la fin ou les vraies questions, et utilisez `Debug.Print` pour suivre le déroulé pendant vos tests.

FICHE DE COURS GRATUITE

VBA EXCEL | CONDITIONS

If Then Else

Une macro devient vraiment utile quand elle sait décider : colorer un retard en rouge, ignorer une ligne vide, appliquer une remise au-delà d'un certain montant. If Then Else est l'instruction qui lui donne ce pouvoir.

Version	V1.2
Dernière mise à jour	29/10/2026
Niveau	1 / 3, Découverte
Durée de lecture	8 min
Auteur	Clément Delobel



Version en ligne
mesetudesdecas.fr/vba/if-then-else.html

01 Définition

EN UNE PHRASE

If Then Else exécute un bloc d'instructions **seulement si une condition est vraie**, et éventuellement un autre bloc si elle est fausse.

02 Syntaxe

LES TROIS FORMES

```
' Sur une seule ligne
If montant > 1000 Then remise = 0.05

' En bloc
If montant > 1000 Then
    remise = 0.05
Else
    remise = 0
End If

' Avec plusieurs cas
If montant >= 5000 Then
    remise = 0.1
ElseIf montant >= 1000 Then
    remise = 0.05
Else
    remise = 0
End If
```

VBA teste les conditions dans l'ordre et s'arrête à la **première vraie**. Placez donc la condition la plus exigeante en premier.

03 Opérateurs de comparaison et logiques

Opérateur	Signification
=	égal à
<>	différent de
> , <	supérieur, inférieur
>= , <=	supérieur ou égal, inférieur ou égal
And	les deux conditions sont vraies
Or	au moins une condition est vraie
Not	inverse la condition

04 Exemples à copier

Signaler les factures en retard

COLOSER LES RETARDS

```
Sub SignalerRetards()
    Dim i As Long
    For i = 2 To Cells(Rows.Count, 1).End(xlUp).Row
        If Cells(i, 4).Value < Date And Cells(i, 5).Value <> "Payée" Then
            Cells(i, 4).Interior.Color = RGB(252, 228, 214)
        End If
    Next i
End Sub
```

Vérifier une saisie

REFUSER UNE CELLULE VIDE OU NON NUMÉRIQUE

```
If Range("B2").Value = "" Or Not IsNumeric(Range("B2").Value) Then
    MsgBox "Saisissez un montant en B2.", vbExclamation
Exit Sub
End If
```

05 Pièges à éviter

⚠ PIÈGE

Oublier End If

Un If écrit en bloc doit se fermer par End If, sinon VBA refuse de lancer la macro : « Bloc If sans End If ».

⚠ PIÈGE

Majuscules et minuscules

Par défaut, "Paris" et "paris" sont différents pour VBA. Comparez avec `LCase(Cells(i, 1).Value) = "paris"` pour ignorer la casse.

 PIÈGE**VBA évalue toute la condition**

Avec `And`, VBA calcule les deux côtés même si le premier est faux. `If Not c Is Nothing And c.Value > 0` plante quand c vaut Nothing : séparez en deux If imbriqués.

 PIÈGE**Trop de Elself**

Au-delà de trois ou quatre cas sur la même valeur, **Select Case** est beaucoup plus lisible.

FICHE DE COURS GRATUITE

VBA EXCEL | CONDITIONS

Select Case

Quand la même valeur doit être comparée à beaucoup de cas, une série de Elself devient vite illisible. Select Case range chaque cas sur sa ligne, comme un aiguillage.

Version	V1.2
Dernière mise à jour	29/10/2026
Niveau	2 / 3, Intermédiaire
Durée de lecture	6 min
Auteur	Clément Delobel



Version en ligne
mesetudesdecas.fr/vba/select-case.html

01 Définition

EN UNE PHRASE

Select Case compare **une seule expression** à une liste de cas et exécute le bloc du **premier cas qui correspond**.

02 Syntaxe

SYNTAXE GÉNÉRALE

```
Select Case expression
  Case valeur1
    ' instructions
  Case valeur2, valeur3
    ' instructions
  Case Else
    ' si aucun cas ne correspond
End Select
```

Forme du cas	Exemple	Correspond à
Valeur unique	Case "Lille"	exactement "Lille"
Liste	Case "Lille", "Lens", "Arras"	une des valeurs
Plage	Case 10 To 19	de 10 à 19 inclus
Comparaison	Case Is >= 20	20 et plus
Sinon	Case Else	tout le reste

03 Exemples à copier

Attribuer une mention selon la note

MENTION AUTOMATIQUE

```
Function Mention(note As Double) As String
    Select Case note
        Case Is >= 16: Mention = "Très bien"
        Case Is >= 14: Mention = "Bien"
        Case Is >= 12: Mention = "Assez bien"
        Case Is >= 10: Mention = "Passable"
        Case Else:      Mention = "Ajourné"
    End Select
End Function
```

Le deux-points permet d'écrire le cas et son instruction sur la même ligne.

Régions à partir du département

PLUSIEURS VALEURS PAR CAS

```
Select Case Left(Cells(i, 3).Value, 2)
    Case "59", "62"
        Cells(i, 4).Value = "Hauts-de-France"
    Case "75", "92", "93", "94"
        Cells(i, 4).Value = "Île-de-France"
    Case Else
        Cells(i, 4).Value = "Autre"
End Select
```

💡 ASTUCE

Select Case True

Pour tester des conditions différentes, et non une seule valeur, on peut écrire `Select Case True` puis `Case montant > 1000 And client = "Pro"`. Le premier cas vrai est retenu.

04 Pièges à éviter

⚠️ PIÈGE

Un seul cas exécuté

Dès qu'un cas correspond, VBA exécute son bloc puis sort. L'ordre compte : avec `Case Is >= 10` avant `Case Is >= 16`, une note de 18 obtiendrait « Passable ».

⚠️ PIÈGE

Les décimales entre deux plages

Avec `Case 0 To 9` puis `Case 10 To 19`, la valeur 9,5 ne correspond à aucun des deux. Préférez `Case Is < 10` pour les nombres décimaux.

 PIÈGE**Pas de Case Else**

Sans Case Else, une valeur qui ne correspond à aucun cas ne déclenche rien, sans message. Ajoutez un Case Else, ne serait-ce que pour signaler la valeur inattendue.

FICHE DE COURS GRATUITE

VBA EXCEL | BOUCLES

La boucle For Next

C'est la boucle que l'on écrit le plus en VBA. Dès qu'il faut répéter la même action sur dix, cent ou dix mille lignes, For Next fait le travail à votre place. Et pour bien la comprendre, le plus efficace est encore de la regarder tourner.

Version	V1.2
Dernière mise à jour	29/10/2026
Niveau	2 / 3, Intermédiaire
Durée de lecture	9 min
Auteur	Clément Delobel



Version en ligne
mesetudesdecas.fr/vba/boucle-for-next.html

01 Définition

EN UNE PHRASE

For Next répète un bloc d'instructions **un nombre de fois connu à l'avance**, en faisant varier un compteur d'une valeur de départ à une valeur de fin.

Le compteur, souvent nommé `i`, sert à deux choses : savoir combien de tours il reste, et désigner une ligne, une colonne ou un élément différent à chaque tour. C'est ce qui permet d'écrire `Cells(i, 1)` pour traiter A1, puis A2, puis A3, sans recopier la même ligne de code.

02 Syntaxe

SYNTAXE GÉNÉRALE

```
For compteur = debut To fin [Step pas]
    ' instructions répétées à chaque tour
[Exit For]
Next [compteur]
```

Élément	Rôle
compteur	Variable numérique, déclarée de préférence <code>As Long</code> . Elle change de valeur à chaque tour.
debut	Valeur du compteur au premier tour.
fin	Dernière valeur traitée. La boucle s'arrête dès que le compteur la dépasse.
Step pas	Facultatif. De combien le compteur avance à chaque tour : 1 par défaut, 2 pour une ligne sur deux, -1 pour remonter.
Exit For	Facultatif. Sort immédiatement de la boucle, sans attendre la fin.

Élément	Rôle
Next compteur	Fin du bloc répété. Rappeler le nom du compteur est facultatif mais rend le code plus lisible.

03 Simulateur pas à pas

Cliquez sur **Pas à pas** (ou appuyez sur **F8**) pour exécuter le code une ligne à la fois, exactement comme dans l'éditeur VBA. La ligne surlignée en jaune est celle qui **va** être exécutée. Observez la variable **i** dans les Variables locales et la feuille qui se remplit.



Simulateur interactif

Cette fiche existe en version interactive : le code s'exécute ligne par ligne, les variables évoluent et la feuille se remplit en direct. Scannez le code ou rendez-vous sur la page en ligne.

mesetudesdecas.fr/vba/boucle-for-next.html

Ce que fait la boucle, tour par tour

Pour `For i = 1 To 5` avec `Cells(i, 1).Value = i` :

Tour	Valeur de i	Action	Après Next
1	1	écrit 1 en A1	i passe à 2
2	2	écrit 2 en A2	i passe à 3
3	3	écrit 3 en A3	i passe à 4
4	4	écrit 4 en A4	i passe à 5
5	5	écrit 5 en A5	i passe à 6 : 6 dépasse 5, la boucle s'arrête

💡 ASTUCE

Le même affichage dans Excel

Placez le curseur dans la macro, appuyez sur F8, puis affichez les Variables locales par le menu **Affichage**. Réduisez la fenêtre VBA pour voir la feuille réagir en même temps. Détails dans la fiche [Déboguer pas à pas](#).

04 Exemples à copier

Traiter toutes les lignes d'un tableau, jusqu'à la dernière

Le cas le plus courant : on ne sait pas combien de lignes contient le tableau, alors on calcule la **dernière ligne** avant de lancer la boucle.

CALCULER UN PRIX TTC SUR CHAQUE LIGNE

```
Sub CalculerTTC()  
    Dim i As Long, derniereLigne As Long  
  
    ' Dernière ligne remplie de la colonne A  
    derniereLigne = Cells(Rows.Count, 1).End(xlUp).Row  
  
    ' Ligne 1 = en-têtes, on commence donc à 2  
    For i = 2 To derniereLigne  
        Cells(i, 3).Value = Cells(i, 2).Value * 1.2  
    Next i  
End Sub
```

Une ligne sur deux avec Step

COLORER UNE LIGNE SUR DEUX

```
Sub UneLigneSurDeux()  
    Dim i As Long, derniereLigne As Long  
    derniereLigne = Cells(Rows.Count, 1).End(xlUp).Row  
  
    For i = 2 To derniereLigne Step 2  
        Rows(i).Interior.Color = RGB(235, 239, 255)  
    Next i  
End Sub
```

Supprimer des lignes en partant du bas

SUPPRIMER LES LIGNES DONT LA COLONNE A EST VIDE

```
Sub SupprimerVides()  
    Dim i As Long  
    For i = Cells(Rows.Count, 2).End(xlUp).Row To 2 Step -1  
        If Cells(i, 1).Value = "" Then Rows(i).Delete  
    Next i  
End Sub
```

Sortir dès qu'on a trouvé, avec Exit For

RECHERCHER UN CLIENT

```
Sub ChercherClient()  
    Dim i As Long, derniereLigne As Long, cherche As String  
    cherche = InputBox("Nom du client recherché ?")  
    derniereLigne = Cells(Rows.Count, 1).End(xlUp).Row  
  
    For i = 2 To derniereLigne  
        If Cells(i, 1).Value = cherche Then  
            MsgBox cherche & " trouvé en ligne " & i  
            Exit For ' inutile de continuer  
        End If  
    Next i  
  
    ' Si la boucle est allée au bout, i vaut derniereLigne + 1  
    If i > derniereLigne Then MsgBox cherche & " introuvable."  
End Sub
```

Deux boucles imbriquées

Une boucle pour les lignes, une autre pour les colonnes : chaque tour de la boucle extérieure déclenche la boucle intérieure en entier.

TABLE DE MULTIPLICATION 10 X 10

```
Sub TableMultiplication()  
    Dim ligne As Long, colonne As Long  
  
    For ligne = 1 To 10  
        For colonne = 1 To 10  
            Cells(ligne, colonne).Value = ligne * colonne  
        Next colonne  
    Next ligne  
End Sub
```

05 Pièges à éviter

⚠ PIÈGE

Supprimer des lignes en descendant

Si vous supprimez la ligne 3, l'ancienne ligne 4 remonte en 3... et la boucle passe directement à 4 : une ligne est sautée. Pour supprimer, on part toujours du bas avec `Step -1`.

⚠ PIÈGE

Une boucle qui ne tourne jamais

`For i = 10 To 1` sans `Step -1` ne fait aucun tour, et sans aucun message d'erreur. Quand le départ est plus grand que la fin, le pas négatif est obligatoire.

 PIÈGE**La fin est calculée une seule fois**

La valeur de fin est lue au démarrage de la boucle. Si votre macro ajoute des lignes pendant la boucle, celles-ci ne seront pas traitées.

 PIÈGE**Integer au lieu de Long**

Un compteur déclaré `As Integer` plante au-delà de 32 767 (erreur 6, dépassement de capacité). Déclarez toujours vos compteurs `As Long`.

 PIÈGE**Modifier le compteur dans la boucle**

Écrire `i = i + 1` à l'intérieur de la boucle s'ajoute au Next et décale tout. Si le pas doit changer, utilisez `Step`, ou passez à une boucle **Do While**.

06 For Next ou For Each ?**For Next**

Quand on a besoin du **numéro** : la ligne en cours, une colonne voisine, un pas particulier ou un sens de parcours inversé.

For Each

Quand on veut simplement passer sur **chaque élément** d'un ensemble : chaque cellule d'une plage, chaque feuille d'un classeur. Voir **For Each**.



FICHE DE COURS GRATUITE

VBA EXCEL | BOUCLES

La boucle For Each

Pas besoin de compteur quand on veut simplement passer sur chaque élément d'un ensemble. For Each prend les cellules, les feuilles ou les classeurs un par un, dans l'ordre, jusqu'au dernier.

Version	V1.2
Dernière mise à jour	29/10/2026
Niveau	2 / 3, Intermédiaire
Durée de lecture	7 min
Auteur	Clément Delobel



Version en ligne
mesetudesdecas.fr/vba/boucle-for-each.html

01 Définition

EN UNE PHRASE

For Each répète un bloc d'instructions **pour chaque élément d'une collection** : chaque cellule d'une plage, chaque feuille d'un classeur, chaque classeur ouvert.

02 Syntaxe

SYNTAXE GÉNÉRALE

```
Dim element As TypeDeLElement
For Each element In collection
    ' instructions sur element
Next element
```

Pour parcourir	Variable	Collection
les cellules d'une plage	<code>Dim c As Range</code>	<code>Range("A2:A100")</code>
les feuilles du classeur	<code>Dim ws As Worksheet</code>	<code>ThisWorkbook.Worksheets</code>
les classeurs ouverts	<code>Dim wb As Workbook</code>	<code>Workbooks</code>
les valeurs d'un tableau VBA	<code>Dim v As Variant</code>	le nom du tableau

03 Exemples à copier

Chaque cellule d'une plage

METTRE EN ROUGE LES MONTANTS NÉGATIFS

```
Sub SignalerNegatifs()  
    Dim c As Range  
    For Each c In Range("C2:C500")  
        If IsNumeric(c.Value) And c.Value < 0 Then  
            c.Font.Color = RGB(200, 55, 45)  
        End If  
    Next c  
End Sub
```

Chaque feuille du classeur

LISTER TOUTES LES FEUILLES

```
Sub ListerFeuilles()  
    Dim ws As Worksheet, ligne As Long  
    ligne = 1  
    For Each ws In ThisWorkbook.Worksheets  
        Worksheets("Sommaire").Cells(ligne, 1).Value = ws.Name  
        ligne = ligne + 1  
    Next ws  
End Sub
```

Seulement les cellules sélectionnées

PASSER LA SÉLECTION EN MAJUSCULES

```
Sub Majuscules()  
    Dim c As Range  
    For Each c In Selection  
        If Not c.HasFormula Then c.Value = UCase(c.Value)  
    Next c  
End Sub
```

💡 ASTUCE

L'ordre de parcours

Sur une plage de plusieurs colonnes, For Each lit ligne par ligne : A1, B1, C1, puis A2, B2, C2, et ainsi de suite.

04 Pièges à éviter

⚠️ PIÈGE

Supprimer pendant le parcours

Supprimer des lignes à l'intérieur d'un For Each sur une plage fait sauter des cellules. Pour supprimer, utilisez **For Next** en partant du bas avec Step -1.

 PIÈGE**Une plage trop large**

`For Each c In Range("A:A")` parcourt plus d'un million de cellules. Limitez la plage à la zone utile avec la **dernière ligne**.

 PIÈGE**Oublier le type de la variable**

Sans `As Range` ou `As Worksheet`, la variable est un Variant : le code fonctionne, mais l'éditeur ne propose plus la saisie semi-automatique des propriétés.

05 For Each ou For Next ?**For Each**

Plus court et plus lisible pour traiter chaque élément de la même façon.

For Next

Indispensable dès qu'on a besoin du numéro de ligne, d'un pas particulier ou d'un parcours à l'envers.

FICHE DE COURS GRATUITE

VBA EXCEL | BOUCLES

Do While **et** Do Until

Parfois, on ne sait pas combien de tours seront nécessaires : on veut continuer tant qu'il reste des données, ou jusqu'à trouver ce qu'on cherche. Do While et Do Until répètent tant qu'une condition le demande.

Version	V1.2
Dernière mise à jour	29/10/2026
Niveau	2 / 3, Intermédiaire
Durée de lecture	7 min
Auteur	Clément Delobel



Version en ligne
mesetudesdecas.fr/vba/boucle-do-while.html

01 Définition

EN UNE PHRASE

Do While répète un bloc **tant que** la condition est vraie. **Do Until** répète **jusqu'à ce que** la condition devienne vraie.

02 Les quatre formes

CONDITION AU DÉBUT OU À LA FIN

```
' Test au début : peut ne faire aucun tour
Do While condition
    ' instructions
Loop

Do Until condition
    ' instructions
Loop

' Test à la fin : fait toujours au moins un tour
Do
    ' instructions
Loop While condition

Do
    ' instructions
Loop Until condition
```

Do While x <> "" et Do Until x = "" font exactement la même chose : choisissez la forme qui se lit le plus naturellement.

03 Exemples à copier

Avancer jusqu'à la première cellule vide

NUMÉROTÉ LES LIGNES REMPLIES

```
Sub Numeroter()
    Dim i As Long
    i = 2
    Do While Cells(i, 2).Value <> ""
        Cells(i, 1).Value = i - 1
        i = i + 1          ' indispensable : sinon la boucle ne s'arrête jamais
    Loop
End Sub
```

Redemander tant que la saisie est incorrecte

EXIGER UN NOMBRE

```
Sub DemanderQuantite()
    Dim saisie As String
    Do
        saisie = InputBox("Quantité (nombre entier) ?")
        If saisie = "" Then Exit Sub
    Loop Until IsNumeric(saisie)
    Range("B2").Value = CLng(saisie)
End Sub
```

Sortir en cours de route avec Exit Do

CHERCHER LE PREMIER MONTANT SUPÉRIEUR À 10 000

```
Dim i As Long
i = 2
Do Until Cells(i, 3).Value = ""
    If Cells(i, 3).Value > 10000 Then
        MsgBox "Premier gros montant en ligne " & i
        Exit Do
    End If
    i = i + 1
Loop
```

04 Pièges à éviter

⚠ PIÈGE

La boucle infinie

Si la condition ne change jamais, la macro tourne sans fin et Excel semble figé. Appuyez sur **Échap** ou **Ctrl + Pause** pour l'interrompre, puis vérifiez que la variable testée évolue bien à chaque tour.

 PIÈGE**Une cellule vide au milieu**

Une boucle qui s'arrête à la première cellule vide s'arrête aussi au premier trou dans les données. Si le tableau peut contenir des lignes vides, calculez plutôt la **dernière ligne** et utilisez For Next.

 PIÈGE**While Wend**

Vous croiserez `While ... Wend` dans de vieux codes. Elle fonctionne, mais ne permet pas de sortir avec Exit : préférez Do While.

FICHE DE COURS GRATUITE

VBA EXCEL | OBJETS EXCEL

Range et Cells

Presque toutes les macros finissent par lire ou écrire dans une cellule. Range et Cells sont les deux façons de désigner l'endroit exact où agir : l'une avec une adresse, l'autre avec des numéros.

Version	V1.2
Dernière mise à jour	29/10/2026
Niveau	1 / 3, Découverte
Durée de lecture	9 min
Auteur	Clément Delobel



Version en ligne
mesetudesdecas.fr/vba/range-et-cells.html

01 Définition

EN UNE PHRASE

Range désigne une cellule ou une plage par son **adresse** ("B3", "A1:D10"). **Cells** désigne une cellule par ses **numéros** de ligne et de colonne, ce qui la rend idéale dans une boucle.

Range("B3")

Lisible, parfait pour une adresse fixe ou une plage nommée.

Cells(3, 2)

Ligne 3, colonne 2 : la même cellule B3, mais les numéros peuvent être des variables.

02 Désigner des cellules

Écriture	Désigne
Range("A1")	la cellule A1
Range("A1:D10")	la plage A1 à D10
Range("TauxTVA")	la plage nommée TauxTVA
Cells(5, 3)	la cellule C5 (ligne 5, colonne 3)
Cells(5, "C")	la même cellule, colonne en lettre
Range(Cells(2, 1), Cells(10, 4))	la plage A2:D10
Rows(4) , Columns("B")	une ligne ou une colonne entière

03 Lire et écrire

VALUE, FORMULA ET EFFACEMENT

```

Range("B2").Value = 150           ' écrire une valeur
total = Range("B10").Value        ' lire une valeur
Range("B10").Formula = "=SUM(B2:B9)" ' formule en anglais, séparateur virgule
Range("B11").FormulaLocal = "=MOYENNE(B2:B9)" ' formule en français
Range("A2:F500").ClearContents    ' efface les valeurs, garde la mise en forme
Range("A2:F500").Clear             ' efface tout, mise en forme comprise

```

04 Se déplacer avec Offset et Resize

- `Offset(lignes, colonnes)` décale une référence : `Range("B2").Offset(1, 0)` désigne B3, `Offset(0, 2)` désigne D2.
- `Resize(lignes, colonnes)` change la taille : `Range("A2").Resize(10, 3)` désigne A2:C11.

ÉCRIRE À DROITE DE CHAQUE CELLULE

```

Dim c As Range
For Each c In Range("A2:A20")
    c.Offset(0, 1).Value = Len(c.Value) ' nombre de caractères en colonne B
Next c

```

05 Inutile de sélectionner

L'enregistreur écrit `Range("A1").Select` puis `Selection.Value = ...`. En VBA écrit à la main, on agit directement : `Range("A1").Value = ...`. Le code est plus court, plus rapide, et ne dépend plus de la cellule sélectionnée.

06 Pièges à éviter

⚠ PIÈGE

La feuille active par défaut

`Range("A1")` sans préciser de feuille désigne la feuille **active** au moment de l'exécution. Qualifiez vos références : `Worksheets("Ventes").Range("A1")`.

⚠ PIÈGE

Cells non qualifié dans Range

`Worksheets("Ventes").Range(Cells(1, 1), Cells(5, 2))` plante si Ventes n'est pas active, car les Cells désignent la feuille active : c'est l'**erreur 1004**. Écrivez `ws.Range(ws.Cells(1, 1), ws.Cells(5, 2))`.

⚠ PIÈGE

Formula attend l'anglais

`.Formula = "=SOMME(A1:A5)"` provoque une erreur. Avec `.Formula`, on écrit les fonctions en anglais et on sépare par des virgules ; avec `.FormulaLocal`, en français avec des points-virgules.

FICHE DE COURS GRATUITE

VBA EXCEL | OBJETS EXCEL

Trouver la dernière ligne

Un tableau grossit chaque mois : la macro doit trouver seule où il s'arrête. Cette ligne de code est sans doute la plus recopiée de tout le VBA, et elle mérite d'être bien comprise.

Version	V1.2
Dernière mise à jour	29/10/2026
Niveau	2 / 3, Intermédiaire
Durée de lecture	6 min
Auteur	Clément Delobel



Version en ligne
mesetudesdecas.fr/vba/derniere-ligne.html

01 Définition

EN UNE PHRASE

Pour trouver la dernière ligne remplie, on part **tout en bas de la feuille** et on **remonte** jusqu'à la première cellule non vide, exactement comme `Ctrl + Flèche haut`.

LA FORMULE À CONNAÎTRE PAR CŒUR

```
derniereLigne = Cells(Rows.Count, 1).End(xlUp).Row
```

Morceau	Signification
<code>Rows.Count</code>	le nombre de lignes de la feuille (1 048 576)
<code>Cells(Rows.Count, 1)</code>	la toute dernière cellule de la colonne A
<code>.End(xlUp)</code>	remonter jusqu'à une cellule remplie
<code>.Row</code>	le numéro de ligne de cette cellule

02 Les variantes utiles

DERNIÈRE COLONNE, PREMIÈRE LIGNE LIBRE, FEUILLE PRÉCISE

```
' Dernière colonne remplie de la ligne 1
derniereColonne = Cells(1, Columns.Count).End(xlToLeft).Column

' Première ligne libre, pour ajouter une donnée
ligneLibre = Cells(Rows.Count, 1).End(xlUp).Row + 1

' Sur une feuille précise, sans dépendre de la feuille active
With Worksheets("Ventes")
    derniereLigne = .Cells(.Rows.Count, 1).End(xlUp).Row
End With
```

ASTUCE

Choisir la bonne colonne

Prenez une colonne **toujours remplie**, comme un identifiant ou une date. Si la colonne testée a des trous en bas du tableau, le résultat sera trop court.

03 Les autres méthodes

Méthode	Avantage	Limite
<code>End(xlUp)</code>	fiable, la plus utilisée	ne regarde qu'une colonne
<code>End(xlDown)</code>	part du haut	s'arrête à la première cellule vide
<code>UsedRange</code>	toute la zone utilisée	compte aussi les cellules seulement mises en forme
<code>CurrentRegion</code>	le bloc autour d'une cellule	s'arrête à la première ligne ou colonne vide
<code>ListObject</code>	tableau structuré, toujours juste	nécessite un tableau (Ctrl + T)

AVEC UN TABLEAU STRUCTURÉ

```
Dim t As ListObject
Set t = Worksheets("Ventes").ListObjects("tblVentes")
MsgBox t.ListRows.Count & " lignes de données"
```

04 Pièges à éviter

PIÈGE

Une feuille vide

Si la colonne est vide, `End(xlUp)` remonte jusqu'en ligne 1 et renvoie 1. Testez ce cas avant de lancer une boucle à partir de la ligne 2.

 PIÈGE**Les filtres actifs**

Sur une feuille filtrée, les lignes masquées peuvent fausser le résultat. Retirez le filtre avant le calcul, ou travaillez avec un tableau structuré.

 PIÈGE**Partir du haut**

`Range("A1").End(xlDown)` s'arrête au premier trou dans les données. Partir du bas est plus sûr.

FICHE DE COURS GRATUITE

VBA EXCEL | OBJETS EXCEL

Feuilles et classeurs

Dès qu'une macro travaille sur plusieurs feuilles ou plusieurs fichiers, il faut lui dire précisément où agir. Bien désigner feuilles et classeurs évite les macros qui écrivent au mauvais endroit.

Version	V1.2
Dernière mise à jour	29/10/2026
Niveau	2 / 3, Intermédiaire
Durée de lecture	8 min
Auteur	Clément Delobel



Version en ligne
mesetudesdecas.fr/vba/worksheets-et-workbooks.html

01 Définition

EN UNE PHRASE

Excel range ses objets comme des poupées russes : l'**Application** contient des **classeurs** (Workbooks), qui contiennent des **feuilles** (Worksheets), qui contiennent des **cellules** (Range).

UNE RÉFÉRENCE COMPLÈTE

```
Workbooks("Ventes.xlsx").Worksheets("Janvier").Range("B2").Value = 100
```

02 Désigner une feuille

Écriture	Désigne
<code>Worksheets("Janvier")</code>	la feuille nommée Janvier (nom de l'onglet)
<code>Worksheets(1)</code>	la première feuille, selon l'ordre des onglets
<code>Feuil1</code>	la feuille par son nom de code, visible dans l'éditeur
<code>ActiveSheet</code>	la feuille affichée à l'écran
<code>Sheets("Graph1")</code>	comme Worksheets, mais inclut les feuilles graphiques

💡 ASTUCE

Le nom de code résiste au renommage

Dans l'éditeur, chaque feuille a un nom de code (propriété Name dans la fenêtre Propriétés). Si l'utilisateur renomme l'onglet, `Feuil1.Range("A1")` continue de fonctionner, alors que `Worksheets("Janvier")` provoque l'**erreur 9**.

03 ThisWorkbook ou ActiveWorkbook ?

ThisWorkbook

Le classeur qui **contient la macro**. Toujours le même, quel que soit le fichier affiché.

ActiveWorkbook

Le classeur **affiché à l'écran**. Il change dès qu'on ouvre ou qu'on active un autre fichier.

04 Exemples à copier

Ajouter et renommer une feuille

CRÉER UNE FEUILLE SYNTHÈSE À LA FIN

```
Sub AjouterSynthese()  
    Dim ws As Worksheet  
    Set ws = ThisWorkbook.Worksheets.Add(After:=ThisWorkbook.Worksheets(ThisWorkbook.Worksheets.Count))  
    ws.Name = "Synthèse"  
End Sub
```

Supprimer une feuille sans message

SUPPRIMER LA FEUILLE BROUILLON

```
Application.DisplayAlerts = False  
ThisWorkbook.Worksheets("Brouillon").Delete  
Application.DisplayAlerts = True
```

Ouvrir un fichier, copier, fermer

RÉCUPÉRER LES DONNÉES D'UN AUTRE CLASSEUR

```
Sub ImporterDonnees()  
    Dim wbSource As Workbook  
    Set wbSource = Workbooks.Open("C:\Exports\export_ventes.xlsx")  
  
    wbSource.Worksheets(1).Range("A1").CurrentRegion.Copy _  
        Destination:=ThisWorkbook.Worksheets("Import").Range("A1")  
  
    wbSource.Close SaveChanges:=False  
End Sub
```

Le trait de soulignement en fin de ligne permet de couper une instruction trop longue sur deux lignes.

05 Pièges à éviter

⚠ PIÈGE

Un nom d'onglet mal orthographié

Une espace en trop ou un accent oublié dans `Worksheets("Janvier ")` déclenche l'**erreur 9**.

 PIÈGE**Compter sur ActiveWorkbook**

Après un `Workbooks.Open`, le classeur actif devient le fichier ouvert. Utilisez des variables (`Set wbSource = ...`) et `ThisWorkbook` pour ne jamais vous tromper de fichier.

 PIÈGE**Un nom de feuille déjà pris**

Renommer une feuille avec un nom qui existe déjà provoque l'**erreur 1004**. Il en va de même pour les caractères interdits : `/ \ ? * [ ] : .`.

FICHE DE COURS GRATUITE

VBA EXCEL | OBJETS EXCEL

With End With

Quand on modifie cinq propriétés de la même plage, recopier cinq fois son adresse alourdit le code. With End With nomme l'objet une seule fois, et toutes les lignes du bloc s'y rapportent.

Version	V1.2
Dernière mise à jour	29/10/2026
Niveau	2 / 3, Intermédiaire
Durée de lecture	5 min
Auteur	Clément Delobel



Version en ligne
mesetudesdecas.fr/vba/with-end-with.html

01 Définition

EN UNE PHRASE

With End With **désigne un objet une seule fois** ; à l'intérieur du bloc, chaque instruction qui commence par un **point** s'applique à cet objet.

02 Avant, après

Sans With

```
Range("A1:F1").Font.Bold = True puis
Range("A1:F1").Font.Size = 12 puis
Range("A1:F1").Interior.Color = ... : l'adresse
est répétée à chaque ligne.
```

Avec With

L'adresse est écrite une fois, chaque ligne commence par un point. Plus court, plus lisible, et plus rapide.

METTRE EN FORME UN EN-TÊTE

```
Sub FormaterEntete()
    With Worksheets("Ventes").Range("A1:F1")
        .Font.Bold = True
        .Font.Size = 12
        .Interior.Color = RGB(46, 82, 255)
        .Font.Color = RGB(255, 255, 255)
        .HorizontalAlignment = xlCenter
    End With
End Sub
```

03 Imbriquer des blocs With

UN WITH DANS UN WITH

```
With Worksheets("Ventes")  
    .Range("A1").Value = "Rapport"  
    With .Range("A2:F2").Font  
        .Bold = True  
        .Italic = True  
    End With  
    .Columns("A:F").AutoFit  
End With
```

💡 ASTUCE

Avec une feuille

With Worksheets("Ventes") est très pratique pour qualifier toutes les références d'une macro : .Range , .Cells , .Rows.Count désignent alors la feuille Ventes, et non la feuille active.

04 Pièges à éviter

⚠️ PIÈGE

Le point oublié

À l'intérieur d'un With Worksheets("Ventes") , écrire Cells(1, 1) sans point désigne la feuille **active**, pas Ventes. Le code ne plante pas, il écrit simplement au mauvais endroit.

⚠️ PIÈGE

Oublier End With

Chaque With doit être fermé. Sans End With, VBA refuse de lancer la macro : « With sans End With ».

FICHE DE COURS GRATUITE

VBA EXCEL | FORMULAIRES

Créer un premier UserForm

Un formulaire de saisie, c'est la meilleure façon d'éviter les erreurs dans un tableau partagé : l'utilisateur remplit des champs propres, la macro contrôle et range tout au bon endroit. Voici comment construire le premier, de bout en bout.

Version	V1.2
Dernière mise à jour	29/10/2026
Niveau	3 / 3, Avancé
Durée de lecture	10 min
Auteur	Clément Delobel



Version en ligne
mesétudesdecas.fr/vba/
userform-premier-formu
laire.html

01 Définition

EN UNE PHRASE

Un UserForm est une **fenêtre personnalisée** que vous dessinez dans l'éditeur VBA : zones de texte, listes, cases à cocher et boutons, reliés à du code.

02 Créer le formulaire

- Ouvrez l'éditeur avec `Alt + F11`, puis **Insertion > UserForm**.
- La **Boîte à outils** s'affiche. Si ce n'est pas le cas : **Affichage > Boîte à outils**.
- Dessinez vos contrôles : un intitulé (Label) et une zone de texte (TextBox) pour le nom, une liste déroulante (ComboBox) pour la ville, deux boutons (CommandButton).
- Dans la fenêtre **Propriétés** (`F4`), renommez chaque élément : propriété **Name** pour le code, propriété **Caption** pour le texte affiché.

💡 ASTUCE

Nommer avec un préfixe

`txtNom`, `cboVille`, `btnValider`, `btnAnnuler` : le préfixe indique le type de contrôle, et le code devient lisible sans retourner voir le formulaire.

Contrôle	Name	Caption
UserForm	frmClient	Nouveau client
TextBox	txtNom	(aucune)
ComboBox	cboVille	(aucune)
CommandButton	btnValider	Valider

Contrôle	Name	Caption
CommandButton	btnAnnuler	Annuler

03 Ouvrir le formulaire

Le formulaire s'affiche avec la méthode **Show**, depuis une macro classique. Reliez cette macro à un bouton dans la feuille (**Développeur > Insérer > Bouton**).

DANS UN MODULE STANDARD

```
Sub OuvrirFormulaire()  
    frmClient.Show  
End Sub
```

04 Préparer le formulaire à l'ouverture

Double-cliquez sur le fond du formulaire pour ouvrir son code, puis choisissez l'événement **Initialize** : il s'exécute juste avant l'affichage.

REEMPLIR LA LISTE DES VILLES

```
Private Sub UserForm_Initialize()  
    With Me.cboVille  
        .AddItem "Lille"  
        .AddItem "Lens"  
        .AddItem "Arras"  
        .AddItem "Valenciennes"  
    End With  
End Sub
```

Me désigne le formulaire lui-même.

05 Le bouton Valider

Double-cliquez sur le bouton Valider dans le formulaire : l'éditeur crée la procédure de son événement **Click**.

CONTRÔLER, ÉCRIRE, FERMER

```
Private Sub btnValider_Click()  
    Dim ligne As Long  
  
    ' 1. Contrôler la saisie  
    If Trim(Me.txtNom.Value) = "" Then  
        MsgBox "Le nom est obligatoire.", vbExclamation  
        Me.txtNom.SetFocus  
        Exit Sub  
    End If  
  
    ' 2. Écrire dans la première ligne libre  
    With Worksheets("Clients")  
        ligne = .Cells(.Rows.Count, 1).End(xlUp).Row + 1  
        .Cells(ligne, 1).Value = Trim(Me.txtNom.Value)  
        .Cells(ligne, 2).Value = Me.cboVille.Value  
        .Cells(ligne, 3).Value = Date  
    End With  
  
    ' 3. Fermer le formulaire  
    Unload Me  
End Sub  
  
Private Sub btnAnnuler_Click()  
    Unload Me  
End Sub
```

06 Pièges à éviter

⚠ PIÈGE

Unload ou Hide

`Unload Me` ferme le formulaire et efface ses valeurs. `Me.Hide` le cache seulement : à la prochaine ouverture, les anciennes saisies sont encore là.

⚠ PIÈGE

Le code au mauvais endroit

Le code des boutons se trouve dans le module du formulaire (double-clic sur le contrôle), pas dans un module standard. La macro qui fait `frmClient.Show`, elle, va dans un module standard.

⚠ PIÈGE

Renommer après avoir codé

Si vous renommez `CommandButton1` en `btnValider` après avoir écrit son code, la procédure `CommandButton1_Click` ne sera plus jamais appelée. Renommez d'abord, codez ensuite.

⚠ PIÈGE

Tout est du texte

Une TextBox renvoie toujours du texte. Pour un montant ou une quantité, vérifiez avec `IsNumeric` puis convertissez avec `Cdbl` ou `CLng` avant d'écrire dans la feuille.

FICHE DE COURS GRATUITE

VBA EXCEL | FORMULAIRES

Les contrôles d'un formulaire

Un bon formulaire guide l'utilisateur : une liste quand les réponses sont connues, une case à cocher pour un oui ou non, des boutons d'option pour un choix unique. Voici chaque contrôle, à quoi il sert et comment le lire en VBA.

Version	V1.2
Dernière mise à jour	29/10/2026
Niveau	3 / 3, Avancé
Durée de lecture	9 min
Auteur	Clément Delobel



Version en ligne
mesetudesdecas.fr/vba/contrôles-userform.html

01 Définition

EN UNE PHRASE

Un contrôle est un **élément interactif** placé sur un UserForm. Chaque type a ses propriétés, dont la plus utilisée est **Value** : ce que l'utilisateur a saisi ou choisi.

02 Quel contrôle pour quel besoin ?

Contrôle	Préfixe	Usage	Propriété clé
Label	lbl	afficher un intitulé ou un message	Caption
TextBox	txt	saisie libre : nom, montant, commentaire	Value
ComboBox	cbo	liste déroulante, un seul choix	Value, ListIndex
ListBox	lst	liste visible, un ou plusieurs choix	Value, Selected
CheckBox	chk	oui ou non	Value (True / False)
OptionButton	opt	un choix parmi plusieurs	Value (True / False)
Frame	fra	regrouper des contrôles	Caption
CommandButton	btn	lancer une action	Caption

03 Remplir une liste

TROIS FAÇONS DE REMPLIR UNE COMBOBOX

```
Private Sub UserForm_Initialize()
    ' 1. Élément par élément
    Me.cboStatut.AddItem "En cours"
    Me.cboStatut.AddItem "Terminé"

    ' 2. D'un coup, depuis une plage
    Me.cboVille.List = Worksheets("Listes").Range("A2:A30").Value

    ' 3. Liée à une plage (propriété RowSource)
    Me.lstProduits.RowSource = "Listes!B2:B50"
End Sub
```

💡 ASTUCE

Interdire la saisie libre

Par défaut, une ComboBox accepte aussi un texte tapé au clavier. Pour obliger à choisir dans la liste, réglez sa propriété **Style** sur `fmStyleDropDownList`.

04 Lire les choix

CHECKBOX, OPTIONBUTTON ET COMBOBOX

```
If Me.chkNewsletter.Value = True Then
    Cells(ligne, 5).Value = "Oui"
End If

If Me.optParticulier.Value Then
    Cells(ligne, 6).Value = "Particulier"
ElseIf Me.optProfessionnel.Value Then
    Cells(ligne, 6).Value = "Professionnel"
End If

If Me.cboVille.ListIndex = -1 Then
    MsgBox "Choisissez une ville." ' -1 = rien de sélectionné
End If
```

05 Une ListBox à sélection multiple

Réglez la propriété **MultiSelect** sur `fmMultiSelectMulti`, puis parcourez la liste avec une **boucle For Next** en testant `Selected`.

RÉCUPÉRER TOUS LES ÉLÉMENTS COCHÉS

```
Dim i As Long, choix As String
For i = 0 To Me.lstProduits.ListCount - 1
    If Me.lstProduits.Selected(i) Then
        choix = choix & Me.lstProduits.List(i) & ", "
    End If
Next i
```

La numérotation d'une liste commence à **0** : le premier élément est `List(0)`.

06 Pièges à éviter

⚠ PIÈGE

Des **OptionButton** qui ne s'excluent pas

Tous les boutons d'option d'un formulaire forment un seul groupe. Pour avoir deux questions séparées, placez chaque groupe dans un **Frame**, ou donnez-leur une propriété **GroupName** différente.

⚠ PIÈGE

RowSource sans nom de feuille

`RowSource = "A2:A30"` lit la feuille active du moment. Indiquez toujours la feuille : `"Listes!A2:A30"`.

⚠ PIÈGE

Value d'une **ListBox** multiple

En sélection multiple, `Value` ne renvoie rien d'utile : il faut parcourir les éléments et tester `Selected(i)`.

FICHE DE COURS GRATUITE

VBA EXCEL | DÉBOGAGE

Débugger pas à pas (F8)

Une macro ne fait pas ce que vous attendez ? Plutôt que de relire le code dix fois, regardez-le s'exécuter. Le débogueur de l'éditeur VBA permet d'avancer ligne par ligne et de voir la valeur de chaque variable à chaque instant.

Version	V1.2
Dernière mise à jour	29/10/2026
Niveau	2 / 3, Intermédiaire
Durée de lecture	8 min
Auteur	Clément Delobel



Version en ligne
mesetudesdecas.fr/vba/debogage-pas-a-pas.htm
1

01 Définition

EN UNE PHRASE

Débugger, c'est **exécuter la macro au ralenti** pour observer ce qu'elle fait vraiment : quelle ligne s'exécute, quelle valeur prend chaque variable, à quel moment le résultat dévie.

02 Les raccourcis essentiels

Touche	Action
F8	exécuter la ligne suivante (pas à pas détaillé)
Maj + F8	exécuter une procédure appelée d'un coup (pas à pas principal)
Ctrl + Maj + F8	terminer la procédure en cours et revenir à l'appelante
Ctrl + F8	exécuter jusqu'à la ligne où se trouve le curseur
F9	poser ou retirer un point d'arrêt
F5	reprendre l'exécution normale
Ctrl + G	afficher la fenêtre Exécution

03 Le pas à pas

Placez le curseur dans la macro et appuyez sur **F8** : la première ligne se surligne en jaune. Cette ligne jaune est celle qui **va** être exécutée. À chaque F8, elle avance d'une ligne. La fiche **For Next** propose un simulateur qui reproduit exactement ce fonctionnement.

 ASTUCE**Survoler une variable**

En mode pas à pas, laissez la souris sur un nom de variable : une info-bulle affiche sa valeur actuelle.

04 Les points d'arrêt

Sur une longue macro, avancer ligne par ligne depuis le début est fastidieux. Cliquez dans la marge grise à gauche d'une ligne (ou **F9**) : un point rouge apparaît. Lancez la macro avec **F5**, elle s'exécute normalement puis **s'arrête juste avant** cette ligne.

05 Les fenêtres d'observation

- **Variables locales** (menu Affichage) : toutes les variables de la procédure, leur valeur et leur type, mises à jour à chaque pas.
- **Exécution** (**Ctrl + G**) : affiche ce que vous écrivez avec `Debug.Print`, et permet de taper `? nomVariable` puis Entrée pour lire une valeur.
- **Espions** : surveillez une expression précise, et demandez l'arrêt quand elle change ou devient vraie.

TRACER UNE BOUCLE SANS L'INTERROMPRE

```
For i = 2 To derniereLigne  
    Debug.Print "Ligne " & i & " : " & Cells(i, 1).Value  
Next i
```

 ASTUCE**Déplacer la flèche jaune**

En mode arrêt, vous pouvez faire glisser la flèche jaune vers une autre ligne pour la réexécuter ou en sauter une. Pratique pour tester une correction sans tout relancer.

06 Pièges à éviter PIÈGE**Rester en mode arrêt**

Tant que la macro est en pause (ligne jaune visible), Excel attend. Pensez à cliquer sur le bouton **Réinitialiser** (carré bleu) pour arrêter proprement.

 PIÈGE**Oublier une instruction Stop**

L'instruction `Stop` met la macro en pause comme un point d'arrêt, mais elle reste dans le code. Supprimez-la avant de livrer le fichier.

 PIÈGE**Déboguer sur les vraies données**

Le pas à pas exécute réellement chaque ligne : une suppression est une vraie suppression. Travaillez sur une copie du fichier.

FICHE DE COURS GRATUITE

VBA EXCEL | DÉBOGAGE

On Error et gestion d'erreurs

Un fichier absent, une feuille renommée, une saisie inattendue : même une macro bien écrite peut rencontrer une erreur. Plutôt que d'afficher la fenêtre grise de VBA à l'utilisateur, on intercepte l'erreur et on réagit proprement.

Version	V1.2
Dernière mise à jour	29/10/2026
Niveau	3 / 3, Avancé
Durée de lecture	8 min
Auteur	Clément Delobel



Version en ligne
mesetudesdecas.fr/vba/on-error.html

01 Définition

EN UNE PHRASE

On Error indique à VBA quoi faire quand une erreur survient : sauter à un gestionnaire d'erreur, passer à la ligne suivante, ou revenir au comportement normal.

Instruction	Effet
<code>On Error GoTo Etiquette</code>	en cas d'erreur, sauter à la ligne Etiquette:
<code>On Error Resume Next</code>	ignorer l'erreur et passer à la ligne suivante
<code>On Error GoTo 0</code>	revenir au comportement normal (arrêt sur erreur)

02 La structure recommandée

UN GESTIONNAIRE D'ERREUR PROPRE

```
Sub ImporterFichier()
    On Error GoTo Erreur

    Workbooks.Open "C:\Exports\ventes.xlsx"
    ' ... suite du traitement ...

    Exit Sub          ' indispensable : on ne passe pas dans le gestionnaire si tout va bien

Erreur:
    MsgBox "Import impossible." & vbCrLf & _
        "Erreur " & Err.Number & " : " & Err.Description, vbCritical
End Sub
```

L'objet **Err** décrit la dernière erreur : `Err.Number` donne son numéro, `Err.Description` son libellé.

03 On Error Resume Next, avec modération

Elle sert à tester une instruction qui peut échouer, puis à vérifier le résultat. Remettez toujours le comportement normal juste après.

VÉRIFIER QU'UNE FEUILLE EXISTE

```
Function FeuilleExiste(nom As String) As Boolean
    Dim ws As Worksheet
    On Error Resume Next
    Set ws = ThisWorkbook.Worksheets(nom)
    On Error GoTo 0
    FeuilleExiste = Not ws Is Nothing
End Function
```

04 Reprendre après l'erreur

Dans le gestionnaire, **Resume** réexécute la ligne qui a échoué (après correction), et **Resume Next** reprend à la ligne qui suit. Utilisez-les seulement si vous avez corrigé la cause, sinon la macro tourne en rond.

05 Pièges à éviter

⚠ PIÈGE

Oublier Exit Sub

Sans **Exit Sub** avant l'étiquette, la macro termine son travail... puis passe dans le gestionnaire et affiche un message d'erreur alors que tout s'est bien passé.

⚠ PIÈGE

Resume Next sur toute la macro

Placé en tête de macro et jamais annulé, **On Error Resume Next** masque toutes les erreurs : la macro produit des résultats faux sans que personne ne le sache.

⚠ PIÈGE

Gérer avant de comprendre

Pendant le développement, laissez les erreurs s'afficher et utilisez le **pas à pas** pour les comprendre. Le gestionnaire se pose à la fin, pour les erreurs que vous ne pouvez pas éviter.

FICHE DE COURS GRATUITE

VBA EXCEL | ERREURS

Erreur 1004

C'est l'erreur VBA la plus fréquente, et la plus frustrante, car son message ne dit presque rien. « Erreur définie par l'application ou par l'objet » signifie simplement qu'Excel a refusé une action. Voici les causes les plus courantes, et comment les corriger.

Version	V1.2
Dernière mise à jour	29/10/2026
Niveau	2 / 3, Intermédiaire
Durée de lecture	6 min
Auteur	Clément Delobel



Version en ligne
mesetudesdecas.fr/vba/erreur-1004.html

01 Le message

EN UNE PHRASE

Erreur d'exécution 1004 : Excel ne peut pas réaliser l'action demandée sur un objet (une plage, une feuille, un classeur). Le libellé exact varie selon la cause.

Cliquez sur **Débogage** : la ligne fautive est surlignée en jaune. Comparez-la aux causes ci-dessous.

02 Les causes les plus fréquentes

Ligne en cause	Pourquoi	Correction
<code>Worksheets("Ventes").Range(Cells(1, 1), Cells(5, 2))</code>	les Cells désignent la feuille active, pas Ventes	qualifier : <code>ws.Cells(1, 1)</code>
<code>Worksheets("Ventes").Range("A1").Select</code>	on ne sélectionne pas sur une feuille inactive	ne pas sélectionner, agir directement
<code>.Formula = "=SOMME(A1:A5)"</code>	Formula attend l'anglais	<code>"=SUM(A1:A5)"</code> ou <code>.FormulaLocal</code>
<code>ws.Name = "Janvier"</code>	ce nom de feuille existe déjà	vérifier ou choisir un autre nom
<code>Cells(0, 1)</code>	la ligne 0 n'existe pas	vérifier le calcul du numéro de ligne
<code>Workbooks.Open "C:\...\fichier.xlsx"</code>	chemin ou nom de fichier introuvable	vérifier avec <code>Dir(chemin) <> ""</code>
écriture dans une cellule	la feuille est protégée	<code>ws.Unprotect</code> avant, <code>ws.Protect</code> après

03 Exemple corrigé

LA VERSION QUI PLANTE, PUIS LA VERSION JUSTE

```
' Plante si la feuille Ventes n'est pas active
Worksheets("Ventes").Range(Cells(2, 1), Cells(10, 4)).ClearContents

' Fonctionne quelle que soit la feuille active
Dim ws As Worksheet
Set ws = Worksheets("Ventes")
ws.Range(ws.Cells(2, 1), ws.Cells(10, 4)).ClearContents
```

💡 ASTUCE

Un réflexe qui évite la plupart des 1004

Déclarez une variable feuille (`Set ws = ...`) en début de macro et préfixez toutes vos références avec `ws.` ou un bloc `With`. La macro ne dépend plus de ce qui est affiché à l'écran.

04 Pièges à éviter

⚠️ PIÈGE

Corriger au hasard

Ne relancez pas la macro en modifiant des lignes au hasard : lisez la ligne jaune, identifiez l'objet concerné (plage, feuille, classeur), puis vérifiez qu'il existe et qu'il est accessible.

⚠️ PIÈGE

Masquer l'erreur

Un `On Error Resume Next` fait disparaître le message mais pas le problème. Corrigez la cause.

FICHE DE COURS GRATUITE

VBA EXCEL | ERREURS

Erreur 91

L'erreur 91 apparaît quand la macro utilise un objet qui n'existe pas encore, ou plus. Dans neuf cas sur dix, c'est un Set oublié ou une recherche qui n'a rien trouvé.

Version	V1.2
Dernière mise à jour	29/10/2026
Niveau	2 / 3, Intermédiaire
Durée de lecture	5 min
Auteur	Clément Delobel



Version en ligne
mesétudesdecas.fr/vba/
erreur-91.html

01 Le message

EN UNE PHRASE

Erreur d'exécution 91 : Variable objet ou variable de bloc With non définie. La variable devrait désigner un objet (cellule, feuille, classeur), mais elle est vide : elle vaut **Nothing**.

02 Les causes les plus fréquentes

Cause	Exemple	Correction
Set oublié	<code>ws = Worksheets("Ventes")</code>	<code>Set ws = Worksheets("Ventes")</code>
Find ne trouve rien	<code>Range("A:A").Find("Dupont").Row</code>	tester <code>Is Nothing</code> avant d'utiliser le résultat
Objet jamais affecté	<code>Dim c As Range</code> puis <code>c.Value = 1</code>	affecter avec Set avant l'usage

03 Le cas classique : Find

`Find` renvoie la cellule trouvée... ou **Nothing** si la valeur est absente. Utiliser directement le résultat déclenche l'erreur 91 dès que la recherche échoue.

CHERCHER SANS PLANTER

```
Sub TrouverClient()  
    Dim trouve As Range  
    Set trouve = Worksheets("Clients").Range("A:A").Find(What:="Dupont", LookAt:=xlWhole)  
  
    If trouve Is Nothing Then  
        MsgBox "Client introuvable."  
    Else  
        MsgBox "Client trouvé en ligne " & trouve.Row  
    End If  
End Sub
```

 ASTUCE**Is Nothing**

On ne compare pas un objet avec `=`. Pour savoir si une variable objet est vide, on écrit `If maVariable Is Nothing Then`.

04 **Pièges à éviter** PIÈGE**Set pour les objets, pas pour les valeurs**

`Set total = Range("B10")` range la cellule ; `total = Range("B10").Value` range sa valeur. Le type déclaré (Range ou Double) doit correspondre.

FICHE DE COURS GRATUITE

VBA EXCEL | ERREURS

Erreur 13

L'erreur 13 signale que VBA a reçu une donnée du mauvais type : du texte là où il attend un nombre, une valeur d'erreur Excel là où il attend une date. Elle vient presque toujours des données, rarement du code lui-même.

Version	V1.2
Dernière mise à jour	29/10/2026
Niveau	2 / 3, Intermédiaire
Durée de lecture	5 min
Auteur	Clément Delobel



Version en ligne
mesetudesdecas.fr/vba/erreur-13.html

01 Le message

EN UNE PHRASE

Erreur d'exécution 13 : Incompatibilité de type. La valeur ne peut pas être rangée dans la variable ou utilisée dans le calcul, parce que son type ne correspond pas.

02 Les causes les plus fréquentes

Cause	Exemple	Correction
Texte dans une variable numérique	<code>Dim q As Long</code> puis <code>q = "douze"</code>	vérifier avec <code>IsNumeric</code>
Cellule contenant une erreur	une cellule affiche <code>#N/A</code> et on fait <code>total = total + c.Value</code>	tester <code>IsError(c.Value)</code>
Date mal formée	<code>CDate("31/02/2026")</code>	vérifier avec <code>IsDate</code>
Calcul sur une cellule texte	<code>Cells(i, 2).Value * 1.2</code> avec "N.C." en B	tester avant de calculer

03 Sécuriser une boucle de calcul

IGNORER LES CELLULES NON NUMÉRIQUES

```
Sub Additionner()
    Dim c As Range, total As Double
    For Each c In Range("C2:C500")
        If Not IsError(c.Value) Then
            If IsNumeric(c.Value) And c.Value <> "" Then
                total = total + c.Value
            End If
        End If
    Next c
    MsgBox "Total : " & total
End Sub
```

Le test `IsError` passe en premier, dans un `If` séparé : comparer une valeur d'erreur avec `<>` déclencherait lui-même l'erreur 13.

Fonction	Renvoie True si
<code>IsNumeric(x)</code>	x peut être converti en nombre
<code>IsDate(x)</code>	x peut être converti en date
<code>IsError(x)</code>	x est une valeur d'erreur Excel (#N/A, #DIV/0!...)
<code>IsEmpty(x)</code>	la variable ou la cellule est vide

04 Pièges à éviter

⚠ PIÈGE

Les nombres stockés en texte

Une cellule peut afficher 120 mais contenir le texte "120" (triangle vert dans le coin). `IsNumeric` renvoie True, et `Cdbl(c.Value)` convertit proprement.

⚠ PIÈGE

Le séparateur décimal

Un texte comme "12.5" peut ne pas être reconnu comme un nombre sur un Excel français, qui attend une virgule. `Val("12.5")` lit toujours le point comme séparateur décimal.

FICHE DE COURS GRATUITE

VBA EXCEL | ERREURS

Erreur 9

L'erreur 9 signifie que VBA cherche un élément qui n'existe pas dans une collection : une feuille au nom légèrement différent, un classeur qui n'est pas ouvert, ou une case d'un tableau au-delà de sa taille.

Version	V1.2
Dernière mise à jour	29/10/2026
Niveau	2 / 3, Intermédiaire
Durée de lecture	5 min
Auteur	Clément Delobel



Version en ligne
mesétudesdecas.fr/vba/
erreur-9.html

01 Le message

EN UNE PHRASE

Erreur d'exécution 9 : L'indice n'appartient pas à la sélection. Le nom ou le numéro demandé ne correspond à aucun élément de la collection (feuilles, classeurs, tableau).

02 Les causes les plus fréquentes

Cause	Exemple	Correction
Nom de feuille différent	<code>Worksheets("Janvier ")</code> avec une espace en trop	copier le nom exact de l'onglet, ou utiliser le nom de code
Feuille renommée par l'utilisateur	l'onglet Ventes est devenu Ventes 2026	utiliser le nom de code (Feuil1)
Classeur non ouvert	<code>Workbooks("Budget.xlsx")</code> alors que le fichier est fermé	l'ouvrir d'abord avec <code>Workbooks.Open</code>
Indice de tableau hors limites	<code>Dim t(1 To 5) puis t(6) = 1</code>	vérifier avec <code>UBound(t)</code>
Tableau dynamique non dimensionné	<code>Dim t() As Long puis t(1) = 5</code>	<code>ReDim t(1 To n)</code> avant l'usage

03 Vérifier avant d'utiliser

TESTER L'EXISTENCE D'UNE FEUILLE

```
Sub OuvrirFeuille()  
    Dim ws As Worksheet  
    On Error Resume Next  
    Set ws = ThisWorkbook.Worksheets("Synthèse")  
    On Error GoTo 0  
  
    If ws Is Nothing Then  
        MsgBox "La feuille Synthèse est introuvable.", vbExclamation  
        Exit Sub  
    End If  
    ws.Activate  
End Sub
```

ASTUCE

Les extensions de fichier

Selon les réglages de Windows, `Workbooks("Budget")` peut échouer alors que `Workbooks("Budget.xlsx")` fonctionne. Indiquez toujours l'extension.

04 Pièges à éviter

PIÈGE

Un tableau qui commence à 0

`Dim t(5)` crée six cases, numérotées de 0 à 5. Pour éviter toute ambiguïté, écrivez les bornes : `Dim t(1 To 5)`.

PIÈGE

Les noms qui se ressemblent

« Synthèse » et « Synthèse » sont deux noms différents. En cas de doute, listez les feuilles avec une boucle **For Each** et `Debug.Print ws.Name`.