AUTOMATISATION • VISUAL BASIC EDITOR • INGÉNIERIE DES PROCÉDURES

MACROS-COMMANDES

Enregistreur, Programmation VBA, Débogage & Personnalisation

Apprendre à concevoir, enregistrer, modifier et déboguer des macros interactives pour automatiser les tâches répétitives, manipuler les objets Excel (Workbooks, Sheets, Ranges) et créer des outils métier sur mesure.

Enregistreur & Déclencheurs
Boutons, Ruban & PERSONAL.XLSB

Code VBA & Objets
Sub, Variables, Boucles & POO

Débogage Professionnel
Pas à pas (F8), Points d'arrêt & VBE

Automatiser Excel : Du clic mécanique au code intelligent

Passer d'une utilisation passive d'Excel à la création de processus métier autonomes et fiables.

Les **macro-commandes** constituent le levier d'efficacité ultime dans Microsoft Excel. Elles permettent de convertir des séquences de manipulation fastidieuses (mises en forme, consolidations, imports de fichiers, calculs) en **traitements instantanés déclenchés en 1 seul clic**.

Ce guide vous accompagne depuis les fondamentaux de l'enregistreur jusqu'à la modification experte du code dans l'éditeur Visual Basic (VBE).

Objectifs Pédagogiques

- Comprendre la terminologie objet et l'architecture des macros.
- Enregistrer des actions en références absolues et relatives.
- Créer des boutons, onglets et menus personnalisés.
- Déboguer pas à pas et inspecter les variables locales.
- Structurer du code VBA : variables, boîtes de dialogue, boucles.
- Manipuler classeurs, feuilles et flux de données externes.

Pré-requis Recommandés

Utilisateurs confirmés d'Excel ayant suivi le cours *Excel Perfectionnement* ou maîtrisant :

- La gestion avancée des classeurs et des feuilles de calcul.
- Les formules de calcul conditionnelles et de recherche.
- Les Tableaux Structurés et la mise en forme de données.

Règle d'or de l'automatisation : Ne jamais coder ce qui peut être résolu nativement par une formule ou Power Query, mais utiliser VBA dès qu'il s'agit d'orchestrer des actions multi-fichiers, d'interagir avec l'utilisateur ou de piloter l'interface Excel.

Rappel sur la terminologie des objets et fonctionnalités

Comprendre comment Excel modélise son environnement sous forme d'une hiérarchie d'objets connectés.

Excel ne voit pas votre fichier comme une grille visuelle, mais comme une **arborescence hiérarchique d'objets**. Pour donner une instruction à Excel (écrire du texte, colorer une case, créer un onglet), vous devez désigner l'objet précis en suivant cette chaîne hiérarchique.

1. Objet vs Collection

Objet : Un élément individuel d'Excel (ex: `Workbook`, `Worksheet`, `Range`, `Chart`).

Collection : Un ensemble regroupant tous les objets du même type. Les collections portent toujours un **S** à la fin : `Workbooks`, `Worksheets`.

2. Propriété vs Méthode

Propriété : Une caractéristique ou un état de l'objet (lecture ou écriture) : `Range("A1").Value`, `Range("A1").Interior.Color`.

Méthode : Une action que l'objet sait exécuter (un verbe) : `Range("A1:B5").ClearContents`, `Workbooks.Add`.

OBJET EXCEL (VBA)	ÉQUIVALENT MÉTIER	PROPRIÉTÉS COMMUNES	MÉTHODES COMMUNES
`Application`	L'application Excel entière	`.ScreenUpdating`, `.DisplayAlerts`	`.Quit`, `.Calculate`
`Workbook`	Un fichier classeur (ex: `Ventes.xlsx`)	`.Name`, `.Path`, `.Saved`	`.Save`, `.SaveAs`, `.Close`
`Worksheet`	Un onglet de calcul	`.Name`, `.Visible`, `.Index`	`.Activate`, `.Copy`, `.Delete`
`Range` / `Cells`	Une cellule ou une plage rectangulaire	`.Value`, `.Formula`, `.Font`, `.Row`	`.Select`, `.Copy`, `.Clear`, `.AutoFit`

SYNTAXE DE RÉFÉRENCEMENT HIÉRARCHIQUE UNIVERSELLE — Modèle P.O.O

```
' Accès complet du général vers le particulier : Application > Classeur > Feuille > Plage
Application.Workbooks("Budget_2026.xlsm").Worksheets("Synthèse").Range("B4").Value = 150000

' Raccourci implicite lorsque le classeur et la feuille sont déjà actifs :
Range("B4").Value = 150000
```

⚠ **Attention au référencement implicite :** Si vous écrivez simplement `Range("A1").Value = 10`, Excel appliquera l'action sur la feuille *actuellement visible à l'écran*. Pour des macros robustes et multi-feuilles, explicitez toujours la feuille cible : `Worksheets("Données").Range("A1").Value = 10`.

Qu'est-ce qu'une macro et quand l'utiliser ?

Identifier les opportunités d'automatisation à fort retour sur investissement (ROI).

Une **macro-commande** est une suite d'instructions écrites en langage **VBA (Visual Basic for Applications)** qui ordonne à Excel d'exécuter automatiquement une séquence de tâches. Elle peut être créée soit via l'**enregistreur de macros** (qui traduit vos clics en code), soit en **écrivant directement le code** dans l'éditeur.

✓ Quand créer une Macro ?

- **Tâches répétitives fréquentes :** Traitements quotidiens ou hebdomadaires identiques (ex: formatage d'un export ERP).
- **Processus multi-étapes complexes :** Enchaînement de 15 opérations successives sans risque d'oubli humain.
- **Standardisation d'équipe :** Offrir un bouton "Valider le rapport" qui applique les règles de l'entreprise.
- **Intégration de flux :** Importer des fichiers CSV externes et les combiner en 1 clic.

✗ Quand Éviter les Macros ?

- **Calculs réalisables par formule :** Préférer les formules dynamiques (`FILTRE`, `RECHERCHEX`).
- **Nettoyage ETL lourd :** Préférer **Power Query** si la tâche consiste uniquement à transformer des colonnes.
- **Tâche ponctuelle unique :** Si une opération n'est exécutée qu'une seule fois dans l'année, l'enregistrer prend plus de temps que la faire à la main.

Formats de Fichiers Excel : Attention à l'Extension !

.XLSX (Standard)
Interdit les macros. Si vous enregistrez un fichier contenant du code en .xlsx, toutes vos macros sont **définitivement supprimées** !

.XLSM (Recommandé)
Classeur prenant en charge les macros. Format XML standard avec conteneur de projets VBA sécurisé.

.XLSB (Binaire)
Classeur Binaire ultra-rapide. Idéal pour les très gros volumes de données et pour le classeur de macros personnelles.

✓ **Bénéfice direct :** Une macro bien conçue élimine 100% des erreurs humaines de saisie et réduit un temps de traitement de 45 minutes manuelles à **moins de 2 secondes** d'exécution machine.

Méthodologie de création et limites de l'enregistreur

Adopter une démarche rigoureuse et comprendre la frontière entre enregistrement et code écrit.

L'enregistreur de macros agit comme un "magnétophone" : il transcrit chaque clic, défilement et sélection en instructions VBA. Sans préparation préalable, vous enregistrerez des dizaines d'actions parasites qui ralentiront l'exécution et provoqueront des bugs.

La Méthode en 5 Étapes pour Réussir son Enregistrement

1. **Définir l'état initial précis (Point de départ)**
   Où doit se situer le curseur avant de cliquer sur "Enregistrer" ? Quelle feuille doit être active ?
2. **Répéter le scénario "à blanc" sans enregistrer**
   Effectuez les clics manuellement une première fois pour mémoriser la trajectoire sans hésitation.
3. **Choisir le mode de référence (Absolu ou Relatif)**
   L'action doit-elle toujours s'appliquer sur la cellule A1 ou sur la cellule actuellement sélectionnée ?
4. **Enregistrer avec fluidité sans clics superflus**
   Évitez les défilements inutiles à la molette et les sélections de cellules sans but.
5. **Arrêter l'enregistrement immédiatement et tester**
   Cliquez sur le bouton "Arrêter" dès la dernière manipulation terminée. Testez sur une copie de données.

CE QUE L'ENREGISTREUR FAIT PARFAITEMENT	CE QUE L'ENREGISTREUR NE PEUT PAS FAIRE (NÉCESSITE VBA)
Générer la syntaxe exacte d'un format de cellule complexe.	**Créer des boucles** pour traiter 50 fichiers ou 10 000 lignes automatiquement.
Appliquer un filtre ou trier une plage connue.	**Poser des conditions logiques :** "Si le montant > 1000 alors faire X sinon faire Y".
Créer un graphique standard ou un TCD.	**Interagir avec l'utilisateur** (Boîte de saisie `InputBox` ou confirmation `MsgBox`).
Insérer des formules et figer les volets.	**Détecter dynamiquement la dernière ligne** si le nombre de lignes change demain.

Le compromis parfait : Utilisez l'enregistreur pour générer 80% du code fastidieux (syntaxes de couleurs, bordures, paramètres), puis ouvrez l'éditeur VBE pour ajouter l'intelligence manquante (boucles, variables et conditions).

Accéder à l'onglet Développeur & Préparer Excel

Activer le panneau de contrôle de l'automatisation et configurer la sécurité des macros.

Par défaut, Microsoft masque l'onglet **Développeur** dans le ruban d'Excel pour ne pas surcharger les utilisateurs débutants. Cet onglet regroupe tous les outils indispensables pour enregistrer, modifier et sécuriser vos scripts.

Procédure d'Activation de l'Onglet Développeur

1. **Ouvrir les Options Excel**
   Cliquez sur le menu **Fichier** (en haut à gauche), puis tout en bas sur **Options**.
2. **Sélectionner "Personnaliser le ruban"**
   Dans la colonne de gauche de la boîte de dialogue, cliquez sur **Personnaliser le ruban**.
3. **Cocher la case "Développeur"**
   Dans la liste des onglets principaux à droite, cochez la case devant **Développeur** et validez par **OK**.

Cartographie des Outils du Groupe "Code" de l'Onglet Développeur

`Alt + F11` Visual Basic

Ouvre l'éditeur VBE pour consulter, modifier ou rédiger directement le code source.

`Alt + F8` Macros

Affiche la liste de toutes les macros disponibles pour les exécuter ou les modifier.

Enregistrer une macro

Lance la capture de vos actions utilisateur pour générer automatiquement la procédure.

Utiliser les références relatives

Bouton bascule pour enregistrer les déplacements par rapport à la cellule active.

Sécurité des macros (Centre de gestion de la confidentialité) : Par mesure de sécurité, vérifiez dans *Développeur* > *Sécurité des macros* que l'option est configurée sur **"Désactiver toutes les macros avec notification"**. Cela vous permet d'autoriser l'exécution de vos macros via un bandeau jaune à l'ouverture de vos classeurs.

Utiliser l'enregistreur et définir des raccourcis clavier

Paramétrer la boîte d'enregistrement, respecter les règles de nommage et assigner des raccourcis.

Lorsque vous cliquez sur **Enregistrer une macro**, Excel affiche une boîte de dialogue fondamentale. Chaque champ doit être configuré avec soin pour garantir la clarté et l'accessibilité de votre macro.

Les 4 Paramètres Clés de la Boîte d'Enregistrement

CHAMP	RÈGLES & BONNES PRATIQUES	EXEMPLE RECOMMANDÉ
Nom de la macro	Doit commencer par une lettre. **Aucun espace**, ni tiret -, ni caractère spécial. Utiliser le CamelCase ou le trait de soulignement _.	`FormaterTableauMensuel` ou `Import_Donnees_ERP`
Touche de raccourci	Permet d'exécuter la macro au clavier. **Privilégier la combinaison avec Maj** (ex: `Ctrl + Maj + R`) pour ne pas écraser les raccourcis système.	`Ctrl + Maj + F`
Enregistrer dans	Définit la portée de visibilité de la macro (Ce classeur, Nouveau classeur ou Classeur de macros personnelles).	*Ce classeur*
Description	Documentation interne expliquant l'objectif de la macro, la date et l'auteur. Très utile pour vos collaborateurs.	*"Nettoie les lignes vides et applique le thème corporate."*

Danger : Conflits de Raccourcis

Si vous affectez le raccourci `Ctrl + C` ou `Ctrl + S` à votre macro, le raccourci natif d'Excel (Copier / Enregistrer) sera **désactivé** dans cette session !

Toujours appuyer sur MAJ lors de la frappe de la lettre.

Le Bouton d'État dans la Barre d'État

Pendant l'enregistrement, un **carré bleu** s'affiche en bas à gauche dans la barre d'état d'Excel.

Un simple clic sur ce carré **arrête immédiatement l'enregistrement** sans devoir retourner sur l'onglet Développeur.

Modifier le raccourci après coup : Si vous avez oublié d'affecter un raccourci ou souhaitez le changer : ouvrez la boîte des macros (`Alt + F8`), sélectionnez la macro et cliquez sur le bouton **Options...**

Choisir l'emplacement et définir PERSONAL.XLSB

Comprendre où sont stockées vos macros pour les rendre universelles ou spécifiques à un classeur.

L'un des choix les plus stratégiques lors de la création d'une macro est son **emplacement d'enregistrement**. Ce choix détermine si votre macro voyagera avec le fichier (pour vos collègues) ou restera installée sur votre ordinateur (pour tous vos fichiers).

EMPLACEMENT	FONCTIONNEMENT TECHNIQUE	CAS D'USAGE RECOMMANDÉ
Ce classeur (ThisWorkbook)	Le code VBA est enregistré directement à l'intérieur du fichier actuel (qui doit être enregistré au format `.xlsm`).	**Macros d'équipe ou métier :** Le fichier est envoyé par email et les collègues peuvent exécuter la macro sur leur PC.
Classeur de macros personnelles (`PERSONAL.XLSB`)	Le code est enregistré dans un fichier spécial masqué stocké sur votre profil Windows (`XLSTART`). Il s'ouvre automatiquement en arrière-plan à chaque lancement d'Excel.	**Boîte à outils personnelle :** Macros utilitaires utilisables sur *n'importe quel classeur* ouvert sur votre machine (ex: nettoyage d'exports).
Nouveau classeur	Excel crée instantanément un nouveau fichier vierge et y injecte la macro enregistrée.	Création de templates ou de bibliothèques de fonctions isolées.

Comment Créer & Gérer son Classeur PERSONAL.XLSB

Le fichier `PERSONAL.XLSB` n'existe pas par défaut. Il est créé automatiquement la **première fois** que vous enregistrez une macro en choisissant l'option "*Classeur de macros personnelles*".

Emplacement Physique sur Windows

`%AppData%\Microsoft\Excel\XLSTART\PERSONAL.XLSB`

Afficher / Masquer le Fichier

Onglet *Affichage* > *Afficher la fenêtre* pour voir le classeur ou *Masquer* pour le garder invisible.

⚠ **Enregistrement lors de la fermeture :** Lorsque vous quittez Excel après avoir créé ou modifié une macro personnelle, Excel affiche le message : "*Voulez-vous enregistrer les modifications apportées au classeur de macros personnelles ?*". Cliquez toujours sur **Enregistrer** pour ne pas perdre vos outils !

Enregistrer des macros en références relatives

Comprendre l'impact fondamental du mode relatif sur la réutilisabilité de vos automatisations.

Par défaut, l'enregistreur fonctionne en **Mode Absolu** : si vous cliquez sur B4, il enregistre `Range("B4").Select`. Si vous activez le bouton **"Utiliser les références relatives"**, Excel enregistre les actions sous forme de **décalages (Offsets)** par rapport à la cellule active.

Mode Absolu (Bouton Éteint)

L'action s'applique **toujours sur les mêmes cellules fixes**, peu importe où se trouve votre curseur au moment de lancer la macro.

CODE VBA GÉNÉRÉ EN ABSOLU

```
Range("A1").Select
ActiveCell.FormulaR1C1 = "Total"
Range("B1").Select
ActiveCell.FormulaR1C1 = "=SUM(R2C:R10C)"
```

Usage : Créer un modèle de tableau toujours situé en A1:F20.

Mode Relatif (Bouton Allumé)

L'action s'applique **par rapport à la cellule sélectionnée** au moment du lancement. Si le curseur est en D10, l'action démarrera en D10.

CODE VBA GÉNÉRÉ EN RELATIF

```
ActiveCell.Offset(0, 0).Range("A1").Select
ActiveCell.FormulaR1C1 = "Total"
ActiveCell.Offset(0, 1).Range("A1").Select
ActiveCell.FormulaR1C1 = "=SUM(R[-5]C:R[-1]C)"
```

Usage : Formater une ligne sur laquelle le curseur est positionné.

Cas Pratique : Insérer une Ligne de Total sous n'importe quel tableau

1. Positionnez votre curseur sur la dernière cellule renseignée de votre colonne de montants.
2. Activez le bouton **"Utiliser les références relatives"** dans le ruban Développeur.
3. Cliquez sur **Enregistrer une macro** (Nom : `Total_Ligne_Suivante`).
4. Appuyez sur la flèche **Bas** (pour descendre d'une case), tapez `=SOMME(` puis sélectionnez les 10 cases au-dessus, validez par Entrée et mettez en Gras.
5. Arrêtez l'enregistrement : votre macro fonctionne désormais sur **n'importe quelle colonne de n'importe quel tableau** !

Combinaison puissante : Vous pouvez **basculer** entre mode absolu et mode relatif *pendant* un même enregistrement ! Désactivez le bouton pour aller en cellule A1, puis réactivez-le pour vous déplacer de 2 lignes vers le bas.

Appeler la macro par la liste des macros

Maîtriser la boîte de dialogue standard (Alt + F8) et administrer les macros existantes.

La boîte de dialogue **Macros** (`Alt + F8`) est le centre de contrôle historique d'Excel pour retrouver, tester, modifier et supprimer toutes les procédures disponibles dans la session actuelle.

Anatomie de la Boîte de Dialogue "Macros" (Alt + F8)

BOUTON / ÉLÉMENT	FONCTION & COMPORTEMENT	RACCOURCI / EFFET
Exécuter	Lance immédiatement la macro sélectionnée à pleine vitesse.	Double-clic sur le nom
Pas à pas détaillé	Ouvre l'éditeur VBE et positionne la flèche d'exécution sur la première ligne (mode débogage).	Touche `F8`
Modifier	Ouvre l'éditeur VBE directement sur le code source de la procédure pour l'éditer.	Ouvre VBE
Créer	Disponible si vous saisissez un nouveau nom : crée automatiquement une procédure vide `Sub Nom() ... End Sub`.	Création rapide
Options...	Permet d'ajouter ou modifier la touche de raccourci clavier et le texte de description.	Paramétrage
Supprimer	Supprime définitivement le code de la macro sélectionnée du module.	Action irréversible

Filtrer par Classeur d'Origine

Le menu déroulant "*Macros dans :*" en bas de la boîte permet de filtrer la liste :

- **Tous les classeurs ouverts :** Affiche toutes les macros actives (y compris `PERSONAL.XLSB`).
- **Ce classeur :** Ne liste que les macros enregistrées dans le fichier actif.

Macros Masquées dans la Liste

Une macro n'apparaît **pas** dans la liste `Alt + F8` si :

- Elle est déclarée comme `Private Sub`.
- Elle attend des arguments obligatoires (ex: `Sub Traiter(ligne As Long)`).
- Elle se trouve dans un module d'objet avec `Option Private Module`.

Astuce Pro : Si votre macro est dans `PERSONAL.XLSB`, elle apparaît sous la forme `PERSONAL.XLSB!NomDeLaMacro`. Vous pouvez l'exécuter sur n'importe quel classeur sans restriction.

Créer un onglet, un groupe et des commandes sur mesure

Transformer vos scripts VBA en véritables boutons intégrés au ruban standard de Microsoft Excel.

Pour déployer vos macros auprès de collègues non-initiés au code, la meilleure approche consiste à créer un **onglet dédié dans le ruban** (ex: "*Outils Finance*" ou "*Mes Outils*") avec des icônes explicites.

Procédure Pas à Pas : Concevoir son Ruban Personnalisé

1 Accéder à la personnalisation du ruban

Faites un clic droit sur n'importe quel onglet du ruban et sélectionnez **Personnaliser le ruban...**

2 Créer un "Nouvel onglet" et un "Nouveau groupe"

En bas de la colonne de droite, cliquez sur **Nouvel onglet**. Renommez l'onglet (ex: " *Automatisation*") et son sous-groupe (ex: "*Imports & Nettoyage*").

3 Sélectionner la catégorie "Macros"

Dans la liste déroulante de gauche "*Choisir les commandes dans les catégories suivantes :*", sélectionnez **Macros**.

4 Ajouter la macro et personnaliser son icône

Sélectionnez votre macro, cliquez sur **Ajouter >>** pour l'insérer dans votre groupe, puis cliquez sur **Renommer...** pour lui donner un libellé propre et lui choisir une icône (calculatrice, dossier, éclair, etc.).

Avantages Ergonomiques

- Aucun besoin de retenir des raccourcis clavier obscurs.
- Boutons toujours visibles au premier plan.
- Info-bulles explicatives au survol de la souris.

Exporter & Partager sa Configuration

En bas à droite de la boîte de dialogue : cliquez sur **Importer/Exporter > Exporter toutes les personnalisations**.

Génère un fichier `.exportedUI` que vous pouvez importer en 1 clic sur les postes de vos collègues.

Recommandation UI : Structurez vos groupes logiquement de gauche à droite : *1. Imports > 2. Traitements & Calculs > 3. Exports PDF & Clôture* pour guider naturellement l'utilisateur dans son flux de travail.

Insertion d'un menu personnalisé et Barre Accès Rapide

Déclencher des macros en 1 clic permanent via la barre d'outils Accès Rapide (QAT).

La **Barre d'outils Accès Rapide (QAT)** reste toujours visible au sommet d'Excel, quel que soit l'onglet du ruban actif. C'est l'emplacement idéal pour héberger vos 3 à 5 macros personnelles les plus fréquemment sollicitées.

Ajouter une Macro à la Barre d'Outils Accès Rapide

1 Ouvrir les options de la barre d'accès rapide

Cliquez sur la petite flèche descendante située tout à droite de la barre d'accès rapide > **Autres commandes...**

2 Sélectionner "Macros" et ajouter la commande

Dans "*Choisir les commandes dans :*", choisissez **Macros**, sélectionnez votre macro et cliquez sur **Ajouter >>**.

3 Personnaliser l'icône et le nom d'affichage

Cliquez sur **Modifier...** en bas à droite pour attribuer un symbole visuel distinctif et validez par **OK**.

Le Super-Pouvoir des Raccourcis Alt + Chiffre

Chaque bouton de la barre d'accès rapide se voit automatiquement assigner un raccourci clavier séquentiel :

`Alt + 1` Déclenche le 1er bouton

`Alt + 2` Déclenche le 2e bouton

`Alt + 3` Déclenche le 3e bouton

Exécutez vos macros d'une seule main en 1 milliseconde !

Portée Locale vs Globale

Dans la boîte des options, vous pouvez choisir d'affecter le bouton :

- **Pour tous les documents (par défaut) :** Idéal pour les macros de `PERSONAL.XLSB`.
- **Pour [Ce classeur uniquement] :** Le bouton ne s'affiche que lorsque ce fichier précis est actif.

Astuce Emplacement : Faites un clic droit sur la barre d'accès rapide et choisissez *"Afficher la barre d'outils Accès rapide en dessous du ruban"* : vos boutons seront plus proches de vos cellules et disposeront d'un espace d'affichage illimité.

Affectation d'une macro à un bouton ou une image

Concevoir des interfaces utilisateurs intuitives directement insérées dans vos feuilles de calcul.

Intégrer des boutons cliquables directement sur la feuille de calcul offre une expérience utilisateur ultra-moderne et élimine tout besoin de manipuler le ruban pour vos utilisateurs finaux.

1. Formes Géométriques (Recommandé)

Insertion > Formes > Rectangle à coins arrondis.

Design moderne, dégradés, ombres portées, texte centré. Clic droit > **Affecter une macro...**

2. Contrôle de Formulaire

Développeur > Insérer > Bouton (Contrôle de formulaire).

Bouton Windows classique gris. La boîte d'affectation de macro s'ouvre automatiquement au tracé.

3. Images & Icônes SVG

Insertion > Images / Icônes.

Logo de votre entreprise, icône de synchronisation ou flèche d'export. Clic droit > **Affecter une macro...**

Procédure d'Affectation & Fixation dans la Grille

1. Dessinez votre forme ou insérez votre image sur la feuille.
2. Faites un **clic droit** sur l'objet et sélectionnez **Affecter une macro...**
3. Sélectionnez la procédure dans la liste et validez par **OK**.
4. **Propriété essentielle :** Clic droit sur la forme > *Taille et propriétés* > *Propriétés* > Cocher **"Ne pas déplacer ou dimensionner avec les cellules"** pour éviter que le bouton ne se déforme lors du masquage de lignes.

Modifier le texte d'un bouton sans lancer la macro : Une fois la macro affectée, un clic gauche déclenche immédiatement le code ! Pour modifier le texte ou déplacer la forme, faites un **clic droit** (ou maintenez la touche `Ctrl` lors du clic gauche).

Visual Basic Editor (VBE) : Projet, Propriétés et Modules

Prendre ses repères dans l'atelier de programmation intégré à Microsoft Excel (Alt + F11).

VBE (Visual Basic Editor) est l'environnement de développement complet où s'écrit et s'exécute le code VBA. Accessible à tout moment via le raccourci magique `Alt + F11`.

1. Explorateur de Projets

Raccourci : `Ctrl + R`

Arborescence de tous les classeurs ouverts, de leurs feuilles de calcul (`Feuil1`, `Feuil2`), de l'objet `ThisWorkbook` et des dossiers de **Modules**.

2. Fenêtre des Propriétés

Raccourci : `F4`

Affiche les attributs de l'objet sélectionné (ex: propriété `(Name)`, `Visible` pour masquer très profondément une feuille).

3. Fenêtre de Code

Raccourci : `F7`

L'éditeur de texte principal où s'affichent vos procédures avec coloration syntaxique automatique (Mots-clés en bleu, commentaires en vert).

Où Insérer son Code ? Les 3 Types de Conteneurs

CONTENEUR	RÔLE & NATURE	COMMENT LE CRÉER ?
Module Standard (`Module1`)	**L'emplacement de 95% de vos macros.** Héberge les macros enregistrées et les fonctions personnalisées.	Menu *Insertion* > *Module*
Objet Feuille (`Feuil1`)	Contient le code événementiel lié aux actions sur cette feuille (ex: clic, changement de cellule `Worksheet_Change`).	Double-clic sur la feuille dans l'explorateur
Objet ThisWorkbook	Contient le code lié au classeur global (ex: exécution automatique à l'ouverture `Workbook_Open`).	Double-clic sur `ThisWorkbook`

Fenêtre disparue ? Si l'explorateur de projets ou la fenêtre de propriétés a été fermée par mégarde, retrouvez-les instantanément dans le menu **Affichage** de VBE ou via `Ctrl + R` et `F4`.

La barre de débogage et l'exécution en mode « pas à pas »

Ralentir le temps et exécuter votre code ligne par ligne pour comprendre et corriger les anomalies.

Le mode **Pas à pas** est l'arme absolue du développeur VBA. Plutôt que de laisser Excel exécuter 50 lignes en 1/100e de seconde, vous avancez **instruction par instruction** avec la touche `F8` en observant l'effet direct sur votre feuille Excel en temps réel.

→ Les Commandes Clés de l'Exécution Pas à Pas

RACCOURCI CLAVIER	ACTION DÉBOGAGE	COMPORTEMENT PRÉCIS
`F8`	**Pas à pas détaillé** (Step Into)	Exécute la ligne surlignée en jaune et s'arrête sur la ligne suivante. Si la ligne appelle une autre macro, il entre à l'intérieur.
`Maj + F8`	**Pas à pas principal** (Step Over)	Exécute une sous-procédure entière d'un coup sans entrer dans son détail ligne par ligne.
`Ctrl + Maj + F8`	**Pas à pas sortant** (Step Out)	Termine immédiatement la sous-procédure en cours et revient à la macro parente.
`F5`	**Continuer / Exécuter**	Reprend l'exécution normale à pleine vitesse jusqu'à la fin ou jusqu'au prochain point d'arrêt.
`Ctrl + Pause` / `Échap`	**Interrompre l'exécution**	Stoppe une boucle infinie ou une macro bloquée en cours de calcul.

Le Surlignage Jaune d'Exécution

La ligne surlignée en **jaune** avec une flèche jaune dans la marge gauche représente l'instruction qui va être exécutée *au prochain appui sur F8* (elle n'est pas encore exécutée).

Déplacer la Flèche Jaune Manuellement

Vous pouvez **cliquer et glisser** la flèche jaune avec la souris vers le haut pour **rejouer** une ligne modifiée, ou vers le bas pour **sauter** une ligne problématique sans l'exécuter !

Organisation des fenêtres : Placez votre fenêtre Excel sur la moitié gauche de votre écran (`Win + Gauche`) et la fenêtre VBE sur la moitié droite (`Win + Droite`). Vous verrez vos cellules changer de couleur à chaque appui sur `F8` .

Utiliser les points d'arrêt et la fenêtre des espions

Interrompre l'exécution à des emplacements stratégiques et surveiller l'évolution des variables complexes.

Lorsque votre macro contient 200 lignes, appuyer 200 fois sur F8 devient fastidieux. Les **Points d'arrêt** permettent d'exécuter la macro à pleine vitesse et de la **mettre automatiquement en pause** exactement à l'endroit suspect.

Points d'Arrêt (Touche F9)

Cliquez dans la **marge grise** à gauche d'une ligne de code (ou placez le curseur et appuyez sur `F9`).

- Un **point bordeaux** apparaît et la ligne devient rouge foncé.
- Lancez la macro avec `F5` : Excel s'exécute instantanément et stoppe net sur cette ligne.
- Vous pouvez alors continuer en pas à pas avec `F8`.
- Pour tout retirer : `Ctrl + Maj + F9`.

L'Instruction "Stop" dans le Code

Vous pouvez écrire le mot-clé `Stop` directement dans votre code VBA :

```
If TotalVentes < 0 Then
    Stop  ' Suspend la macro si anomalie
End If
```

Contrairement aux points d'arrêt F9 qui disparaissent à la fermeture d'Excel, l'instruction `Stop` est enregistrée dans le classeur.

La Fenêtre des Espions (Watch Window)

Accessible via le menu *Débogage* > *Ajouter un espion...* ou *Affichage* > *Fenêtre Espion*.

TYPE D'ESPION	COMPORTEMENT	USAGE IDÉAL
Expression d'espion	Affiche en continu la valeur d'une variable ou formule dans un tableau.	Surveiller le contenu de `MaVariable`.
Arrêt si la valeur est Vraie	Interrompt automatiquement la macro dès qu'une condition logique devient vraie.	Stopper dès que `Compteur > 500`.
Arrêt si la valeur change	Interrompt la macro dès que la variable est modifiée par n'importe quelle ligne de code.	Traquer quelle ligne modifie un paramètre.

Survol à la souris (DataTips) : Pendant une pause en mode pas à pas, survolez simplement n'importe quelle variable dans votre code avec le curseur de votre souris : une petite bulle s'affiche instantanément avec sa valeur actuelle !

Fenêtre des variables locales et Exécution immédiate

Inspecter l'état complet de la mémoire et tester des instructions à la volée sans relancer le classeur.

Les deux fenêtres complémentaires indispensables dans VBE sont la **Fenêtre Variables locales** (qui liste automatiquement toutes les variables actives et leurs sous-propriétés) et la **Fenêtre Exécution immédiate** (`Ctrl + G`), véritable console interactive.

1. Fenêtre Variables Locales

Menu *Affichage* > *Fenêtre Variables locales*.

- Détecte **automatiquement** toutes les variables de la procédure sans devoir créer d'espions manuels.
- Affiche 3 colonnes : **Expression**, **Valeur** et **Type**.
- Permet de déplier les objets (ex: déplier `ActiveSheet` pour voir ses 50 propriétés).

2. Fenêtre Exécution Immédiate (`Ctrl + G`)

Console de commande directe pour interroger Excel en direct :

- **Interroger une valeur :** Tapez `?` suivi de l'expression et validez par Entrée.
- **Exécuter une action :** Tapez l'instruction et appuyez sur Entrée (effet immédiat dans Excel).

EXEMPLES DE COMMANDES DANS LA FENÊTRE EXÉCUTION IMMÉDIATE (CTRL + G) — Console VBE

```
' 1. Connaître la valeur de la cellule A1 de la feuille active :
?Range("A1").Value
 → Affiche : 45 200

' 2. Connaître le nombre de feuilles du classeur actif :
?Worksheets.Count
 → Affiche : 8

' 3. Forcer la réactivation des alertes et du rafraîchissement d'écran bloqués :
Application.ScreenUpdating = True
Application.DisplayAlerts = True
```

Utiliser Debug.Print dans son Code

L'instruction `Debug.Print` permet d'écrire des logs dans la console d'exécution sans perturber l'écran de l'utilisateur avec des boîtes de dialogue :

```
For i = 2 To 100
    Debug.Print "Traitement de la ligne : " & i & " | Client : " & Cells(i, 1).Value
Next i
```

Sauvetage d'urgence : Si votre écran semble "gelé" après l'arrêt imprévu d'une macro en plein débogage, ouvrez la fenêtre Exécution (`Ctrl + G`), tapez `Application.ScreenUpdating = True` et appuyez sur Entrée pour retrouver l'affichage normal.

Visual Basic pour Application : Modèle Objet (P.O.O)

Maîtriser la grammaire universelle de VBA : Objets, Propriétés, Méthodes et Événements.

Le langage VBA repose sur le paradigme de la **Programmation Orientée Objets (P.O.O)**. Tout comme une voiture possède des caractéristiques (couleur, vitesse) et des actions possibles (démarrer, freiner), un objet Excel possède des **Propriétés** et des **Méthodes**.

1. Propriétés (Ce que l'objet EST ou POSSÈDE)

Une propriété contient une valeur (texte, nombre, couleur, état). On utilise le signe = pour lui assigner une nouvelle valeur :

```
' Modifier le contenu textuel
Range("A1").Value = "Chiffre d'Affaires"

' Modifier la couleur d'arrière-plan
Range("A1").Interior.Color = RGB(79, 70, 229)

' Mettre en gras
Range("A1").Font.Bold = True
```

2. Méthodes (Ce que l'objet FAIT)

Une méthode exécute une action concrète. Elle peut recevoir des paramètres optionnels :

```
' Effacer le contenu sans effacer les bordures
Range("A1:F20").ClearContents

' Copier une plage vers une autre destination
Range("A1:D10").Copy Destination:=Range("H1")

' Ajuster automatiquement la largeur des colonnes
Columns("A:F").AutoFit
```

L'Autocomplétion IntelliSense : Le Point Séparateur

Dans l'éditeur VBE, dès que vous tapez le nom d'un objet suivi d'un **point .**, un menu déroulant surgit avec toutes les propriétés (icône main pointant un doigt) et méthodes (icône cube volant vert) valides !

Raccourci pour forcer l'affichage de la liste : `Ctrl + Espace`.

La propriété par défaut : Pour un objet Range, la propriété par défaut est `.Value`. Ainsi, écrire `Range("A1") = 100` équivaut à `Range("A1").Value = 100`. Il est toutefois fortement recommandé de toujours expliciter `.Value` pour la lisibilité du code.

Structure d'une macro, commentaires et bloc With

Écrire du code propre, lisible, documenté et optimisé selon les standards professionnels.

Une macro VBA doit être structurée avec rigueur. L'application systématique des **commentaires**, d'une **indentation soignée** et des **blocs d'instructions With** transforme un script opaque en code limpide et ultra-performant.

✓ 1. Les Commentaires (Apostrophe ')

Toute ligne commençant par une **apostrophe** ' est ignorée par Excel lors de l'exécution et s'affiche en **vert**.

Indispensable pour expliquer *pourquoi* vous faites une action, préciser la date de modification ou désactiver temporairement une ligne de code sans la supprimer.

2. L'Indentation (Touche Tab)

Décaler le code vers la droite à l'intérieur de chaque bloc (procédures, boucles, conditions) avec la touche `Tab`.

Permet d'identifier immédiatement le début et la fin logique de chaque bloc de calcul.

3. Optimisation Majeure : Le Bloc With ... End With

Le bloc `With` évite de répéter le nom d'un même objet plusieurs fois d'affilée. Il accélère la vitesse d'exécution et allège le code :

CODE LOURD SANS WITH

```
Range("A1:E1").Font.Bold = True
Range("A1:E1").Font.Size = 12
Range("A1:E1").Interior.Color = 65535
Range("A1:E1").HorizontalAlignment = xlCenter
```

CODE ÉLÉGANT AVEC WITH

```
With Range("A1:E1")
    .Font.Bold = True
    .Font.Size = 12
    .Interior.Color = 65535
    .HorizontalAlignment = xlCenter
End With
```

✓ **Coupure de ligne longue (_) :** Si une ligne de code dépasse la largeur de votre écran, insérez un **espace suivi d'un tiret bas _** puis appuyez sur Entrée. VBA interprétera la suite comme faisant partie de la même ligne.

Initiation à l'utilisation des variables et types

Déclarer, typer et manipuler des valeurs dynamiques avec Option Explicit et l'instruction Dim.

Une **variable** est une boîte mémoire nommée qui conserve temporairement une valeur pendant l'exécution du script (un taux de TVA, un prénom, un numéro de ligne). Déclarer explicitement ses variables évite 90% des bugs de frappe.

La Règle d'Or : "Option Explicit" en Tête de Module

Écrivez `Option Explicit` sur la toute première ligne de chaque module. Cela force VBA à **refuser l'exécution** si une variable n'a pas été déclarée avec `Dim`, ce qui détecte immédiatement les fautes d'orthographe dans vos noms de variables.

Activation automatique : VBE > Outils > Options > Cocher "Déclaration des variables obligatoire".

TYPE VBA	NATURE & PLAGE DE VALEURS	MÉMOIRE	EXEMPLE DE DÉCLARATION
`Long`	Nombre entier (-2 milliards à +2 milliards)	4 octets	`Dim derLig As Long`
`Double`	Nombre décimal à virgule flottante	8 octets	`Dim tauxTVA As Double`
`String`	Texte et chaînes de caractères	Variable	`Dim nomClient As String`
`Boolean`	Valeur logique (`True` ou `False`)	2 octets	`Dim estValide As Boolean`
`Date`	Date et heure calendaires	8 octets	`Dim dateCloture As Date`
`Worksheet`	Objet Feuille de calcul (Objet complexe)	Pointeur	`Dim ws As Worksheet`
`Variant`	Type universel par défaut (Reçoit tout)	16+ octets	`Dim temp As Variant`

SYNTAXE DE DÉCLARATION ET D'AFFECTATION — Bonnes Pratiques

```vba
Sub CalculerRemise()
    Dim montantHT As Double
    Dim tauxRemise As Double
    Dim ws As Worksheet

    ' 1. Affectation de variables simples (sans Set)
    montantHT = Range("B4").Value
    tauxRemise = 0.05

    ' 2. Affectation d'un objet complexe (OBLIGATION d'utiliser le mot-clé Set)
    Set ws = ThisWorkbook.Worksheets("Facturation")
    ws.Range("B5").Value = montantHT * (1 - tauxRemise)
End Sub
```

Le piège du Dim multiple : Si vous écrivez `Dim a, b, c As Long`, seule la variable c sera de type Long ! Les variables a et b seront créées en **Variant**. Il faut écrire : `Dim a As Long, b As Long, c As Long`.

Boîtes de dialogue interactives : MsgBox et InputBox

Créer des alertes, demander des confirmations OUI/NON et récupérer des saisies utilisateurs sécurisées.

Les boîtes de dialogue permettent à votre macro de communiquer avec l'utilisateur : lui afficher un rapport de fin de traitement, lui demander confirmation avant une suppression critique ou lui faire saisir une date.

1. MsgBox Simple (Information)

Affiche un simple bouton OK pour informer l'utilisateur :

```
MsgBox "Traitement terminé avec succès !", _
       vbInformation, _
       "Clôture Mensuelle"
```

Constantes d'icônes : `vbInformation` (i bleu), `vbExclamation` (! jaune), `vbCritical` (X rouge).

2. MsgBox Interactive (OUI / NON)

Récupère le choix de l'utilisateur dans une variable :

```
Dim rep As VbMsgBoxResult
rep = MsgBox("Voulez-vous purger la base ?", _
             vbYesNo + vbQuestion, _
             "Confirmation")

If rep = vbYes Then
    Range("A2:F500").ClearContents
Else
    MsgBox "Opération annulée.", vbInformation
End If
```

3. InputBox & Application.InputBox Typé

Pour demander une valeur à l'utilisateur, préférez `Application.InputBox` qui permet de **forcer le type de donnée** :

```
Dim anneeSaisie As Variant
' Type:=1 impose un nombre obligatoire (empêche de saisir des lettres)
anneeSaisie = Application.InputBox(Prompt:="Entrez l'année d'exercice :", _
                                   Title:="Sélection Année", _
                                   Default:=2026, _
                                   Type:=1)

If anneeSaisie = False Then
    MsgBox "Vous avez cliqué sur Annuler.", vbExclamation
    Exit Sub
End If
```

Types d'Application.InputBox : `Type:=1` (Nombre), `Type:=2` (Texte), `Type:=8` (Sélection d'une cellule avec la souris).

Règle de syntaxe avec parenthèses : Si vous récupérez le résultat d'une MsgBox ou InputBox dans une variable (ex: `rep = MsgBox(...)`), les parenthèses sont **obligatoires**. Si vous n'attendez aucun retour (simple affichage), n'écrivez **pas de parenthèses** (ex: `MsgBox "Hello"`).

Structures conditionnelles : If...Then...Else et Select Case

Donner de l'intelligence à vos macros pour adapter les traitements en fonction des données.

Les structures conditionnelles permettent à la macro d'évaluer une situation et d'emprunter des branches d'exécution différentes selon que le test est VRAI ou FAUX.

1. La Structure If...ElseIf...Else...End If

```
Dim ca As Double
ca = Range("B4").Value

If ca >= 100000 Then
    Range("C4").Value = "Objectif Dépassé (+15%)"
    Range("C4").Interior.Color = RGB(209, 250, 229)
ElseIf ca >= 50000 Then
    Range("C4").Value = "Objectif Atteint"
Else
    Range("C4").Value = "Sous-Performance"
    Range("C4").Interior.Color = RGB(254, 226, 226)
End If
```

2. La Structure Select Case (Multi-choix)

Beaucoup plus lisible qu'une cascade de 10 `ElseIf` :

```
Dim statut As String
statut = Range("D2").Value

Select Case statut
    Case "VIP", "PREMIUM"
        tauxRemise = 0.20
    Case "STANDARD"
        tauxRemise = 0.05
    Case "PROSPECT"
        tauxRemise = 0.0
    Case Else
        MsgBox "Statut client inconnu !"
End Select
```

Opérateurs Logiques Combinés

Vous pouvez combiner plusieurs conditions dans un même test avec `And` (ET), `Or` (OU) et `Not` (NON) :

```
If (ca > 50000 And region = "Nord") Or clientVIP = True Then
    Call ValiderDossier
End If
```

Syntaxe sur une seule ligne : Si vous n'avez qu'une seule instruction à exécuter, vous pouvez l'écrire sur la même ligne sans `End If` : `If x < 0 Then x = 0`.

Boucles d'automatisation : For...Next et For Each

Répéter une opération sur 10 000 lignes ou parcourir tous les onglets en une fraction de seconde.

Les **boucles** représentent la véritable raison d'être de la programmation. Ce qui prendrait 4 heures de manipulations répétitives à la main est exécuté en quelques millisecondes par une boucle VBA.

1. Boucle Numérique For...Next

Répète un bloc avec un compteur indexé :

```
Dim i As Long, derLig As Long
derLig = Cells(Rows.Count, "A").End(xlUp).Row

For i = 2 To derLig
    ' Multiplie la colonne B par la colonne C
    Cells(i, "D").Value = Cells(i, "B").Value *
Cells(i, "C").Value
Next i
```

Pour reculer de bas en haut (indispensable pour supprimer des lignes) : `For i = derLig To 2 Step -1`.

2. Boucle For Each (Sur Collection)

Parcourt chaque objet d'une collection sans indice :

```
Dim ws As Worksheet

' Déprotéger et afficher tous les onglets
For Each ws In ThisWorkbook.Worksheets
    ws.Visible = xlSheetVisible
    ws.Unprotect Password:="Admin2026"
Next ws
```

3. Boucle Conditionnelle Do While ... Loop

Répète le bloc **tant qu'une condition reste vraie** (très utile pour descendre une colonne jusqu'à la première case vide) :

```
Dim ligne As Long: ligne = 2
Do While Cells(ligne, 1).Value <> ""
    Cells(ligne, 2).Value = UCase(Cells(ligne, 2).Value) ' Met en MAJUSCULES
    ligne = ligne + 1
Loop
```

⚠ **Sortie prématurée d'urgence :** Si une condition exceptionnelle survient dans votre boucle, utilisez l'instruction `Exit For` ou `Exit Do` pour interrompre la répétition et passer immédiatement à la suite sans bloquer Excel.

Gestion et manipulation des feuilles de calcul

Créer, renommer, dupliquer, réordonner, masquer et supprimer des onglets sans alertes intempestives.

L'orchestration des onglets via la collection `Worksheets` est au cœur des macros de consolidation. Elle permet de générer automatiquement un onglet par client, de dupliquer un modèle ou d'épurer les feuilles inutiles.

OPÉRATIONS FONDAMENTALES SUR LES FEUILLES (WORKSHEETS) — Snippets Essentiels

```
' 1. Ajouter une nouvelle feuille en dernière position et la nommer
Dim wsNew As Worksheet
Set wsNew = Worksheets.Add(After:=Worksheets(Worksheets.Count))
wsNew.Name = "Rapport_" & Format(Date, "yyyymmdd")

' 2. Dupliquer une feuille modèle existante
Worksheets("Modele_Facture").Copy After:=Worksheets("Modele_Facture")
ActiveSheet.Name = "Client_Dupont"

' 3. Supprimer une feuille sans message de confirmation Windows
Application.DisplayAlerts = False  ' Désactive la boîte "Êtes-vous sûr ?"
Worksheets("Temp_Import").Delete
Application.DisplayAlerts = True   ' Toujours réactiver après !
```

Les 3 Niveaux de Visibilité d'une Feuille (.Visible)

CONSTANTE VBA	VISIBILITÉ DANS L'INTERFACE EXCEL	COMMENT LA RÉAFFICHER ?
`xlSheetVisible` (-1)	Onglet parfaitement visible et sélectionnable.	État normal
`xlSheetHidden` (0)	Onglet masqué standard.	Clic droit > *Afficher...*
`xlSheetVeryHidden` (2)	**Ultra-Masqué :** N'apparaît même pas dans le menu *Afficher...* d'Excel !	**Uniquement par code VBA** ou dans la fenêtre Propriétés de VBE.

Protection des tables de paramètres : Utilisez `Worksheets("Config_Admin").Visible = xlSheetVeryHidden` pour masquer totalement vos tables de prix ou règles sensibles. Vos utilisateurs ne sauront même pas que l'onglet existe.

Gestion des classeurs et automatisation à l'ouverture

Ouvrir, enregistrer, fermer des fichiers externes et déclencher des actions automatiques au démarrage.

VBA permet de piloter plusieurs classeurs simultanément. Vous pouvez ouvrir un fichier distant, y puiser des données, le fermer silencieusement et exécuter des tâches d'accueil automatiques dès que l'utilisateur ouvre le classeur.

PILOTER DES FICHIERS EXTERNES (WORKBOOKS) — Multi-Classeurs

```
Dim wbSource As Workbook, chemin As String
chemin = ThisWorkbook.Path & "\Donnees_Brutes.xlsx"

' 1. Ouvrir le fichier en mode lecture seule (plus rapide)
Set wbSource = Workbooks.Open(Filename:=chemin, ReadOnly:=True)

' 2. Copier les données de la source vers notre classeur
wbSource.Worksheets(1).Range("A1:D100").Copy _
    Destination:=ThisWorkbook.Worksheets("Donnees").Range("A1")

' 3. Fermer le fichier source sans enregistrer de modifications
wbSource.Close SaveChanges:=False
```

Méthode 1 : Workbook_Open (Recommandé)

À placer impérativement dans le module d'objet **ThisWorkbook** :

```
Private Sub Workbook_Open()
    Worksheets("Accueil").Activate
    Range("A1").Select
    MsgBox "Bienvenue dans l'outil de gestion !", _
           vbInformation, "Système 2026"
End Sub
```

Méthode 2 : Auto_Open (Historique)

À placer dans un **Module Standard** (`Module1`) :

```
Sub Auto_Open()
    ' S'exécute automatiquement au lancement
    Call ActualiserTousLesTCD
End Sub
```

Compatible avec les anciennes versions d'Excel.

ThisWorkbook vs ActiveWorkbook : `ThisWorkbook` désigne TOUJOURS le classeur contenant le code VBA en train de tourner. `ActiveWorkbook` désigne le classeur qui a le focus à l'écran (qui peut être le fichier externe ouvert temporairement). Utilisez toujours `ThisWorkbook` pour référencer vos feuilles de calcul principales !

Navigation dynamique : Range, Cells, End(xlUp) et Offset

Trouver la dernière ligne renseignée et naviguer dans la grille sans jamais coder de coordonnées figées.

L'erreur n°1 des macros enregistrées est d'écrire en dur `Range("A1:D50")`. Le mois suivant, votre tableau compte 75 lignes et les 25 dernières sont ignorées ! La technique **End(xlUp)** résout ce problème définitivement.

→ La Formule Sacrée : Trouver la Dernière Ligne Renseignée

Équivaut à se positionner tout en bas de la feuille (ligne 1 048 576) et faire `Ctrl + Flèche Haut` :

```
Dim derLig As Long
derLig = Cells(Rows.Count, "A").End(xlUp).Row

' Pour trouver la dernière colonne vers la droite :
Dim derCol As Long
derCol = Cells(1, Columns.Count).End(xlToLeft).Column
```

Range("A1") vs Cells(Ligne, Colonne)

Range : Idéal pour les plages fixes (`Range("A1:C10")`).

Cells : Indispensable dans les boucles car les coordonnées sont des **nombres** : `Cells(i, 2)` désigne la ligne `i`, colonne 2 (B).

Combinaison Ultime :
`Range(Cells(1, 1), Cells(derLig, derCol))`

Offset & CurrentRegion

CurrentRegion : Sélectionne tout le bloc de cellules contiguës (équivalent à `Ctrl + A` sur une cellule) :

`Range("A1").CurrentRegion.Select`

Offset(Lig, Col) : Décalage relatif :
`Range("A1").Offset(1, 0)` → Cellule A2.

EXEMPLE PRATIQUE : AJOUTER UNE NOUVELLE LIGNE À LA FIN D'UN TABLEAU — Ajout Dynamique

```
Sub AjouterNouveauClient()
    Dim ligneVide As Long
    ligneVide = Cells(Rows.Count, "A").End(xlUp).Row + 1

    Cells(ligneVide, "A").Value = Now             ' Date et Heure
    Cells(ligneVide, "B").Value = "SARL ACME"      ' Nom Client
    Cells(ligneVide, "C").Value = 8500             ' Montant
End Sub
```

✓ **Vitesse x100 :** Évitez absolument les `.Select` et `.Activate` générés par l'enregistreur ! Écrivez directement `Range("A1").Value = "OK"` au lieu de `Range("A1").Select` suivi de `ActiveCell.FormulaR1C1 = "OK"`.

Mise en forme automatisée et accélération des calculs

Appliquer des chartes graphiques professionnelles et multiplier la vitesse d'exécution par 50.

Une macro d'ingénierie doit non seulement produire des résultats visuellement impeccables, mais elle doit s'exécuter de façon instantanée. L'activation des **accélérateurs système VBA** élimine le scintillement d'écran et décuple les performances.

Le Patron de Performance Ultime (Booster x50)

Désactivez les composants lourds d'Excel en début de procédure et réactivez-les impérativement à la fin :

```vba
Sub MacroUltraRapide()
    ' === 1. DÉSACTIVATION DES RALENTISSEURS ===
    Application.ScreenUpdating = False        ' Fige l'écran (aucun scintillement)
    Application.DisplayAlerts = False         ' Masque les confirmations Windows
    Application.Calculation = xlCalculationManual ' Suspend les recalculs de formules

    ' === 2. CORPS DU TRAITEMENT (VOS 10 000 CALCULS) ===
    ' ... Traitement lourd ...

    ' === 3. RÉACTIVATION OBLIGATOIRE ===
    Application.Calculation = xlCalculationAutomatic
    Application.DisplayAlerts = True
    Application.ScreenUpdating = True
End Sub
```

Mise en Forme et Formats Numériques en VBA

BORDURES & COULEURS

```vba
With Range("A1:D1")
    .Interior.Color = RGB(30, 41, 59)
    .Font.Color = RGB(255, 255, 255)
    .Font.Bold = True
    .Borders.LineStyle = xlContinuous
End With
```

FORMATS DE NOMBRES

```vba
' Format Monétaire Euro
Columns("C").NumberFormat = "#,##0.00 €"

' Format Pourcentage
Columns("D").NumberFormat = "0.0%"

' Format Date Courte
Columns("A").NumberFormat = "dd/mm/yyyy"
```

Gestion d'erreur impérative : Si votre code plante pendant que `ScreenUpdating = False`, Excel restera figé pour l'utilisateur. Intégrez toujours une gestion d'erreur `On Error GoTo FinPropre` pour garantir la réactivation des propriétés quoi qu'il arrive.

Importation de données externes et fichiers CSV

Automatiser l'ingestion de fichiers textes bruts, gérer les séparateurs et consolider les données.

L'export de données sous format **CSV (Comma-Separated Values)** ou texte tabulé est le format d'échange standard des logiciels ERP (SAP, Sage, Salesforce). VBA permet d'importer et de parser ces flux sans aucune intervention humaine.

SCRIPT UNIVERSEL D'IMPORTATION DE FICHIER CSV AVEC SÉLECTION DE FICHIER — Import Robuste

```
Sub ImporterFichierCSV()
    Dim fd As FileDialog, cheminFichier As String
    Dim wsCible As Worksheet

    ' 1. Boîte de dialogue pour sélectionner le fichier sur le disque
    Set fd = Application.FileDialog(msoFileDialogFilePicker)
    fd.Title = "Sélectionnez le fichier CSV d'export ERP"
    fd.Filters.Clear
    fd.Filters.Add "Fichiers CSV", "*.csv"

    If fd.Show = -1 Then
        cheminFichier = fd.SelectedItems(1)
    Else
        Exit Sub ' L'utilisateur a cliqué sur Annuler
    End If

    ' 2. Préparer la feuille cible
    Set wsCible = ThisWorkbook.Worksheets("Donnees_Import")
    wsCible.Cells.ClearContents

    ' 3. Ouvrir le CSV avec le bon séparateur (Point-virgule Semicolon:=True)
    Workbooks.OpenText Filename:=cheminFichier, _
                       DataType:=xlDelimited, _
                       Semicolon:=True, _
                       Local:=True

    ' 4. Copier vers notre classeur et fermer le fichier brut
    ActiveWorkbook.ActiveSheet.UsedRange.Copy Destination:=wsCible.Range("A1")
    ActiveWorkbook.Close SaveChanges:=False

    MsgBox "Importation terminée avec succès !", vbInformation
End Sub
```

Le Paramètre "Local:=True"

Par défaut, VBA utilise les paramètres régionaux américains (point pour les décimales, virgule pour séparateur). En précisant `Local:=True`, VBA respecte vos paramètres régionaux Windows français (virgule décimale et point-virgule).

Consolidation Multi-Fichiers (Boucle Dir)

La fonction `Dir("C:\Exports\*.xlsx")` permet de boucler sur tous les fichiers d'un dossier pour les copier les uns sous les autres automatiquement en 1 seule passe.

Alternative Power Query : Si l'importation ne nécessite aucune interaction conditionnelle complexe, vous pouvez piloter le rafraîchissement d'une requête Power Query directement en VBA via : `ThisWorkbook.Connections("NomRequete").Refresh`.

Atelier : Générateur de rapport mensuel & Export PDF

Synthèse de tous les modules : Import, nettoyage, calculs, mise en page et export PDF en 1 seul clic.

Scénario Métier : Chaque fin de mois, le contrôleur de gestion doit nettoyer un export brut de 500 lignes, supprimer les lignes sans montant, calculer les totaux, appliquer la charte graphique et générer un rapport PDF prêt à être envoyé à la direction.

CODE COMPLET DU MODULE MÉTIER : GENERATEURRAPPORTMENSUEL — Production Ready

```
Option Explicit

Sub GenererRapportMensuelComplet()
    Dim ws As Worksheet, derLig As Long, i As Long
    Dim cheminPDF As String

    On Error GoTo GestionErreur
    Application.ScreenUpdating = False
    Application.DisplayAlerts = False

    Set ws = ThisWorkbook.Worksheets("Ventes")
    derLig = ws.Cells(ws.Rows.Count, "A").End(xlUp).Row

    ' 1. Nettoyage : Supprimer les lignes où le montant en col C est vide ou <= 0
    For i = derLig To 2 Step -1
        If ws.Cells(i, "C").Value <= 0 Or IsEmpty(ws.Cells(i, "C")) Then
            ws.Rows(i).Delete
        End If
    Next i

    ' 2. Recalcul de la nouvelle dernière ligne après suppression
    derLig = ws.Cells(ws.Rows.Count, "A").End(xlUp).Row

    ' 3. Ligne de Totalisation automatique
    ws.Cells(derLig + 1, "A").Value = "TOTAL GÉNÉRAL"
    ws.Cells(derLig + 1, "C").Formula = "=SUM(C2:C" & derLig & ")"

    ' 4. Mise en forme du tableau corporate
    With ws.Range("A1:C1")
        .Interior.Color = RGB(79, 70, 229)  ' Indigo Header
        .Font.Color = RGB(255, 255, 255)
        .Font.Bold = True
    End With
    ws.Range("C2:C" & derLig + 1).NumberFormat = "#,##0.00 €"
    ws.Columns("A:C").AutoFit

    ' 5. Exportation instantanée au format PDF
    cheminPDF = ThisWorkbook.Path & "\Rapport_Mensuel_" & Format(Date, "yyyy-mm") & ".pdf"
    ws.ExportAsFixedFormat Type:=xlTypePDF, Filename:=cheminPDF, Quality:=xlQualityStandard

    ' 6. Clôture et notification
    Application.ScreenUpdating = True
    MsgBox "Rapport généré et exporté avec succès en PDF :" & vbCrLf & cheminPDF, _
           vbInformation, "Succès"
    Exit Sub

GestionErreur:
    Application.ScreenUpdating = True
    MsgBox "Erreur survenue : " & Err.Description, vbCritical, "Erreur #" & Err.Number
End Sub
```

✓ **Résultat :** Affectez cette macro à un bouton sur votre tableau de bord. Votre rapport mensuel complet (nettoyage, calculs, mise en forme et PDF) est produit en **moins de 0,8 seconde** !

Guide de survie VBE & Top 15 des Snippets VBA

Les raccourcis indispensables, les 15 lignes de code incontournables et la checklist de sécurité.

Raccourcis Clavier VBE Indispensables

`Alt + F11`	Bascule Excel ⇄ Éditeur VBE
`F8` / `Maj+F8`	Exécution pas à pas détaillé / principal
`F5`	Exécuter la macro à pleine vitesse
`F9`	Poser / retirer un point d'arrêt
`Ctrl + G`	Ouvrir la fenêtre Exécution immédiate
`Ctrl + R` / `F4`	Explorateur de projets / Propriétés
`Ctrl + Espace`	Forcer l'autocomplétion IntelliSense

Checklist de Sécurité avant Déploiement

- `Option Explicit` présent en tête de chaque module.
- Toutes les variables déclarées et typées (`Dim ... As`).
- Aucun `.Select` ou `.Activate` inutile.
- Dernière ligne calculée avec `End(xlUp).Row`.
- `ScreenUpdating` et `DisplayAlerts` réactivés à `True`.
- Gestion d'erreurs `On Error GoTo` en place.
- Fichier enregistré au format `.xlsm` ou `.xlsb`.

Top 15 des Snippets VBA les Plus Utiles

OBJECTIF MÉTIER	INSTRUCTION VBA EN 1 LIGNE
Dernière ligne colonne A	`derLig = Cells(Rows.Count, "A").End(xlUp).Row`
Figer l'écran (Accélération)	`Application.ScreenUpdating = False`
Supprimer sans confirmation	`Application.DisplayAlerts = False`
Effacer valeurs sans formats	`Range("A2:F100").ClearContents`
Copier-coller valeurs seules	`Range("A1:B10").Copy: Range("D1").PasteSpecial xlPasteValues`
Ajuster toutes les colonnes	`Worksheets("Data").Columns.AutoFit`
Date et heure du jour	`Range("A1").Value = Now: Range("A1").NumberFormat = "dd/mm/yyyy hh:mm"`
Couleur de fond RGB	`Range("A1").Interior.Color = RGB(79, 70, 229)`
Masquer onglet ultra-secret	`Worksheets("Admin").Visible = xlSheetVeryHidden`
Ignorer erreurs temporaires	`On Error Resume Next ' À utiliser avec parcimonie`
Rétablir gestion erreurs	`On Error GoTo 0`
Quitter macro immédiatement	`Exit Sub`
Actualiser tous les TCD	`ThisWorkbook.RefreshAll`
Boîte confirmation Oui/Non	`If MsgBox("Continuer ?", vbYesNo + vbQuestion) = vbNo Then Exit Sub`
Chemin du dossier actif	`dossier = ThisWorkbook.Path & "\"`