**Manuel de formation**

**Excel – Macros & Programmation VBA**

Table des matières

[1. Introduction aux macros et à l’automatisation 4](#_Toc214959502)

[1.1 Qu’est-ce qu’une macro ? Principe et utilité 4](#_Toc214959503)

[1.2 Utiliser l’enregistreur de macros 4](#_Toc214959504)

[1.3 Exemples pratiques d’automatisation 5](#_Toc214959505)

[1.4 Affecter une macro à un bouton, un objet, une barre d’outils 5](#_Toc214959506)

[1.5 Générer un graphique par macro 5](#_Toc214959507)

[1.6 Enregistrer une macro dans le classeur personnel (Personal Macro Workbook – Perso.xlsb) 6](#_Toc214959508)

[2. Environnement de développement VBA (VBE) 7](#_Toc214959509)

[2.1 Découvrir l’environnement Visual Basic Editor (VBE) 7](#_Toc214959510)

[2.2 Notions de Programmation Par Objets (PPO / POO) 8](#_Toc214959511)

[2.3 Créer une procédure 8](#_Toc214959512)

[2.4 Comprendre la syntaxe des procédures 8](#_Toc214959513)

[2.5 Affecter un bouton à une procédure 9](#_Toc214959514)

[2.6 Utiliser les modules et organiser le code 9](#_Toc214959515)

[2.7 Raccourcis utiles dans VBE 9](#_Toc214959516)

[3. VBA et les objets Excel 9](#_Toc214959517)

[3.1 L’objet Range et ses extensions 9](#_Toc214959518)

[3.2 L’objet Worksheet (Feuille de calcul) 10](#_Toc214959519)

[3.3 L’objet Workbook (Classeur) 11](#_Toc214959520)

[3.4 Interaction entre feuilles et classeurs 11](#_Toc214959521)

[3.5 Comment structurer et commenter son code 11](#_Toc214959522)

[4. Bases de la programmation VBA 12](#_Toc214959523)

[4.1 Les variables : déclaration, typage et portée 12](#_Toc214959524)

[4.2 Les variables objets 12](#_Toc214959525)

[4.3 Les instructions de programmation 12](#_Toc214959526)

[4.3.1 Les conditions 12](#_Toc214959527)

[4.3.2 Les boucles 13](#_Toc214959528)

[4.4 Les boîtes de dialogue prédéfinies (MsgBox, InputBox) 14](#_Toc214959529)

[4.5 Les fonctions 14](#_Toc214959530)

[5. Débogage et traitement d’erreurs 15](#_Toc214959531)

[5.1 Les types d’erreur 15](#_Toc214959532)

[5.2 Utiliser le déroulement pas à pas (F8) 15](#_Toc214959533)

[5.3 Placer des "espions" pour identifier les problèmes 15](#_Toc214959534)

[5.4 Fenêtre d’exécution (Immediate Window) 15](#_Toc214959535)

[5.5 Gestion simple des erreurs 16](#_Toc214959536)

[6. Projet professionnel 16](#_Toc214959537)

[6.1 Objectif du mini-projet 16](#_Toc214959538)

[6.2 Exemples de mini-projets 16](#_Toc214959539)

[6.3 Présentation des résultats 17](#_Toc214959540)

[6.4 Documentation du projet 17](#_Toc214959541)

# 1. Introduction aux macros et à l’automatisation

## 1.1 Qu’est-ce qu’une macro ? Principe et utilité

**Définition :**  
Une macro est une **suite d’instructions** qui permet d’automatiser une ou plusieurs actions dans Excel (mise en forme, calculs, génération de rapport…).

**Idée clé :**

Une macro = un “script” qui rejoue automatiquement ce que vous feriez manuellement.

**Utilité :**

* Gagner du temps sur les tâches répétitives (rapports mensuels, mises en forme standardisées…).
* Réduire les erreurs humaines.
* Standardiser les traitements dans une équipe.
* Création d’outils internes (petites applications Excel).

## 1.2 Utiliser l’enregistreur de macros

L’enregistreur est le point d’entrée idéal pour débuter en VBA.

**Activation de l’onglet Développeur :**

1. Fichier → Options.
2. Personnaliser le ruban.
3. Cocher **Développeur** → OK.

**Enregistrer une macro simple :**

1. Onglet **Développeur** → **Enregistrer une macro**.
2. Donner un nom à la macro (ex. MiseEnFormeRapport).
3. (Optionnel) Assigner un raccourci clavier (ex. Ctrl+Maj+R).
4. Choisir où stocker la macro :
   * Ce classeur
   * Nouveau classeur
   * Classeur de macros personnelles (Perso.xlsb – voir plus loin).
5. Réaliser les actions à automatiser (par ex. :
   * Sélectionner une zone
   * Appliquer un format tableau
   * Mettre le titre en gras, centré, taille 14
   * Mettre le fond en gris
6. Cliquer sur **Arrêter l’enregistrement**.

**Limites de l’enregistreur :**

* Génère un code parfois peu lisible.
* Enregistre tout, même les clics "inutiles".
* Ne gère pas la logique conditionnelle (si ceci, alors cela…).

## 1.3 Exemples pratiques d’automatisation

**Exemple 1 : Mise en forme automatique d’un rapport**

* Sélectionner un tableau de données.
* Appliquer une mise en forme conditionnelle.
* Ajuster les largeurs de colonnes.
* Ajouter une ligne de total.

Possible enregistrement → macro FormaterRapportMensuel.

**Exemple 2 : Nettoyage d’une base de données**

* Supprimer les lignes vides.
* Appliquer un format date standard.
* Remplacer des valeurs (ex. N/A → vide).

## 1.4 Affecter une macro à un bouton, un objet, une barre d’outils

**A. Bouton de formulaire sur la feuille :**

1. Onglet **Développeur** → **Insérer** → **Bouton (Contrôle de formulaire)**.
2. Dessiner le bouton sur la feuille.
3. La boîte de dialogue **Affecter une macro** s’ouvre → choisir la macro → OK.
4. Renommer le texte du bouton (clic droit → Modifier le texte).

**B. Forme (objet graphique) :**

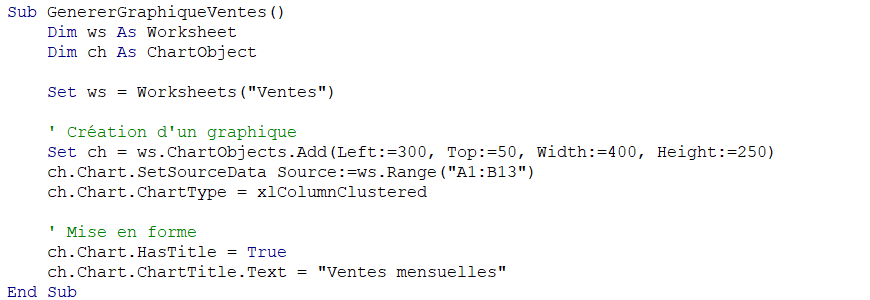
1. Insertion → Formes → choisir une forme (rectangle, etc.).
2. Clic droit → **Affecter une macro…** → choisir la macro.

**C. Barre d’outils / Ruban :**

* Personnalisation avancée (optionnelle) pour ajouter un bouton à la barre d’accès rapide ou au ruban.

## 1.5 Générer un graphique par macro

Exemple simple de macro générée par enregistrement, puis nettoyée :



## 1.6 Enregistrer une macro dans le classeur personnel (Personal Macro Workbook – Perso.xlsb)

Le **classeur personnel** est un classeur caché qui s’ouvre à chaque démarrage d’Excel et qui permet de stocker des macros “globales”.

**Étapes :**

1. Lancer l’enregistrement d’une macro.
2. Dans **Stocker la macro dans** → choisir **Classeur de macros personnelles**.
3. Arrêter l’enregistrement.
4. Excel crée automatiquement le fichier PERSONAL.XLSB.

Pour afficher le classeur :

* Vue → **Réafficher** → choisir PERSONAL.XLSB.

### Enregistrer le fichier Excel ave prise en charge de Macro (xlsm)

Lorsque vous créez une macro dans Excel, il est essentiel d’enregistrer votre classeur dans un format compatible avec le langage VBA.  
Par défaut, Excel utilise le format **.xlsx**, qui **ne prend pas en charge les macros**.  
Cela signifie que **toute macro enregistrée ou écrite sera perdue** si vous enregistrez votre fichier dans ce format.

**➡ Pourquoi changer de format ?**

Les macros sont du **code exécuté dans un environnement sécurisé**.  
Excel ne permet pas l’exécution de code dans les fichiers .xlsx, d’où la nécessité de passer au format **.xlsm**.

**➡ Comment enregistrer votre classeur avec prise en charge des macros ?**

1. Cliquez sur **Fichier > Enregistrer sous**
2. Choisissez un emplacement (Bureau, Documents…).
3. Dans la liste **Type de fichier**, sélectionnez :  
   **Classeur Excel prenant en charge les macros (\*.xlsm)**
4. Donnez un nom à votre fichier, par exemple :  
   Projet\_Automatisation.xlsm
5. Cliquez sur **Enregistrer**

**➡ Changement visuel de l’icône**

Lorsque votre fichier est enregistré au format .xlsm, l’icône du classeur devient légèrement différente pour indiquer qu’il contient des macros.

**➡ Avertissement de sécurité au prochain lancement**

À l’ouverture d’un fichier .xlsm, Excel affiche généralement un bandeau jaune indiquant :

**“Sécurité des macros – Les macros ont été désactivées.”**

Ce bloqueur de sécurité existe car le VBA peut contenir du code malveillant.  
Vous devez cliquer sur :

* **Activer le contenu**

**Seulement si vous faites confiance au fichier et à son auteur.**

**➡ Résumé de bonnes pratiques**

* Toujours enregistrer vos fichiers VBA en **.xlsm**
* N’activer les macros que si la source est fiable
* Ajouter une signature numérique pour les macros en entreprise
* Informer les utilisateurs avant de distribuer un fichier macro

# 2. Environnement de développement VBA (VBE)

## 2.1 Découvrir l’environnement Visual Basic Editor (VBE)

**Accès :**

* Onglet **Développeur** → **Visual Basic**  
  ou
* Raccourci : Alt + F11.

**Éléments principaux :**

* **Explorateur de projet (Project Explorer)** : liste des classeurs ouverts, feuilles, modules.
* **Fenêtre Propriétés** : propriétés de l’objet sélectionné (sheet, userform…).
* **Fenêtre Code** : zone d’édition du code.
* **Fenêtre Exécution (Immediate Window)** : pour tester des commandes en direct.

## 2.2 Notions de Programmation Par Objets (PPO / POO)

Dans Excel, tout est un **objet** :

* Application
* Workbook (classeur)
* Worksheet (feuille)
* Range (cellules, plages)
* Chart, ListObject, etc.

Chaque objet a :

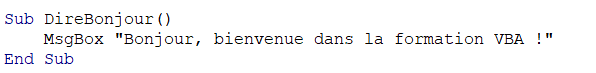
* Des **propriétés** (ex : Range("A1").Value, Worksheet.Name).
* Des **méthodes** (ex : Range("A1").ClearContents, Worksheet.Activate).
* Des **événements** (ex : Worksheet\_Change, Workbook\_Open).

## 2.3 Créer une procédure

Deux principaux types de procédures :

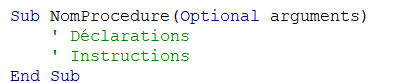
* **Sub** (procédure) : exécute une action.
* **Function** : retourne une valeur (utilisable dans des formules Excel ou dans le code).

Exemple Sub simple :

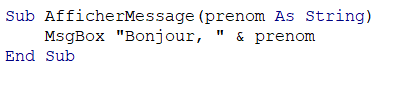


## 2.4 Comprendre la syntaxe des procédures

Structure d’une Sub :



Exemple avec paramètre :



## 2.5 Affecter un bouton à une procédure

Même principe que pour une macro enregistrée :

* Insérer un bouton (Contrôle de formulaire).
* Clic droit → **Affecter une macro** → choisir la Sub que tu as écrite.

## 2.6 Utiliser les modules et organiser le code

**Types de modules :**

* Module standard (Module1, Module2…) : pour les macros génériques.
* Modules de feuille : code spécifique à une feuille (événements, etc.).
* Module ThisWorkbook : événements liés au classeur (ouverture, fermeture…).
* Modules de classe (plus avancé).

**Bonne pratique :**

* Créer un module par “thème” ou par fonctionnalité.
* Renommer les modules (ex : modUtilitaires, modImportDonnees).

## 2.7 Raccourcis utiles dans VBE

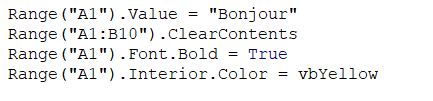
* F5 : exécuter la procédure.
* F8 : exécuter pas à pas.
* Ctrl + Espace : Intellisense (auto-complétion).
* Ctrl + G : afficher/cacher la fenêtre d’exécution (Immediate).
* Ctrl + R : afficher/cacher l’explorateur de projet.

# 3. VBA et les objets Excel

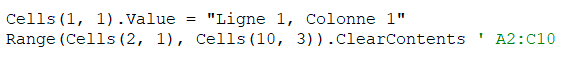
## 3.1 L’objet Range et ses extensions

Range représente une cellule ou une plage.

Exemples :



Références relatives via Cells :



**Propriétés courantes :**

* .Value
* .Font, .Interior, .NumberFormat
* .Offset(RowOffset, ColumnOffset) pour se déplacer par rapport à une cellule.

Exemple avec Offset :



## 3.2 L’objet Worksheet (Feuille de calcul)

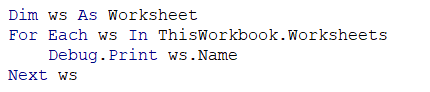
Accès à une feuille :



Changement de nom :



Boucle sur les feuilles :



## 3.3 L’objet Workbook (Classeur)

Accès au classeur :

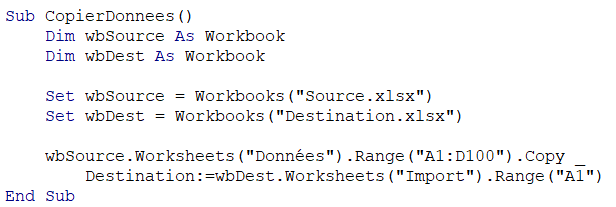


Ouvrir / fermer un classeur :



## 3.4 Interaction entre feuilles et classeurs

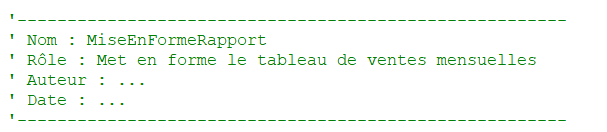
Exemple : copier des données d’un classeur à un autre.



## 3.5 Comment structurer et commenter son code

**Bonnes pratiques :**

* Indenter le code.
* Mettre un en-tête de procédure :

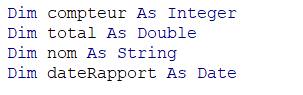


* Commenter les étapes importantes (' pour commenter)
* Utiliser des noms de variables explicites (dernLigne, totalVentes…)

# 4. Bases de la programmation VBA

## 4.1 Les variables : déclaration, typage et portée

**Déclaration :**



**Option Explicit :**

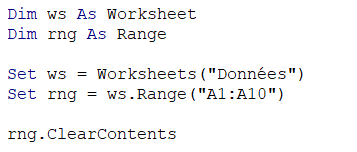
* Ajoute Option Explicit en haut du module pour obliger la déclaration des variables (évite les fautes de frappe).

**Portée :**

* Variable locale (dans une procédure) : accessible uniquement dans la Sub/Function.
* Variable module (Dim en haut du module) : accessible dans toutes les procédures du module.
* Variable globale (Public en haut d’un module standard) : accessible dans tout le projet.

## 4.2 Les variables objets

Exemple :

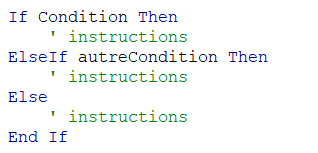


Note : Les variables objets nécessitent le mot-clé **Set**.

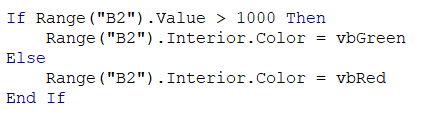
## 4.3 Les instructions de programmation

### 4.3.1 Les conditions

Structure :

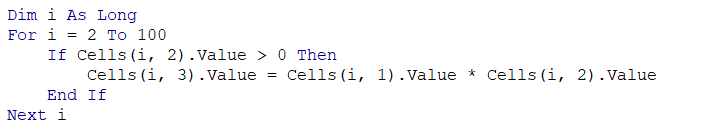


Exemple :

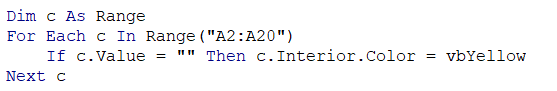


### 4.3.2 Les boucles

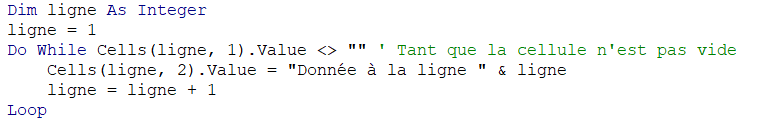
**For…Next :**



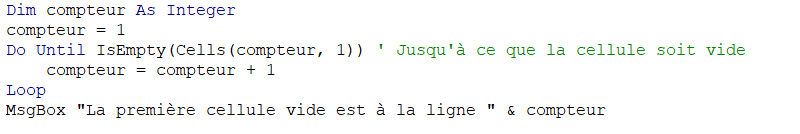
**For Each :**



**Do While :**



**Do Until :**

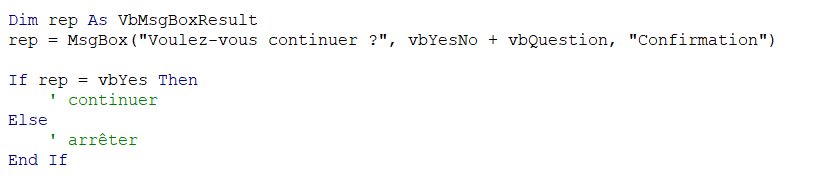


## 4.4 Les boîtes de dialogue prédéfinies (MsgBox, InputBox)

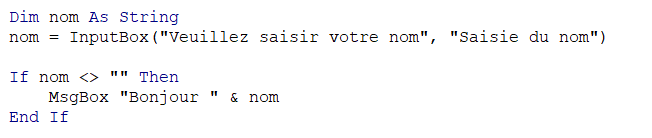
**MsgBox :**



**MsgBox avec boutons :**

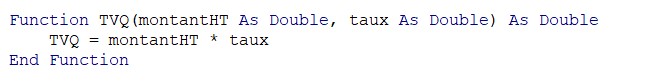


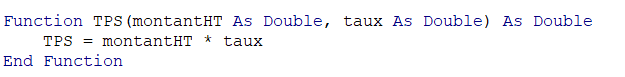
**InputBox :**



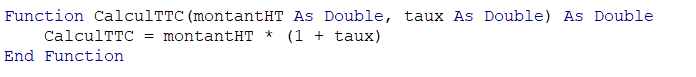
## 4.5 Les fonctions

**Fonction utilisée dans le code :**





**Fonction utilisable dans Excel :**



**Dans Excel :**  
=CalculTTC(A2; 0,2) si A2 contient le montant HT.

# 5. Débogage et traitement d’erreurs

## 5.1 Les types d’erreur

* **Erreur de compilation** : faute de syntaxe, variable non déclarée…
* **Erreur d’exécution** : division par zéro, cellule non trouvée, fichier introuvable…
* **Erreur logique** : le code s’exécute mais ne donne pas le bon résultat.

## 5.2 Utiliser le déroulement pas à pas (F8)

* Place le curseur dans la procédure.
* Appuie sur F8 pour exécuter ligne par ligne.
* Surveille les valeurs de variables dans une **fenêtre Espions** ou en survolant.

## 5.3 Placer des "espions" pour identifier les problèmes

* Menu **Débogage** → **Ajouter un espion**.
* Saisir le nom de la variable à surveiller.
* Voir l’évolution en temps réel lors du pas à pas.

## 5.4 Fenêtre d’exécution (Immediate Window)

* Raccourci : Ctrl + G.
* Permet de tester des lignes de code ponctuelles ou d’afficher des informations :



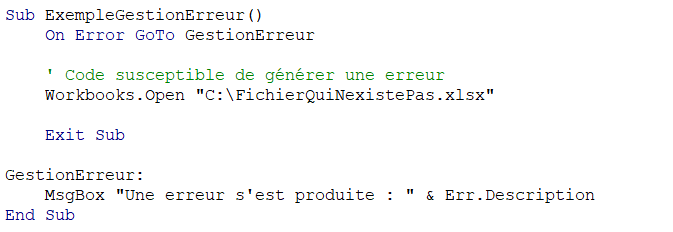
* Permet aussi d’exécuter des instructions :



## 5.5 Gestion simple des erreurs

**On Error Resume Next** (ignorer et continuer – à utiliser avec prudence).

**On Error GoTo** pour gérer l’erreur :



# 6. Projet professionnel

## 6.1 Objectif du mini-projet

Mettre en pratique l’ensemble des notions vues :

* Enregistreur de macros.
* Manipulation d’objets (Range, Worksheet, Workbook).
* Variables, boucles, conditions.
* Boîtes de dialogue.
* Débogage de base.

## 6.2 Exemples de mini-projets

* + 1. **Générateur de rapport de ventes :**
       1. Import des données d’un onglet “Données brutes”.
       2. Nettoyage (suppression des lignes vides, formats).
       3. Calcul des totaux et des moyennes.
       4. Création automatique d’un tableau récapitulatif et d’un graphique.
    2. **Outil de suivi d’actions / tâches :**
       1. Saisie des tâches dans un tableau.
       2. Bouton “Mettre à jour le statut” qui colore les lignes selon l’avancement.
       3. Filtre automatique sur les tâches en retard.
    3. **Outil de facturation simple :**
       1. Sélection d’un client via InputBox ou liste.
       2. Calcul automatique du montant TTC.
       3. Possibilité de générer une facture sur une nouvelle feuille.

## 6.3 Présentation des résultats

Chaque participant doit :

* Expliquer le contexte du mini-projet (problème métier).
* Montrer le fonctionnement de la solution (démonstration).
* Présenter la structure du code (modules, procédures).
* Expliquer les difficultés rencontrées et les solutions adoptées.

## 6.4 Documentation du projet

Livrables possibles :

* Classeur Excel contenant le projet et le code VBA.
* Documentation courte (1 à 3 pages) incluant :
  + Objectif du projet.
  + Description des fonctionnalités.
  + Instructions d’utilisation.
  + Limites et pistes d’amélioration.