Introduction

Visual Basic for Applications est un langage de programmation directement intégré dans Excel. VBA offre la possibilité d'automatiser des tâches dans Excel, notamment le formatage de données, la génération de rapports, ainsi que l'importation, le traitement et l'exportation de données. VBA offre la possibilité de gérer les données (lecture, écriture, modification et analyse) au sein des feuilles de calcul, des classeurs et des objets. Il permet également de créer des applications dans Excel.

Variables

Une variable est un conteneur où on peut stocker des informations, chaque variable a un nom unique. Les variables permettent de conserver des informations importantes pour les réutiliser.

Types de variable

Type	Description
Integer	Nombre entier de -32768 à 32767.
Long	Nombre entier de - 2147483648 à 2147483647.
Single	Nombre à virgule flottante de -3.402823E38 à 3.402823E38.
Double	Nombre à virgule flottante de -1.79769313486232E308 à 1.79769313486232E308.
String	Chaînes de caractères (texte).
Boolean	Vrai ou Faux
Date	Date et heure
Object	Objets Excel (feuilles de calcul, classeurs, cellules, etc.).
Variant	Tout type de données (type par défaut si la variable n'est pas déclarée).

Déclaration de variables

La variable doit être déclarée avant de l'utiliser, souvent déclarée au début du code. Dim est le mot-clé pour déclarer une variable et le nom de la variable ne doit pas contenir d'espace.

Exemple :

```
Sub Exemple()
Dim nom As String 'Déclaration d'une variable nom de type chaîne de caractère
Dim age As Integer 'Déclaration d'une variable age de type entier
Dim date_naissance As Date 'Déclaration d'une variable date_naissance de type date
```

Affectation des variables

```
age = 30
nom = "Robert" 'Les chaînes de caractères doivent être entourées de guillemets
date_naissance = #01/01/1990# 'La date doit être entourée de dièses
```

Utilisation avec des objets Excel

```
'Déclaration de la variable mafeuille de type objet feuille de calcul
Dim mafeuille As Worksheet
'Affectation de la variable mafeuille à la feuille de calcul (feuille)
Set mafeuille = ThisWorkbook.Sheets("feuille")
'Insertion de "valeur insérée" dans la cellule A1
mafeuille.Range("A1").Value = "Valeur inséré"
```

Portée des variables

Portée locale : les variables déclarées à l'intérieur d'une procédure Sub ou Function ne sont accessibles que dans l'intérieur d'une procédure.

Portée globale : les variables déclarées en dehors des procédures sont accessibles dans toutes les procédures du module. Pour déclarer une variable globale, utilisez le mot-clé Public au lieu de Dim.

Exemple :

Public total As Double

Procédures

Sub (macro publique ou globale)

Une macro déclarée avec Sub est accessible dans tous les modules du classeur Excel.

Exemple :

```
Sub procedure1()
Range("A2").Select 'Sélectionne la cellule A2
End Sub
```

Private Sub (macro privée)

Une macro déclarée avec Private Sub est uniquement accessible à l'intérieur du module défini.

Exemple :

```
Private Sub procedure2()
MsgBox "Bonjour" 'Affiche Bonjour dans la boîte de message
End Sub
```

Public Sub (macro publique explicite)

Une macro déclarée avec Public Sub se comporte de la même manière qu'une macro Sub.

Passage de paramètre aux procédures

ByVal

ByVal est utilisé pour passer des paramètres par valeur, la procédure travaille sur une copie et ne peut pas modifier la variable d'origine.

ByRef

ByRef est utilisé pour passer des paramètres par référence, la procédure travaille directement sur la variable d'origine et peut la modifier.

Objets

Un objet est une entité qui possède des propriétés et des méthodes.

Une entité peut être une feuille de calcul, un classeur, une cellule. Les propriétés sont les caractéristiques d'un objet et les méthodes représentent les actions qu'il peut effectuer.

Pour accéder à une propriété ou une méthode :

Objet.Propriété Range("A1").Value

Pour définir une propriété et une méthode

Objet.Propriété ou objet.methode = valeur

Collections

Une collection est un objet qui contient d'autres objets du même type, utilisé pour organiser le contenu et gérer plusieurs objets similaires.

Workbooks : Collection de tous les classeurs ouverts.

Sheets : Collection de toutes les feuilles de calcul dans un classeur Worksheets.

Charts : Collection des graphiques dans un classeur.

Pour accéder aux éléments d'une collection :

Par index : Collection(index) 'l'index commence à 1

Par nom : Collection("objet")

Exemple :

'Accéder à la première feuille de calcul

Sheets(1).Activate

'Accéder à la feuille1

Sheets("feuille1").Activate

Les fonctions

Les fonctions sont essentielles pour automatiser des tâches dans Excel, elles permettent de manipuler des données, d'interagir avec des utilisateurs et d'effectuer des calculs. Pour définir une fonction en VBA, vous devez spécifier le type de données que la fonction va renvoyer. Le type de retour doit être déclaré, en utilisant le mot-clé As suivi du type de données.

ThisWorkbook

Représente le classeur Excel actuel dans lequel le code VBA est en cours d'exécution

ThisWorkbook.Save

Enregistre le classeur actuel

ThisWorkbook.Close

Ferme le classeur actuel

Worksheets

Représente la collection de toutes les feuilles de calcul du classeur

Worksheets.Item(index) ou Worksheets(index)

Permet d'accéder à une feuille de calcul spécifique par son index.

Exemple : Worksheets(1) Renvoie à la première feuille du classeur

Worksheets.Item("nom_de_la_feuille") ou Worksheets("Feuille2")

Renvoie à la feuille nommée feuille2.

Worksheet.index

Renvoie l'index (position) d'une feuille de calcul dans le classeur

Sheets.Add

Ajoute une nouvelle feuille de calcul au classeur

Vous pouvez spécifier la position de la nouvelle feuille et le type de feuille

Cells

Représente toutes les cellules d'une feuille de calcul. Vous pouvez y accéder en utilisant les numéros de ligne

Exemple Cells(1, 1) renvoie à la cellule A1 (ligne 1, colonne 1)

Cells.Find

Permet de rechercher une valeur spécifique dans une plage de cellules

Cells.ClearContents

Efface le contenu d'une cellule ou d'une plage de cellules

Cells.ClearFormats

Efface la mise en forme d'une cellule ou d'une plage de cellules

Range

Représente une ou plusieurs cellules d'une feuille de calcul. Vous pouvez définir des plages de cellules

Range("A1")

Fait référence à une seule cellule A1 dans cet exemple

Range(“A1:B5”)

Fait référence à une plage de cellules de A1 à B5 dans cet exemple

Range.Value

Permet de lire ou d’écrire la valeur d’une cellule ou d’une plage de cellules

Range.Formula

Permet de lire ou d’écrire une formule dans une cellule ou une plage de cellules

Exemple : Range(“A1”).Formula = “B1+C1” écrit la formule “=B1+C1” dans la cellule A1

Range.Interior

Permet de modifier la couleur de fond et d’autres propriétés de mise en forme.

Rows

Représentent les lignes d’une feuille de calcul

Vous pouvez les utiliser pour effectuer des opérations sur des lignes ou des colonnes entières

Rows(row_number)

Fait référence à une ligne spécifique

Exemple : Rows(1) fait référence à la première ligne

Columns

Représentent les colonnes d’une feuille de calcul

Columns(column_number) ou Columns(“column_letter”)

Fait référence à une colonne spécifique

Exemples : Columns(1) fait référence à la colonne A, Columns(“B”) fait référence à la colonne B

ActiveSheet

Représente la feuille de calcul active

Selection

Représente la sélection actuelle dans la feuille de calcul

Les fonctions les plus utilisées

WorksheetFunction

Cette fonction permet d’accéder aux fonctions Excel directement depuis VBA, utilisé pour effectuer des calculs complexes.

Exemple :

```
Sub CalculerMoyenne()
  Dim moyenne As Double
  ' Calcule la moyenne des valeurs dans la plage A1:A10
  moyenne = Application.WorksheetFunction.Average(Range("A1:A10"))
  MsgBox "La moyenne est : " & moyenne
End Sub
```

Cells

Cette fonction permet de référencer des cellules dans une feuille de calcul, utilisée pour accéder à des cellules par leur numéro de ligne et de colonne.
Exemple :
Cells(1,1).Value = “Bonjour” ‘Insère Bonjour dans la cellule A1

Range

Cette fonction permet de travailler avec une plage de cellules, utilisée pour lire ou écrire des valeurs dans des cellules spécifiques.

Exemple :

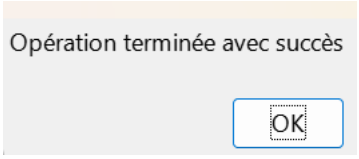
Range(“B1:B10”).Value = 100 ‘Remplit les cellules B1 à B10 avec la valeur 100

MsgBox

Cette fonction permet d’afficher une boîte de message à l’utilisateur, utilisée pour afficher des informations ou des alertes.

Exemple :

MsgBox “Opération terminée avec succès”



InputBox

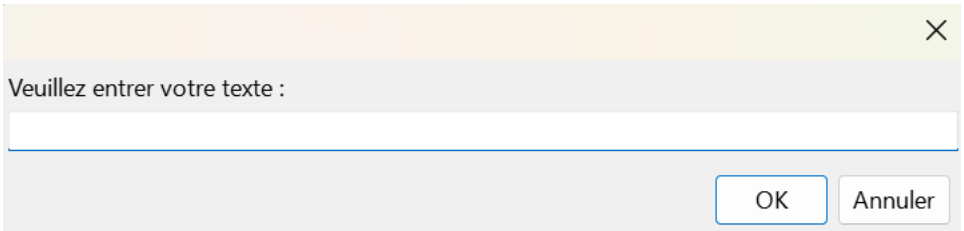
Cette fonction permet d’afficher une boîte de dialogue pour que l’utilisateur saisisse une valeur, utilisée pour obtenir des entrées utilisateurs.

Exemple :

Dim userInput As String ‘Déclaration d’une variable userInput de type chaîne de caractère

UserInput = InputBox(“Veuillez entrer votre texte : ”)

Exemple :



Date et heure

Cette fonction permet d'obtenir la date et l'heure, utilisée pour des opérations liées au temps.

Exemple :

Dim currentDate As Date 'Déclaration d'une variable currrentDate de type date

currentDate = Date 'Obtient la date actuelle

Les instructions conditionnelles

If...Then...Else

Cette fonction permet de créer une structure conditionnelle pour exécuter du code en fonction des conditions, utilisée pour la logique de programmation.

Exemple :

```
If Cells(1,1).Value > 10 Then 'Si la valeur de la cellule A1 est supérieur à 10
    MsgBox "La valeur est supérieure à 10" 'Affiche La valeur est supérieure à 10
Else 'Sinon
    MsgBox  "La valeur est 10 ou moins" 'Affiche La valeur est 10 ou moins
End If
```

Select *case* and select

Les instructions Select Case et Select sont des structures de contrôle qui permettent d'exécuter un groupe d'instructions en fonction de la valeur d'une expression.

For...Next

Cette fonction est utilisée pour créer une boucle qui répète une action un nombre de fois défini, utilisée pour traiter des séries de données.

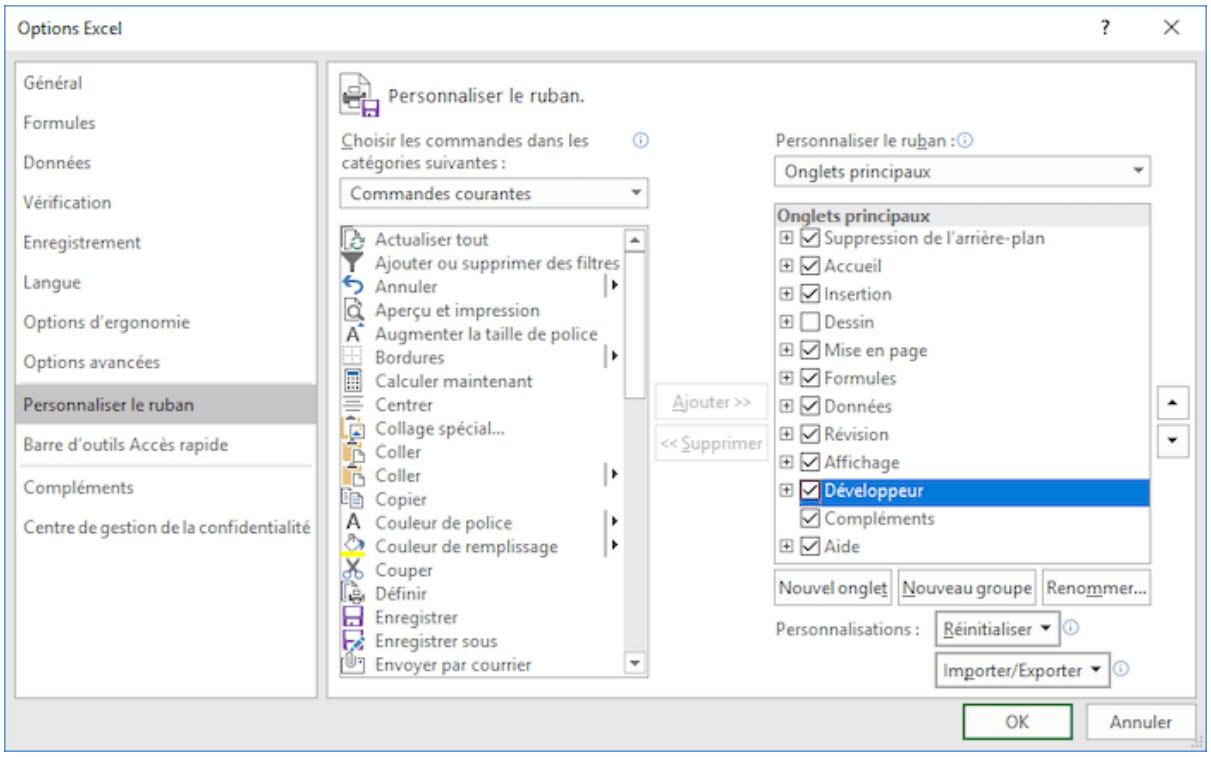
Exemple :

```
Dim i As Integer
For i = 1 To 10
   Cells(i,1).Value = i 'Remplit la colonne A avec les nombres de 1 à 10
Next i
```

	A
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9
10	10

Activer l'onglet développeur

1. Aller dans Fichier > Options ou cliquez sur ALT + F11
2. Cliquez sur Personnaliser le ruban
3. Cocher la case Développeur dans la liste de droite
4. Cliquez sur OK



Réaliser une macro

Une macro est un programme codé en VBA qui permet d'effectuer une série d'instructions au sein du logiciel.

Aller dans l'éditeur VBA en cliquant sur Visual Basic ou en appuyant sur ALT + F11.

Allez dans Insérer > Module.

Écrire Sub suivi du nom de la macro et appuyez sur Entrée.

Écrire les instructions entre Sub et End Sub. Vous pouvez par exemple, utiliser les fonctions et les objets VBA pour effectuer des tâches.

Exemple de macro :

```
Sub AfficherMessage()
  MsgBox “Voici la macro”
End Sub
```

Pour lancer la macro :

Cliquez sur macro dans l’onglet Développeur

Sélectionner le nom de la macro dans la liste

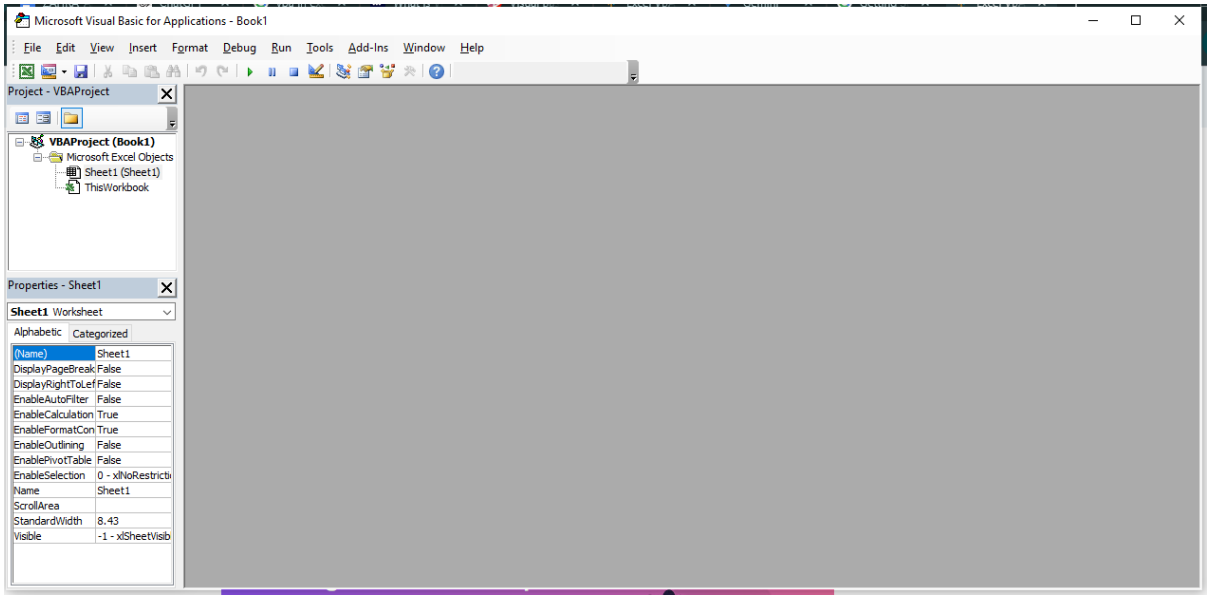
Cliquez sur Exécuter

Pour enregistrer le classeur :

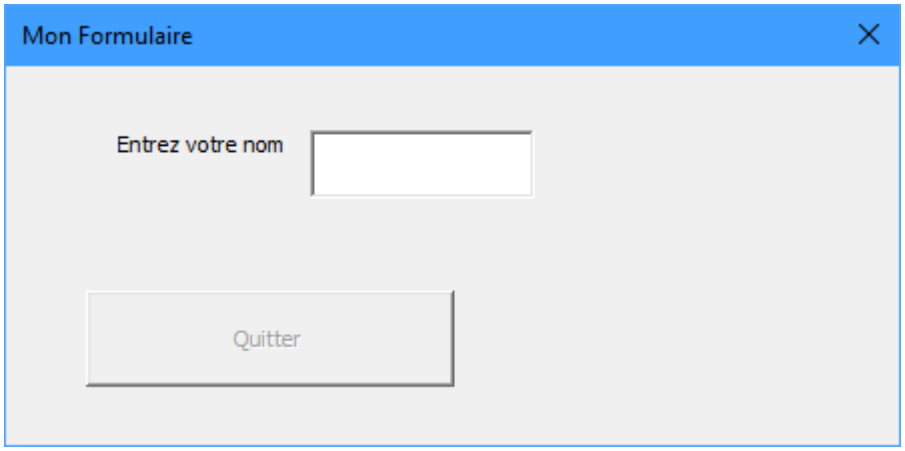
Aller dans Fichier puis enregistrer sous (format .xlsm)

Dans le menu déroulant Type, Sélectionner le Classeur Excel avec prise en charge du format .xlsm

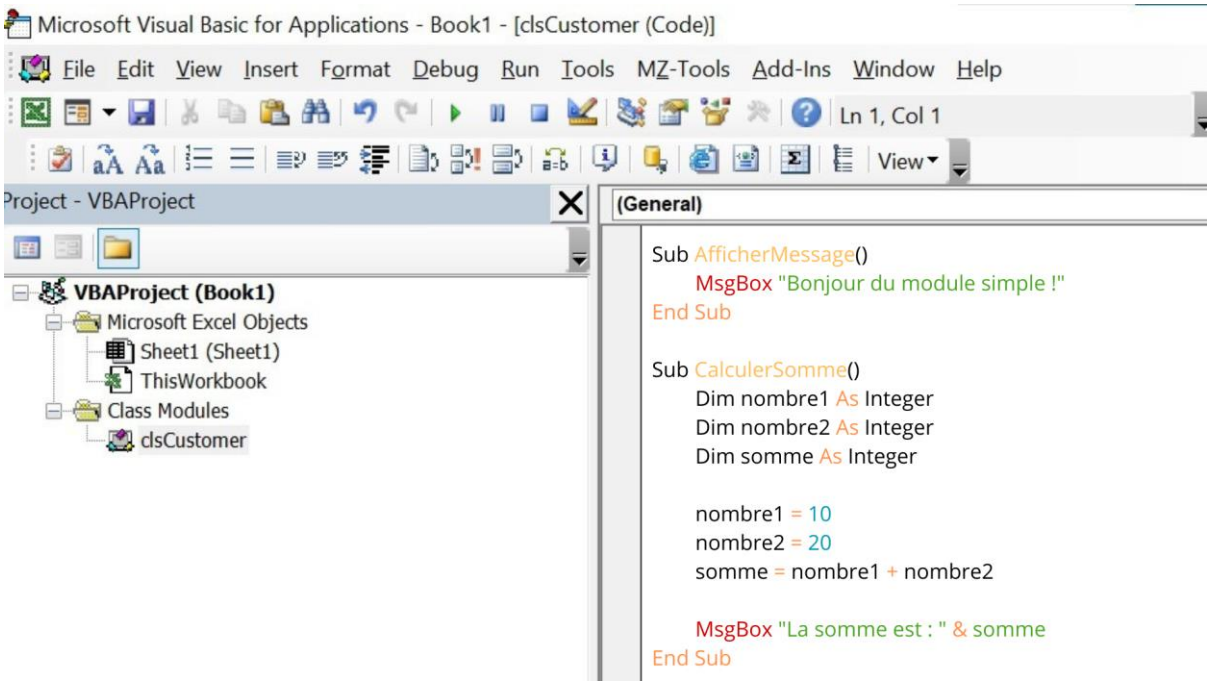
Environnement de développement VBA



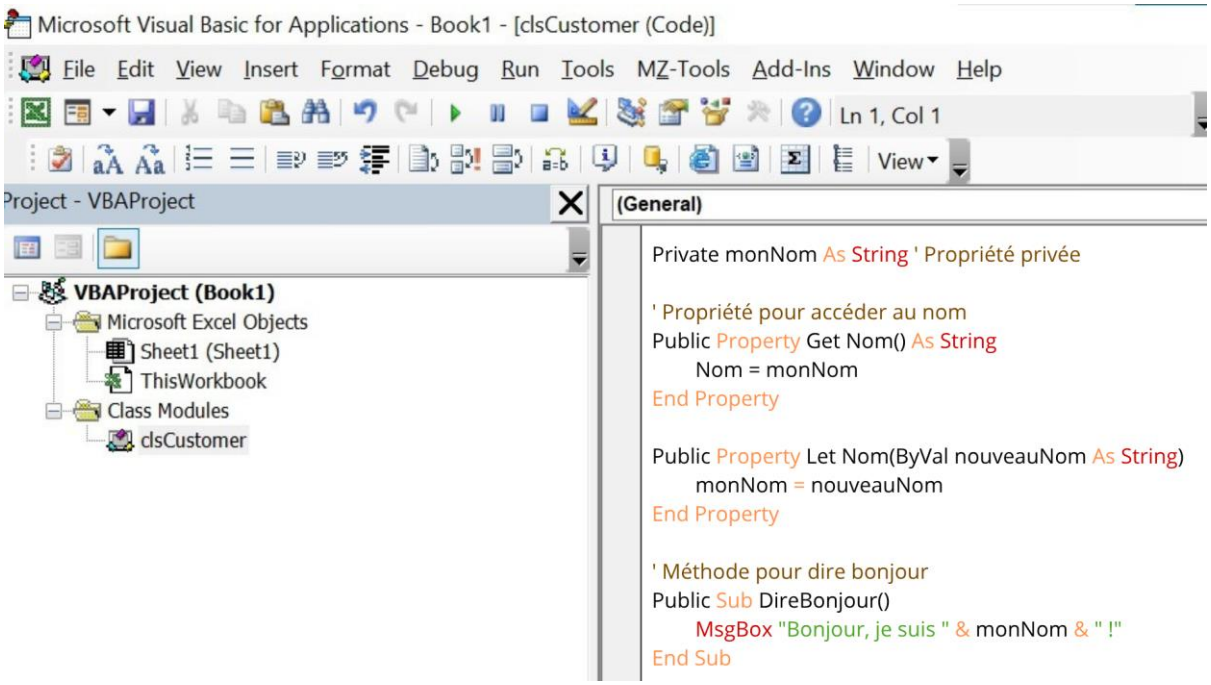
UserForms



Module



Module de classe



Les opérateurs arithmétiques

Opérateurs	Description
+	Addition
-	Soustraction
*	Multiplication
/	Division
^	Exposant
\\	Division entière
Mod	Modulo

Gestion des erreurs

On Error GoTo

Permet de gérer l'erreur, si une erreur se produit alors le code va afficher un message, enregistrer une erreur.

On Error Resume Next

Permet de contourner l'erreur, si une erreur se produit alors il va passer à la ligne suivante

API Windows

L'API Windows, accessible via VBA Excel, permet d'étendre les fonctionnalités d'Excel en interagissant directement avec le système d'exploitation Windows. Cela se fait en utilisant des fonctions déclarées à partir de bibliothèques DLL (comme user32.dll et kernel32.dll). Des fonctions courantes incluent l'affichage de boîtes de messages (MessageBox), la récupération d'informations système (GetSystemMetrics) et la manipulation de fenêtres (SetWindowPos).