~

Syntaxe pour le VBA

~

Table des matières

Pour **implémenter un algorithme** dans votre ordinateur (c'est-à-dire « rentrer l'algorithme sous forme de programme dans l'ordinateur »), il faut traduire chacune des instructions dans le langage du logiciel.
Cette syntaxe est donc un dictionnaire qui permet la traduction du langage naturel (ou pseudo-code) en VBA.

Cette syntaxe contient les bases, vous avez un tutoriel plus complet sur : http://www.excel-pratique.com/

I / Qu'est-ce que le VBA ?

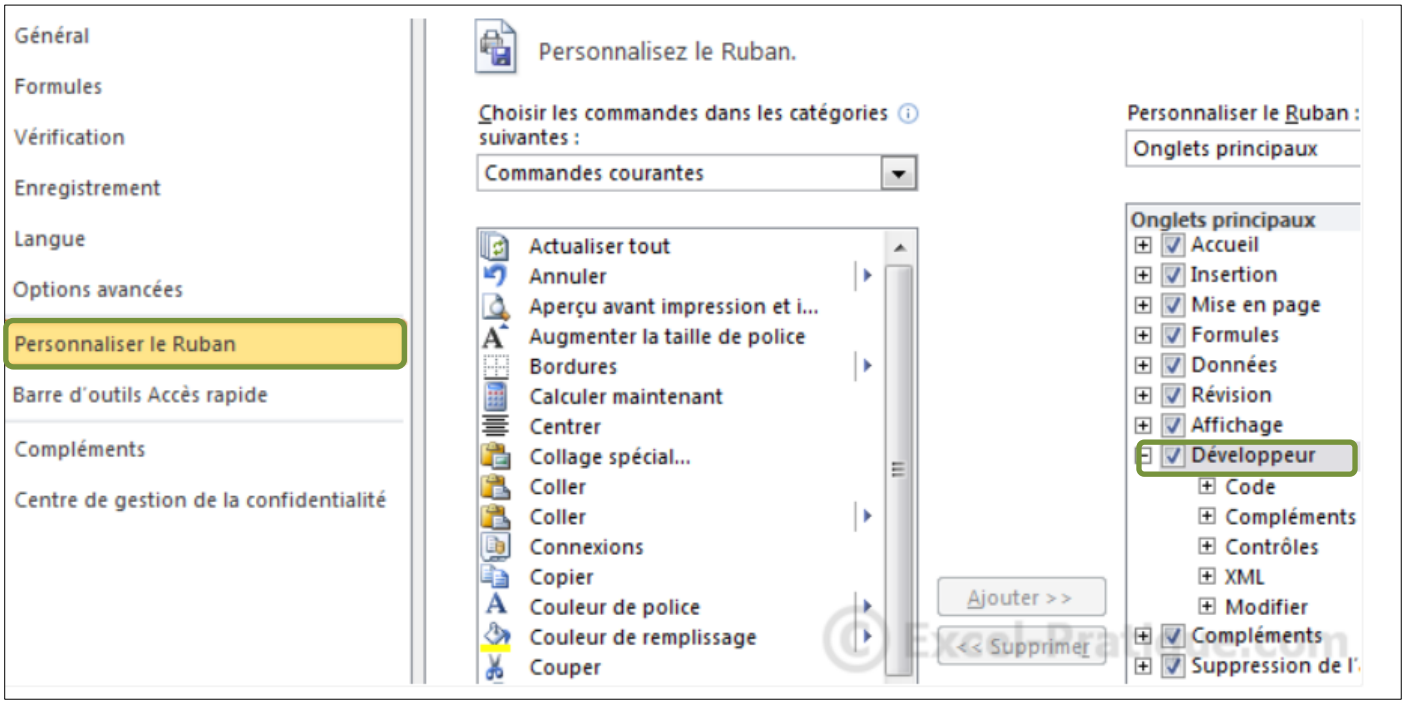
VBA : VISUAL BASIC for APPLICATION.
Le VBA est le langage de programmation pour les applications de Microsoft Office, en particulier celle d'Excel. Cela permet notamment d'automatiser des taches, de faire des calculs complexes, d'effectuer des instructions dans un ordre précis, de remplir (ou d'interroger) toutes les cellules de toutes les feuilles d'un fichier Excel, de saisir des informations à travers une boite de dialogue…

Attention, pour pouvoir enregistrer un fichier Excel contenant une macro, il faut l'enregistrer sous format *.xlsm.

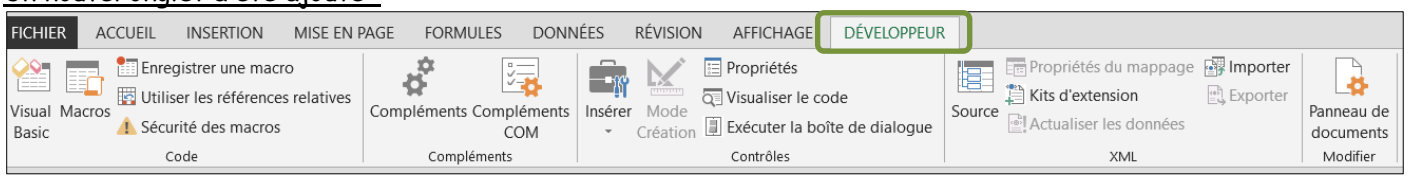
II / Personnaliser le ruban avec l'outil développeur

La première des choses à faire est de personnaliser le ruban avec l'outil développeur afin d'afficher les outils Excel qui nous seront utiles par la suite.

Cliquez Sur **FICHIER** / **Options** / **Personnaliser le ruban** et cochez **Développeur**.



Un nouvel onglet a été ajouté :

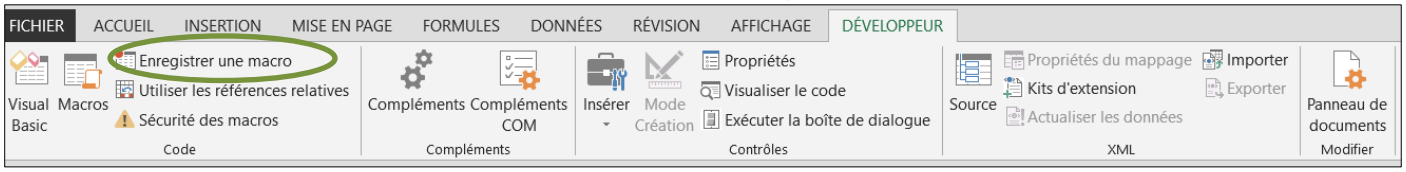


III / Enregistrer et paramétrer une macro :

Les macros sont enregistrées dans le fichier et ne sont pas reliées à une cellule en particulier.
Il existe deux manières d'enregistrer une macro : Le mode automatique et le mode normal.

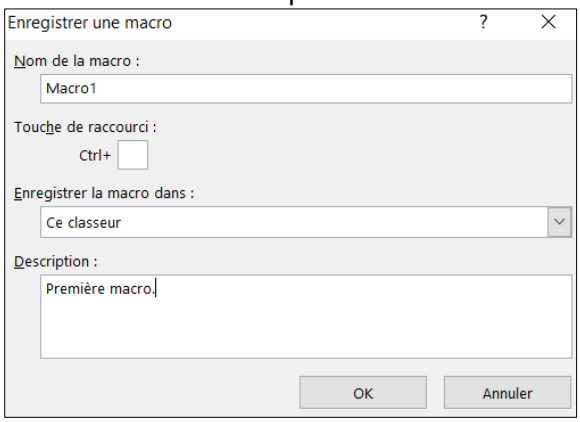
1. Le mode automatique :

Dans l'outil DEVELOPPEUR du Ruban, activer la commande **Enregistrer une macro**.



La fenêtre ci-dessous apparaît et vous permet de paramétrer la macro qui va être crée, c'est-à-dire :

- De préciser le nom de la macro ;
- De préciser une touche si nécessaire de la macro ;
- De préciser à quel classeur elle sera rattachée ;
- De faire une description du contenu de la macro.



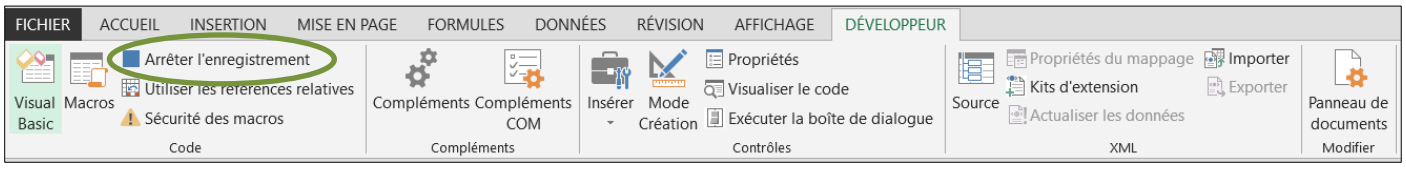
Le menu déroulant Enregistrer la macro dans vous propose trois options :

- Classeur de macros personnelles ;
- Ce classeur ;
- Nouveau classeur.

Si vous sélectionnez « **Classeur de macros personnelles** », la macro est enregistrée dans un classeur spécial appelé **PERSO.xlam**. Ce classeur sera ouvert automatiquement lors du lancement d'Excel et ses macros seront disponibles dans tous les classeurs ouverts.
Si vous sélectionnez « **Ce classeur** », la macro est enregistrée dans le classeur actif.
Si vous sélectionnez « **Nouveau classeur** », la macro est enregistrée dans un nouveau classeur.

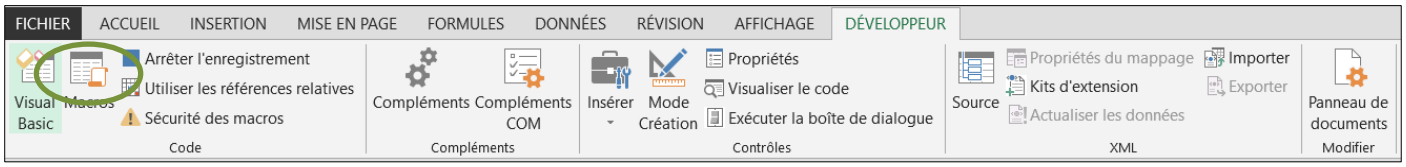
Remarque : Pour ce module, nous ferons les enregistrements dans le classeur actif (ce classeur).

Attention : Dès que vous cliquez sur « Enregistrer une macro », la macro commence à enregistrer automatiquement toutes les manipulations que vous ferez. Lorsque vous avez fini, il va falloir arrêter l'enregistrement.

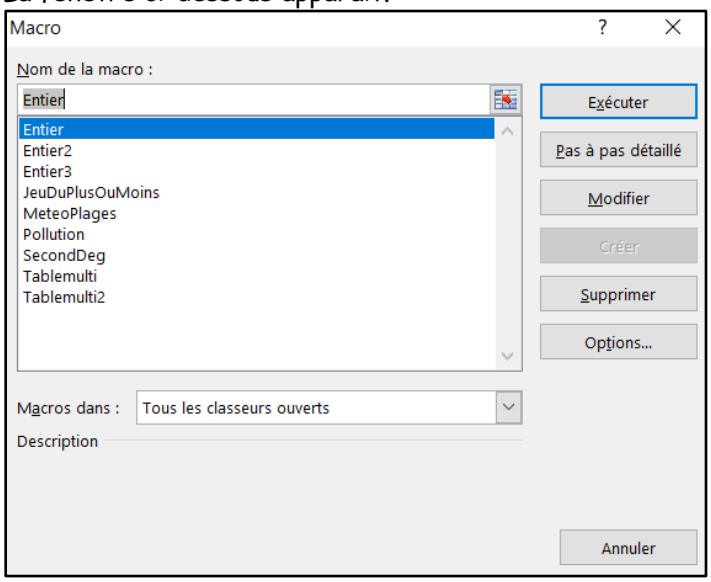


2. Le mode normal :

Dans l'outil DEVELOPPEUR du Ruban, activer la commande **Macro**.



La fenêtre ci-dessous apparait.



Elle permet de **créer** une nouvelle macro ou d'**exécuter**, de **modifier**, de **supprimer** une ancienne macro.

IV / Macro ou Programme en VBA

1. Le nom de la macro :

Exemple de macro dans VBA :

Sub prix() Dim TVA As Integer, PrixTTC As Single, PrixHT As Single TVA = InputBox("Quel est le montant de la TVA ?") PrixHT = InputBox("Quel est le prix HT ?") PrixTTC = PrixHT * (1 + TVA/100) MsgBox(PrixTTC) End Sub	La macro porte un **nom**. Dans l'exemple ci-contre le nom est : **prix.**

Les règles d'or pour le nommage des macros :

- Pas d'espaces dans les noms de macros ;
- Deux macros ne peuvent pas porter le même nom ;
- Une macro ne peut pas porter le même nom qu'une instruction VBA telle que ActiveCell, Range, Sub... ;
- Il est aussi très nettement déconseillé de donner des noms de macros tels que Sheet, Sheet1, Workbook, Excel, VBA, Classeur, ou tout nom qui pourrait prêter à confusion ;
- Pas d'accents ni de caractères spéciaux tel que *, ç, %, ?, etc.
- Le trait de soulignement est interdit au début d'une macro ;
- Une macro ne peut pas commencer par un chiffre, mais peut se terminer par un chiffre.

2. Une macro possède un début et une fin :

Sub prix() Dim TVA As Integer, PrixTTC As Single, PrixHT As Single TVA = InputBox("Quel est le montant de la TVA ?") PrixHT = InputBox("Quel est le prix HT ?") PrixTTC = PrixHT * (1 + TVA/100) MsgBox(PrixTTC) End Sub	La macro commence toujours par **Sub** suivi du nom de la macro et se termine par **End Sub**.

Remarque : Vous noterez qu'il y a une indentation (un espace) dans le corps de la macro. Cela n'est pas obligatoire mais permet de rendre la macro plus lisible.

3. Des commentaires :

Sub prix() ' Déclaration des variables Dim TVA As Integer, PrixTTC As Single, PrixHT As Single ' Entrée TVA = InputBox("Quel est le montant de la TVA ?") PrixHT = InputBox("Quel est le prix HT ?") ' Traitement ou calcul PrixTTC = PrixHT * (1 + TVA/100) 'Sortie MsgBox(PrixTTC) End Sub	Le texte en vert (précédé d'une apostrophe) est un commentaire. Il s'agit d'une aide à la compréhension de la macro. Il n'est pas pris en compte dans l'exécution de celle-ci.

4. Structure d'un algorithme ou d'un programme

Variables : Déclaration des variables.
Entrée : Saisi des données utiles pour le traitement de l'algorithme.
Initialisation des variables.
Traitement : Instructions faites par l'algorithme.
Sortie : Affichage des résultats.

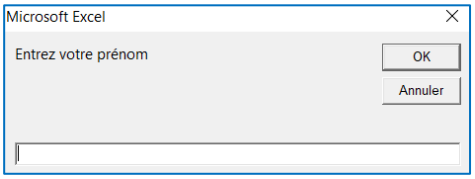
V / Communiquer avec l'utilisateur

1. InputBox

La fonction VBA InputBox affiche une boîte de dialogue invitant l'utilisateur à entrer du texte ou un nombre. Ce texte ou ce nombre pourra être mémorisé dans une variable.

Exemples :

```
• Sub Prenom()
        Prenom = InputBox ("Entrez votre prénom")
  End Sub
```



L'ordinateur demande à l'utilisateur d'entrer un prenom et enregistre la réponse dans la variable « Prenom ».

```
• Sub Prenom()
        InputBox ("Entrez votre prénom")
  End Sub
```

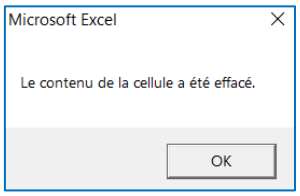
Attention : Le prénom ne sera enregistré dans aucune variable.

2. MsgBox

- La fonction VBA MsgBox affiche une information dans une boîte de dialogue. Cette information peut être un texte ou un nombre. Elle peut également afficher la valeur d'une variable ou un mélange d'un texte et d'une variable.
- Exemples :

Affichage uniquement d'un texte :

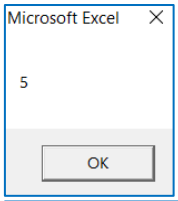
```
• Sub Prenom()
        MsgBox ("Le contenu de la cellule a été effacé. ")
  End Sub
```



Le texte doit être mis entre guillemé.

Affichage uniquement de la valeur d'une variable :

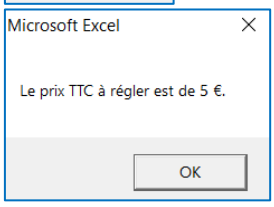
```
• Sub Prix()
        PrixTTC = 5
        MsgBox (PrixTTC)
  End Sub
```



Aucun guillemé ne doit être présent pour afficher une variable.

Affichage d'un texte et de la valeur d'une variable :

```
• Sub Prix()
        PrixTTC = 5
        MsgBox ("Le prix TTC à régler est de " & PrixTTC & " €.")
  End Sub
```

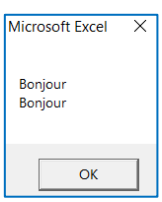


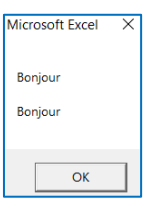
Entre le texte et la variable, il y a un &.

- Pour aller à la ligne ou sauter une ligne dans une boite de dialogue MsgBox, vous devez insérer le caractère correspondant au saut de ligne Chr(10) dans la fonction.

Exemples :

```
• Sub bonjour()
        MsgBox ("Bonjour" & Chr(10) & "Bonjour")
  End Sub
• Sub bonjour()
        MsgBox ("Bonjour" & Chr(10) & Chr(10) & "Bonjour")
  End Sub
```





VI / Sélection des cellules

Dans un programme VBA, il existe deux modes de repérages des cellules :

- le **mode absolu** ;
- le **mode relatif**.

Dans le **mode absolu**, que quel que soit la cellule active au moment où vous lancez votre macro, les cellules sélectionnées seront toujours les mêmes (celles écrites la macro).
Dans certains cas, il est utile que le résultat d'une macro dépende de ce qui a déjà été effectué auparavant et donc que le placement des actions de la macro soit déterminé de manière **relative** par rapport à cela. On veut, par exemple, qu'une écriture soit faite toujours dans la cellule sélectionnée (active) à ce moment-là.

Langage naturel	En langage VBA
Sélectionner la cellule A3	Range ("A3") .Select
Sélectionner les cellules A1 et B7	Range ("A1, B7") .Select
Sélectionner la plage des cellules de A1 à B7	Range ("A1:B7") .Select
Affecte « maplage » à la plage de cellules allant de A1 à B5.	Set maplage = Range("A1 : B5")
Sélectionne la plage de cellule renommée « maplage ».	Range("maplage") .Select
Active la feuille 2 et sélectionne la cellule A3.	Sheets("Feuil2") . Activate Range("A3") .Select
Sélectionne la cellule de la ligne 3 et de la colonne 2 :	Cells(3, 1) .Select
Sélectionner la cellule active actuelle.	ActiveCell.Select
Sélectionner la cellule décalée de 2 lignes vers le bas et de 1 colonne vers la droite par rapport à la cellule active actuelle.	ActiveCell.Offset(2, 1) .Select
Sélectionne les lignes allant de 2 à 6.	Range("2:6") .Select Rows("2:6") Select
Sélectionne les colonnes allant de B à G.	Range("B:G") .Select Columns("B:G") Select

VII / Attribuer une valeur ou un texte à une cellule

Langage naturel	En langage VBA
La cellule A3 prend la valeur 17.	Range("A3") .Value = 17
La cellule A3 affiche la texte « Rififi ».	Range("A3") .Value = "Rififi"
La cellule A3 de la feuille 2 prend la valeur 17.	Sheets("Feuil2") .Range("A3") .Value = 17
La cellule A3 de la feuille 2 du classeur 4 prend la valeur 17.	Workbooks("Classeur2.xlsx") Sheets("Feuil2") .Range("A3") .Value = 17
La cellule A3 prend la valeur de la cellule A1.	Range("A3") = Range("A1") Range("A3") .Value = Range("A1") .Value
La cellule A3 prend la valeur de la cellule A1 + 1.	Range("A3") = Range("A1") + 1

VIII / Modifier la mise en forme d'une cellule ou d'une plage

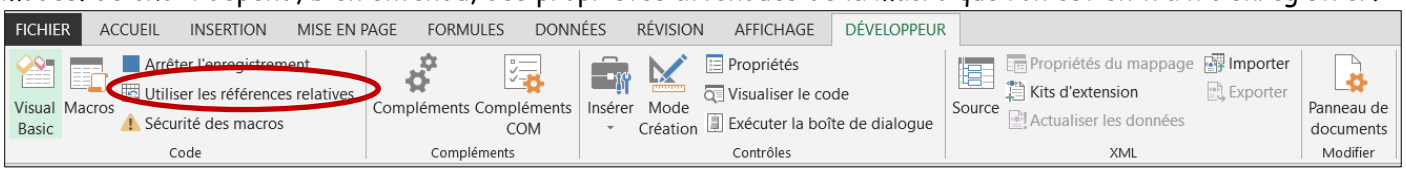
Langage naturel	En langage VBA
Ecrit « TOTO » dans la cellule active.	ActiveCell.FormulaR1C1 = "TOTO"
Modifie la taille du texte des cellules allant de A1 à A7.	Range("A1:A7") .Font.Size = 18
Modifie la police du texte des cellules allant de A1 à A7.	Range("A1:A7") .Font.Name = "Arial"
Modifie le texte en gras de la cellule A3. (Bold = Caractère gras)	Range("A3") .Font.Bold = True
Supprime la mise en forme « gras » de la cellule A3.	Range("A3") .Font.Bold = False
Met ou enlève la mise en forme « italique » de la cellule A3.	Range("A3") .Font.Italic = True Range("A3") .Font.Italic = False
Souligne le texte de la cellule A3.	Range("A3") .Font.Underline = True Range("A3") .Font.Underline = False
Ajoute ou enlève les bordures des cellules allant de A1 à A7.	Range("A1:A7") .Borders.Value = 1 Range("A1:A7") .Borders.Value = 0
Masque ou rend visible une feuille de calcul.	Sheets("Feuille3").Visible = 0 Sheets("Feuille3").Visible = -1
Recopie la mise en forme « taille » de la cellule A1.	Range("A3") .Font.Size = Range("A1") .Font.Size
Supprimer le contenu texte d'une feuille Excel	Cells.ClearContents

IX / Le mode absolu ou le mode relatif

Il existe deux modes de repérages des cellules dans un programme VBA :

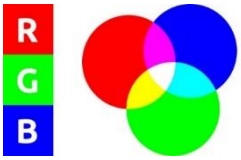
- le **mode absolu** ;
- le **mode relatif**.

Dans le **mode absolu**, que quel que soit la cellule active au moment où vous lancez votre macro, les textes seront toujours écris dans les mêmes cellules B2 et E2 (positionnement absolu).
Par opposition, dans certains cas, il est utile que le résultat d'une macro dépende de la cellule active et donc que le placement des actions de la macro soit déterminé de manière **relative** par rapport à cela. On veut, par exemple, qu'une écriture soit faite toujours dans la cellule sélectionnée (active) à ce moment-là.
Lorsque vous êtes en enregistrement automatique, le bouton ci-dessus permet de faire un choix entre ces deux modes. Ce choix dépend, bien entendu, des propriétés attendues de la macro que l'on est en train d'enregistrer.



X / Les différentes couleurs

La couleur d'une cellule est construite selon trois canaux : Rouge, Vert et Bleu (RVB). Pour obtenir une couleur particulière, nous devons ajuster le mélange de chacune d'entre elles. Les intensités de ses trois couleurs peuvent varier de 0 à 255.



Exemple : RGB (150 ; 014 ; 050) : La couleur demandée correspond à une intensité du rouge de 150, une intensité du vert de 14, une intensité du bleu de 50.

Langage naturel	En langage VBA
Colorier l'intérieur de la cellule active en rouge	Selection.Interior.Color = RGB(255, 0, 0)
Colorier l'intérieur de la cellule A1 en rouge	Range("A1").Interior.Color = RGB(255, 0, 0)
Colorier l'e texte de la cellule active en blanc	Selection.Font.Color = vbWhite
Colorier l'e texte de la cellule A1 en blanc	Range("A1").Font.Color = vbWhite
Effacer le contenu de la feuille (y compris la couleur)	Cells.Clear
Les couleurs	
RGB (rouge, verte, bleue)	RGB(r, g, b)
Rouge	vbRed ou RGB(255, 0, 0)
Verte	vbGreen ou RGB(0, 255, 0)
Bleue	vbBlue ou RGB(0, 0, 255)
Bleu turquoise	vbCyan
Violet	vbMagenta
Jaune	vbYellow
Noire	vbBlack ou RGB(0, 0, 0)
Blanche	vbWhite ou RGB(255, 255, 255)

XI / Déclarer ses variables, leur attribuer une valeur

Les variables permettent de stocker toutes sortes de données : lettres, mots, nombre, booléenne (VRAI, FAUX)… C'est un emplacement dans la mémoire.

La première chose à faire dans un algorithme en langage courant est de **déclarer (ou initialiser) ces variables**, c'est-à-dire de les définir comme réel, entier, texte… Cela se fait dans les premières lignes du programme.

Exemple de macro dans VBA :

```
Sub prix()
    Dim TVA As Integer
    Dim PrixTTC As Single
    Dim PrixHT As Single

    TVA = 20
    PrixHT = 130

    PrixTTC = PrixHT * (1 + TVA/100)
    MsgBox (PrixTTC)
End Sub
```

Dim : permet de déclarer une variable
TVA, PrixTTC, PrixHT : sont des noms de variable
As : Permet de déclarer le type de variable
Integer, Single : sont des types de variables.

L'**affectation** consiste à attribuer une donnée à une variable.

Les principaux types de variables :

- **Integer** est un **nombre entier** relatif (positif ou négatif) compris entre -32 768 et 32 767.
- **Single** est un **nombre décimal** (= flottant) compris entre $-3.402823*10^{38}$ et $3.402823*10^{38}$.
- **String** est un **texte**.
- **boolean** (booléen) ne peut prendre que deux valeurs **VRAI** ou **FAUX**.
- **date** est une **date**.
- **variant** est une variable qui peut contenir pratiquement tous les types de variables.

Dans VBA, déclarer les variables n'est pas obligatoire mais **recommandé.** Cela permet de se retrouver facilement et dans certains cas de résoudre des problèmes plus facilement.
Si vous ne spécifiez pas un type de données, le type de données **Variant** est affecté par défaut.

Attention : L'instruction Dim TVA, PrixTTC, PrixHT As Single permet de déclarer « PrixHT » comme réel mais ne permet pas la déclaration des variables TVA et PrixTTC. Ces dernières sont déclarées comme « variant ».

Remarque : Il est recommandé d'attribuer un nom explicite une variable (TVA ; PrixTTC ; PrixHT …)

L'**affectation** consiste à attribuer une donnée à une variable. (Exemple : TVA = 20 ou PrixHT = 130)

Les constantes :

Par opposition aux variables, existent les constantes. Dès leur déclaration, une valeur est leur est attribuée et celle-ci ne pourra pas changer tout au long de l'exécution du programme.

Exemples :
```
Const TVA = 19.6
Const PI = 3.14
```

XII / Quelques calculs particuliers

Langage naturel	En langage VBA
Lorsque vous utilisez un nombre aléatoire dans une macro, il faudra mettre la fonction randomize avant	randomize
Nombre aléatoire entier entre 0 et 3 Nombre aléatoire entier entre 1 et 4	Int(4 * Rnd) Int(4 * Rnd) + 1
Racine carrée de x ou $\sqrt{x}$	Math.Sqr(x)
Arrondir la variable A à 2 chiffres après la virgule	Round(A,2)

XIII / Insérer un bouton

Sous l'onglet DEVELOPPEUR, dans le groupe contrôles, cliquez sur Insérer, puis sous Contrôles de formulaire, cliquez sur **bouton**. Cliquez sur l'emplacement de feuille de calcul où vous souhaitez le coin supérieur gauche du **bouton** apparaisse.

XIV / Les différents operateurs et mots-clés

Langage naturel	En langage VBA
Est égal à	=
Est différent de	<>
Est plus petit que	<
Est plus petit ou égal	<=
Est plus grand que	>
Est plus grand ou égal	>=

Langage naturel	En langage VBA
Et (les deux conditions doivent être vraies)	AND
ou (Au moins une des deux conditions doit être vraie)	OR
Ou exclusif (Une des deux conditions seulement doit être vraie).	NOR
Faux la condition doit être fausse	NOT

XV / Structure conditionnelle

Les conditions sont très utiles en programmation, elles nous serviront à effectuer des actions en fonction de critères précis (même principe que la fonction SI).

- La principale instruction est **If**. Elle se termine obligatoirement par **End If**.
  Si la condition est remplie alors on exécute l'instruction.
  Exemple de macro :

```
Sub Bulletin()
    Dim Note As Single, Commentaire As String
    Note = InputBox (" Quelle est ta note ?")
    If Note >= 18 Then
        Commentaire = " Excellent résulat !"
    End If
    MsgBox(Commentaire)
End Sub
```

  Remarque : Mettre des indentations (après If, Else if et Else afin de rendre plus lisible la macro.

- L'instruction facultative Sinon : **Else**.
  Exemple de macro :

```
Sub Bulletin()
    Dim Note As Single, Commentaire As String
    Note = InputBox (" Quelle est ta note ?")
    If Note >= 18 Then
        Commentaire = " Excellent résulat !"
    Else
        Commentaire = " Sans commentaire."
    End If
    MsgBox(Commentaire)
End Sub
```

- **ElseIf** permet d'ajouter plusieurs conditions à la suite :
  Exemple de macro :

```
Sub Bulletin()
    Dim Note As Single, Commentaire As String
    Note = InputBox (" Quelle est ta note ?")
    If Note >= 18 Then
        Commentaire = " Excellent résulat !"
    ElseIf Note >= 12 Then
        Commentaire = " Bon résultat."
    Else
        Commentaire = " Sans commentaire."
    End If
    MsgBox(Commentaire)
End Sub
```

XVI / Les Boucles

Une boucle permet de répéter plusieurs fois des instructions sans avoir à les réécrire.

1. La boucle POUR

Lorsque le nombre de traitement est connu à l'avance, on utilise une boucle POUR.
On utilise un compteur souvent initialisé à 1 (ou 0) et qui **s'incrémente** (augmente) automatiquement de 1 à chaque itération jusqu'à n : à chaque répétition de boucle, la variable i est automatiquement augmentée de 1.

Exemple :

• Sub TropPoli() For I = 1 to 5 MsgBox ("Bonjour") Next End Sub	L'ordinateur affichera 5 fois le message « Bonjour ».	I va prendre 5 les valeurs de 1 à 5 et exécuter les instructions (Ici, c'est afficher le message « Bonjour ».

2. La boucle Tant QUE (While)

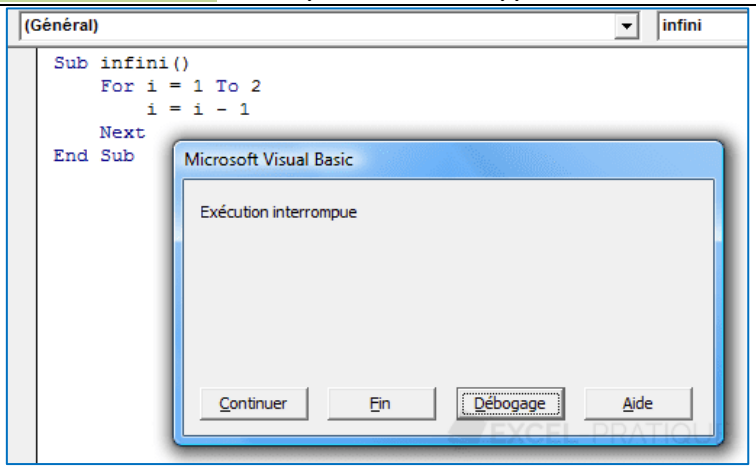
Lorsque le nombre de traitement n'est pas connu à l'avance mais dépend d'une condition, on utilise une boucle TANT QUE (While). Tant que la condition est vraie, la boucle est exécutée.
Attention à ne pas faire une boucle infinie.

Exemple :

• Sub Seuil() N = 0 While N < 3 MsgBox (N) N = N + 1 Next End Sub	L'ordinateur affichera 0, 1, 2 puis arrêtera le programme.	Le programme sera exécuté tant que la condition (ici N < 3) est rempli. Il s'arretera après.

3. Sortir d'une boucle infinie

Il peut arriver qu'on lance une macro bien trop lente ou pire encore, une boucle infinie ...
Heureusement, avec Ctrl + Pause (Break) il est possible de stopper une macro en cours d'exécution !



XVII / Macros : des procédures et des fonctions

- Un « sub procédure » est une macro en VBA qui effectue une série d'instruction mais de ne retourne rien. Elle peut afficher certains nombres ou texte demandé(s) par l'utilisateur.
  C'est ce que vous avez utilisé jusqu'à présent.
- Une « function » est une macro en VBA qui effectue une série d'instruction et qui retourne un résultat (nombre ou valeur).

1. Utilisation d'une macro dans une autre macro :

Exemple de macros :

```
• Sub avertissement()
    MsgBox ("Attention !!!")
  End Sub
```

```
• Sub exemple()
    If Range("A1") = "" Then
      avertissement
    End If
  End Sub
```

Pour exécuter une macro à partir d'une autre macro, entrez simplement son nom.
Ici si la macro exemple est lancée et que la cellule A1 est vierge alors la macro avertissement est lancée.

Remarque : Par défaut, les variables ne sont pas accessibles depuis les autres macros.

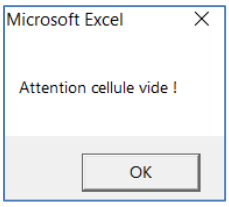
2. Les arguments :

Les arguments permettent de transmettre des valeurs d'une procédure à une autre.

Exemple de macros :

```
• Sub Avertissement(texte As string)
    MsgBox ("Attention " & texte & " !")
  End Sub
```

Ajout d'un argument « texte » dans la procédure Avertissement.
Pour exécuter la procédure « Avertissement » il faudra rentrer un argument de type string.




```
• Sub exemple()
    If Range("A1") = "" Then
      Avertissement("cellule vide")
    End If
  End Sub
```

Si la macro exemple est lancée et que la cellule A1 est vierge alors la macro avertissement est lancée.

Remarque : En cas d'arguments multiples, ceux-ci doivent être séparés par des virgules.

3. Les fonctions :

Une fonction est une portion de programme effectuant une tâche ou un calcul relativement indépendant du reste du programme. Elle doit **renvoyer un résultat** (souvent une valeur) au programme qui l'appelle. La principale différence entre Sub (procédure) et Fonction est qu'une fonction retourne une valeur.

Pour créer une fonction :

```
Function carre(nombre)
    carre = nombre ^ 2 'La fonction "carre" renvoie la valeur de "carre"
End Function
```