

COURS DEVELOPPEMENT



HODABALO Lidaou

Ingénieur Infra SAP Basis

PLAN DU COURS

Table des matières

I.	Introduction.....	3
I.1	Révision Excel : Cas pratique : Gestion simple d'un petit magasin	3
I.2	Démarrage.....	5
I.3	Première macro	7
II.	Les feuilles et cellules.....	11
II.1	Les sélections	11
II.2	Les propriétés.....	16
II.3	Les couleurs.....	23
III.	Les variables	28
III.1	Les variables (Partie 1).....	28
III.2	Les variables (Partie 2).....	34
IV.	Les conditions	39
IV. 1	Les conditions (Partie 1)	39
IV.2	Les conditions (Partie 2)	46
V.	Les boucles	48
V.1	Boucles	48
V.2	Boucles (exercice)	52
VI.	Les procédures et fonctions	56
VI.1	Lancer une procédure depuis une procédure.....	56

I. Introduction

I.1 Révision Excel : Cas pratique : Gestion simple d'un petit magasin

Contexte

Tu es responsable d'un petit magasin. Tu dois créer un tableau pour suivre :

- les produits vendus
- leur prix unitaire
- la quantité vendue
- le chiffre d'affaires généré

Objectif :

Construire un fichier Excel qui calcule automatiquement le chiffre d'affaires par produit et le chiffre d'affaires total.

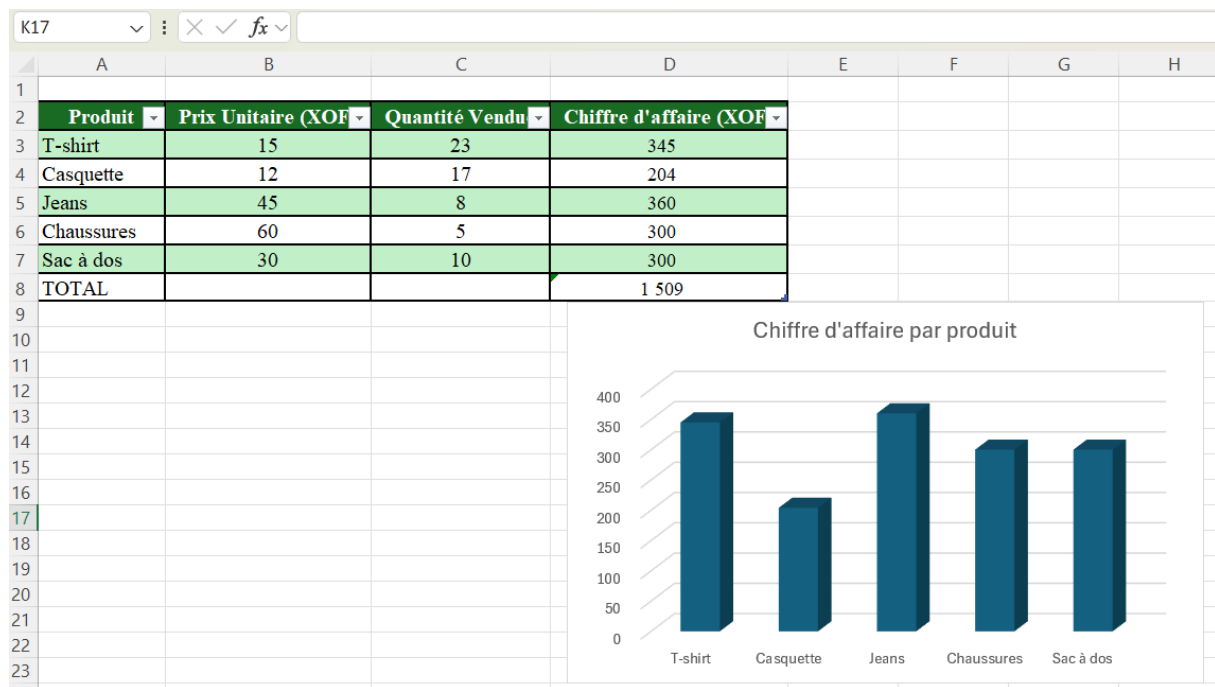
Données de départ :

Produit	Prix Unitaire (XOF)	Quantité Vendue
T-shirt	15	23
Casquette	12	17
Jeans	45	8
Chaussures	60	5
Sac à dos	30	10

Étapes à réaliser :

1. **Créer un tableau propre avec ces données (soigne les bordures, alignements, etc.).**
2. **Ajouter une colonne "Chiffre d'Affaires" qui calcule pour chaque produit :**
 - Formule = Prix Unitaire × Quantité Vendue
3. **Mettre en forme le tableau :**
 - Format monétaire pour les colonnes Prix et Chiffre d'Affaires (€).
 - Style de tableau (bande alternée, en-tête gras).
4. **Ajouter une cellule qui calcule le Chiffre d'Affaires total :**
 - Utiliser une formule de somme automatique.
5. **Appliquer un filtre pour pouvoir trier les produits par quantité ou par chiffre d'affaires.**
6. **(Optionnel bonus) Ajouter un graphique simple (ex: histogramme) pour visualiser les chiffres d'affaires par produit.**

Résultat attendu :



TD N°1 : Cas Pratique : Gestion de Ventes

Vous travaillez pour une boutique en ligne qui vend différents produits. On vous donne un tableau partiellement rempli à télécharger sur le lien : <https://we.tl/t-JGEpRxMvme> (le téléchargement doit être fait au **plus tard le 30/04/2025 à 24H00** au risque que le fichier ne soit plus disponible), et vous devez :

1. Compléter la colonne "Total Avant Remise"
2. Compléter la colonne "Total Après Remise"
3. Appliquer une mise en forme conditionnelle :
 - Si "Total Après Remise" > 500 €, surligner en vert.
 - Si "Total Après Remise" < 100 €, surligner en rouge.
4. Créer un filtre pour trier par catégorie.
5. Ajouter un tableau croisé dynamique : objectif : obtenir le total des ventes par catégorie.

Q1.

À partir du tableau donné, créez un tableau croisé dynamique pour afficher :

- Les catégories de produits en lignes.
- Le total des ventes (Total Après Remise) en valeurs.
- Question : Quelle catégorie a généré le plus de ventes ?

Q2.

Modifiez votre tableau croisé dynamique pour afficher :

- Les produits en lignes.
- Le nombre de commandes (compte des ID Commande) en valeurs.
- Question : Quel produit a été commandé le plus souvent ?

Q3.

Ajoutez un filtre au tableau croisé dynamique sur la date (par mois).

Question : En quel mois les ventes ont-elles été les plus élevées ?

Q4.

Ajoutez les remises (%) en colonnes du tableau croisé dynamique.

Question : Quelle remise a été la plus utilisée et dans quelle catégorie ?

Q5.

Créez un champ calculé dans votre tableau croisé dynamique qui calcule :

Marge estimée = Total Après Remise * 0.30

Question : Quelle est la marge totale pour la catégorie "Électronique" ?

NB :

Chaque étudiant enregistrera son classeur sous son Nom_Prenom.

En exemple, si l'étudiant s'appelle ATAKONA Pilambou, il enregistrera son classeur sous **ATAKONA_Pilambou**.

Les devoirs devront être déposés au plus tard **le mardi 06 mai 2025 à 24H00** dans le répertoire portant le Nom et Prénom de l'étudiant. Chaque étudiant recevra à cet effet un lien dans sa boîte mail (gmail) afin qu'il puisse déposer son devoir.

I.2 Démarrage

Le **VBA (Visual Basic for Applications)** est un langage proche du Visual Basic qui nécessite une application hôte pour s'exécuter (Excel dans notre cas).

Une **application hôte**, c'est un gros programme qui sert de "maison" pour faire tourner d'autres petits programmes à l'intérieur. C'est comme un centre commercial (**application hôte**) qui accueille plein de petites boutiques (**modules, plugins, applis**).

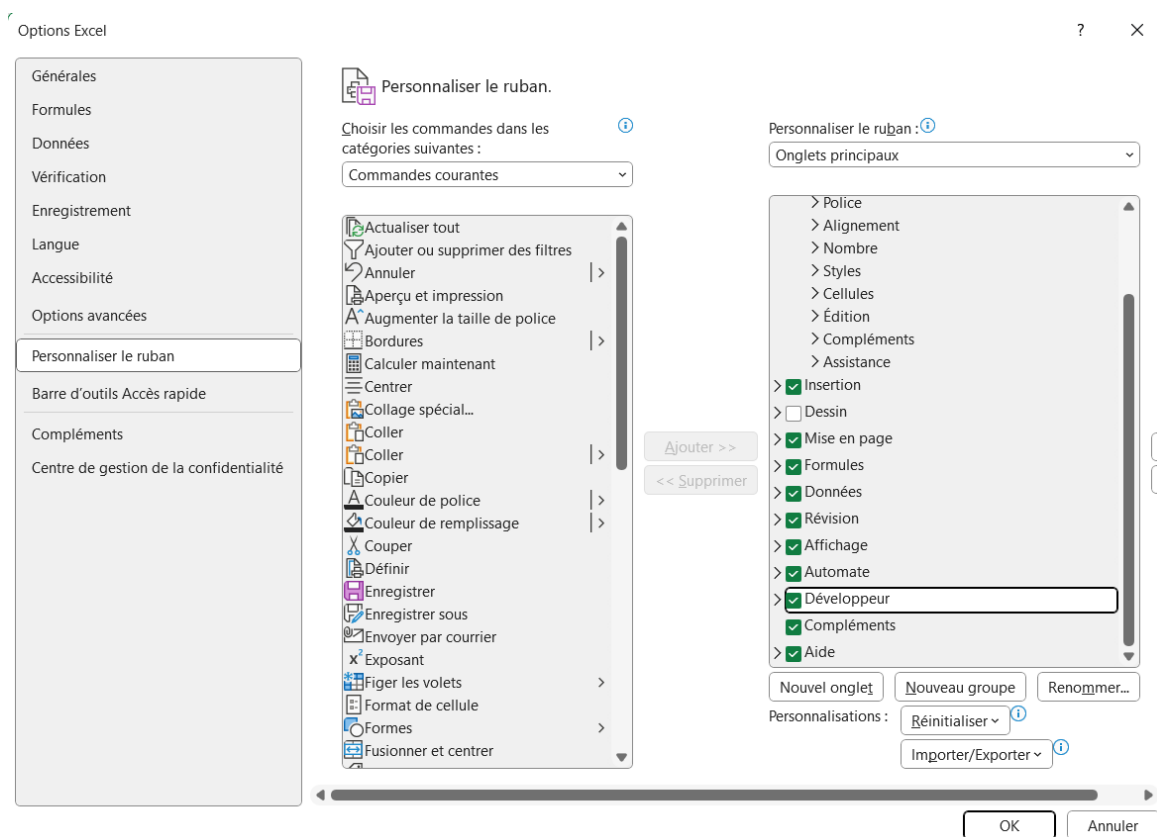
Le VBA est donc un langage de programmation intégré à Excel pour **automatiser des tâches, créer des applications personnalisées et interagir avec les feuilles de calcul**.

Grâce au VBA nous allons pouvoir réaliser à peu près tout ce que l'on souhaite avec Excel ...

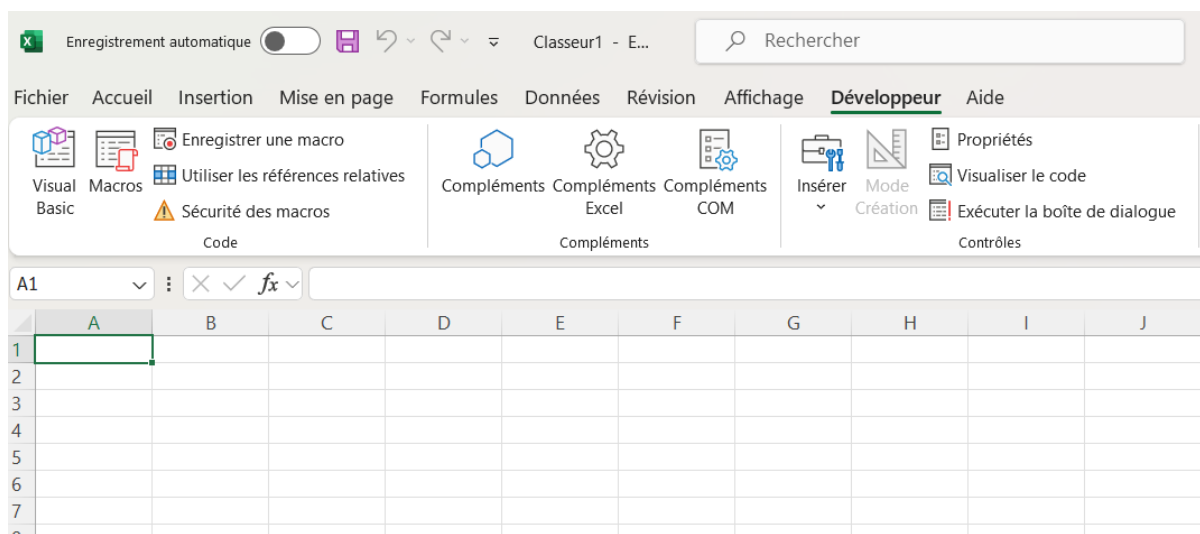
Mais avant de démarrer, commençons par afficher les outils qui nous seront utiles.

Démarrer Excel puis observer les onglets existants. On remarque qu'il n'existe pas l'onglet : **Développeur**.

Cliquez sur Fichier > Options > Personnaliser le Ruban puis cochez Développeur puis cliquer sur o.k

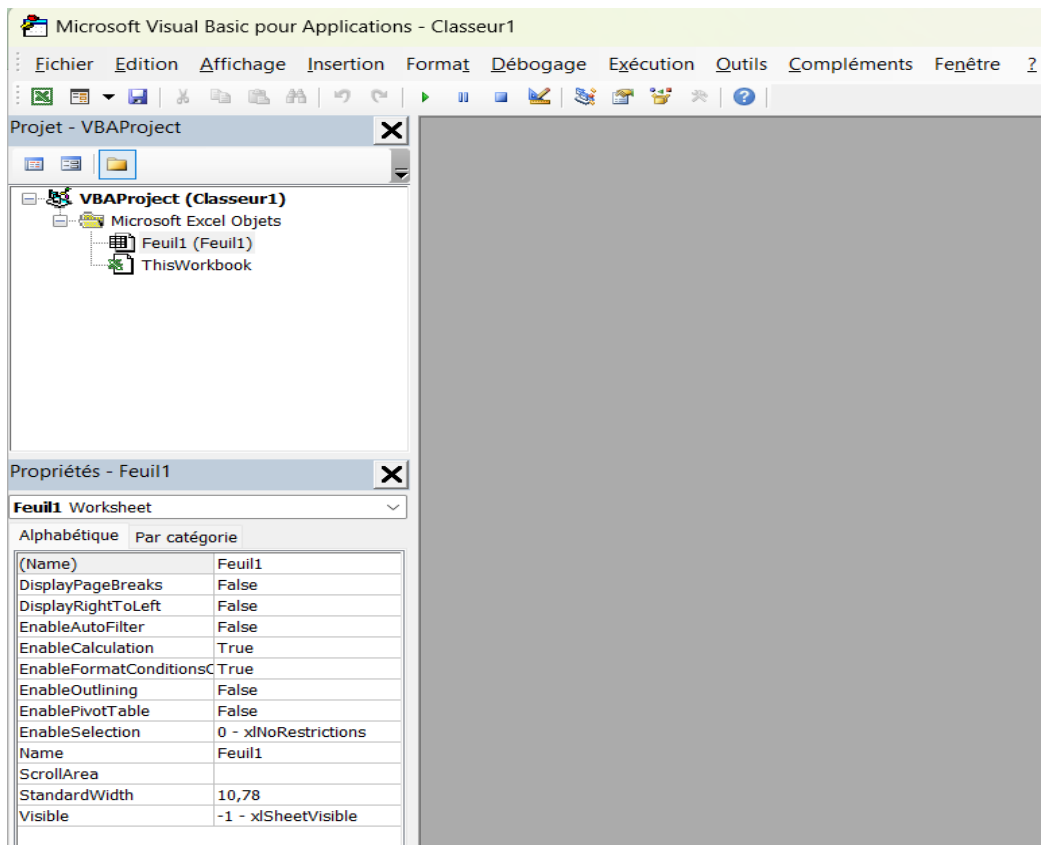


Vous constatez qu'un nouvel onglet appelé **Développeur** s'ajout. :



Pour travailler avec du code VBA, nous avons besoin d'un éditeur, celui-ci est déjà installé. Pour l'ouvrir :

- Lancer Excel
- Faites le raccourci **Alt + F11** ou en cliquant sur **Visual Basic** depuis l'onglet **Développeur** : on obtient le résultat ci-dessous :

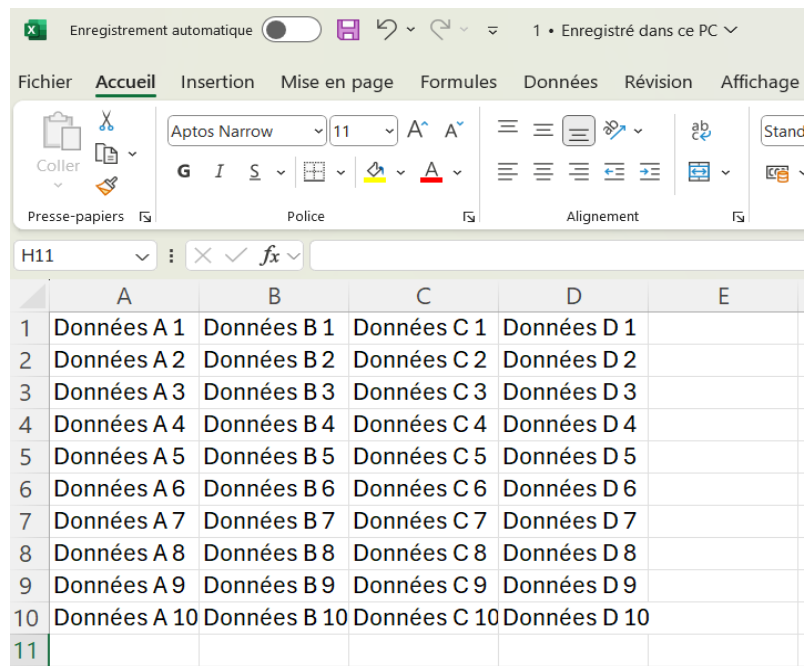


Nous y reviendrons, retenez simplement le raccourci **Alt + F11** pour le moment ...

I.3 Première macro

Il est possible d'automatiser certaines tâches en toute simplicité grâce à l'enregistreur de macros. Pour prendre un exemple simple,

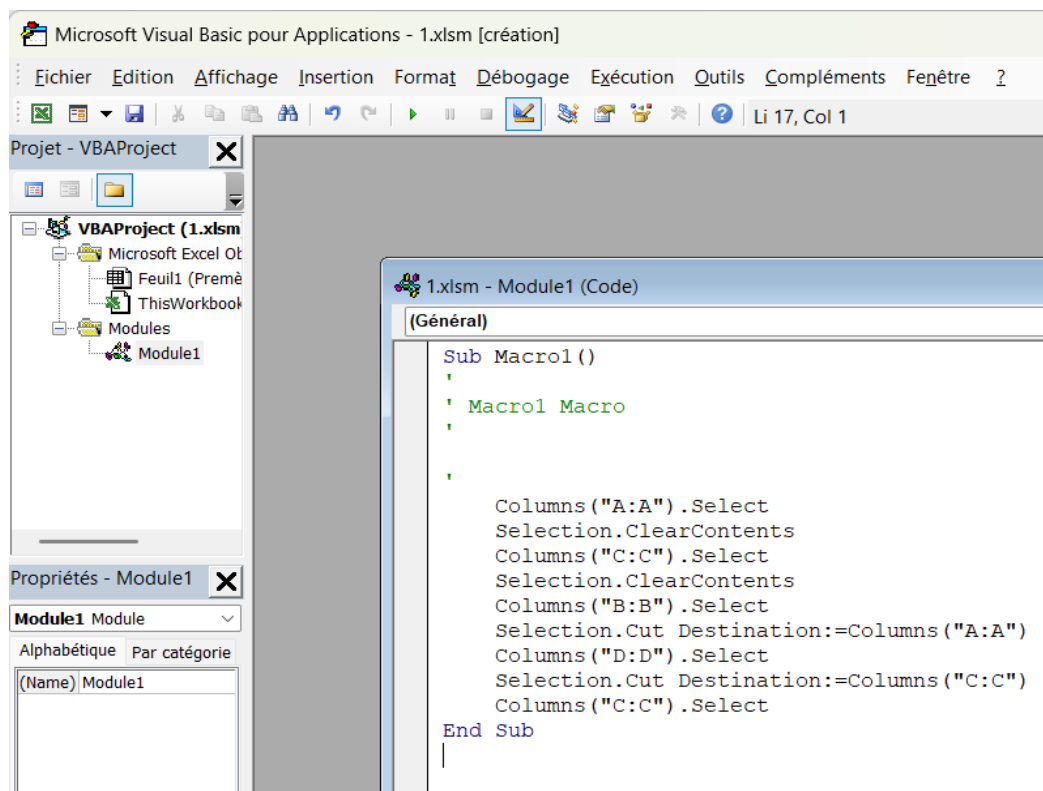
- Créer sur votre bureau un répertoire que vous nommerez Cours VBA Excel.
- Ouvrir un classeur Excel et l'enregistrer dans le dossier VBA Excel sous le nom 1 en choisissant le type **.xlsm** (Classeur Excel (prenant en charge les macros)).
- Renseigner Excel avec les données présente sur la capture ci-dessous
- Quelles opérations désirons-nous réaliser ? Nous souhaitons à l'aide d'une macro automatiser les opérations suivantes :
 - Effacer le contenu de la colonne A
 - Effacer le contenu de la colonne B
 - Déplacer le contenu de la colonne B dans la colonne A
 - Déplacer le contenu de la colonne D dans la colonne C



Pour ce faire, cliquez sur Enregistrer une macro puis sur Ok, exécutez les opérations décrites ci-dessus sans interruption (car toutes les manipulations sont enregistrées) et pour terminer cliquez sur Arrêter l'enregistrement.

Excel a enregistré vos manipulations et les a traduites en code VBA.

Pour voir votre macro, ouvrez l'éditeur (Alt + F11), **Dérouler Modules** puis double-cliquez sur **Module1** :



Le code ci-dessus correspond aux manipulations enregistrées.

Nous allons nous arrêter quelques instants sur le code généré :

```
Sub Macro1()  
'  
' Macro1 Macro  
'  
'  
  
Columns("A:A").Select  
Selection.ClearContents  
Columns("C:C").Select  
Selection.ClearContents  
Columns("B:B").Select  
Selection.Cut Destination:=Columns("A:A")  
Columns("D:D").Select  
Selection.Cut Destination:=Columns("C:C")  
Columns("C:C").Select  
End Sub
```

Sub et **End Sub** délimitent le début et la fin de la macro, **Macro1** correspond au nom de cette macro :

```
Sub Macro1()  
  
End Sub
```

Nous allons maintenant modifier le nom de cette macro et lui attribuer un nom qui soit un peu plus parlant. Pour cela, remplacez simplement Macro1 par **manipulationsDesColonnes** (le nom ne doit pas contenir d'espaces) :

```
Sub manipulationsDesColonnes()
```

Le texte en vert (texte précédé d'une apostrophe) est un commentaire, il n'est pas pris en compte à l'exécution du code :

```
'  
' Macro1 Macro  
'  
'
```

Les commentaires sont très utiles pour s'y retrouver lorsque l'on dispose de beaucoup de code ou pour ne pas exécuter certaines lignes de code sans pour autant les supprimer.

```

Sub manipulationsDesColonnes()

    'Mon premier commentaire !

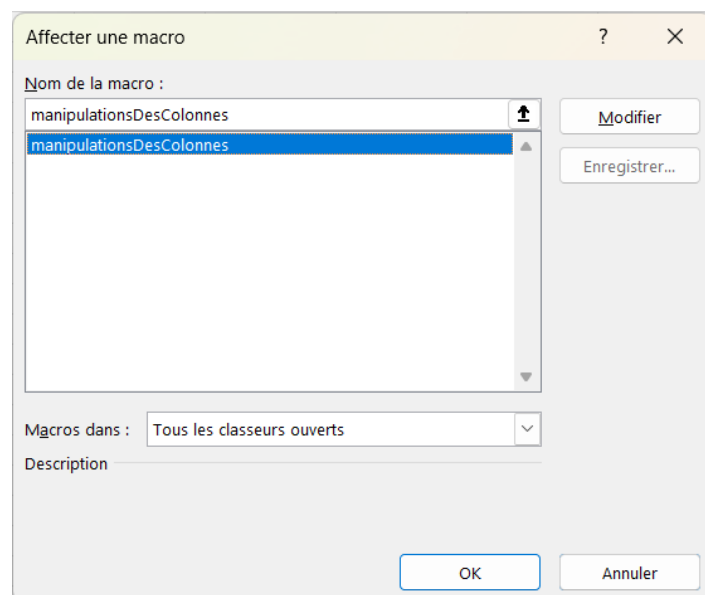
    Columns("A:A").Select
    Selection.ClearContents
    Columns("C:C").Select
    Selection.ClearContents
    Columns("B:B").Select
    Selection.Cut Destination:=Columns("A:A")
    Columns("D:D").Select
    Selection.Cut Destination:=Columns("C:C")
    Columns("C:C").Select
End Sub

```

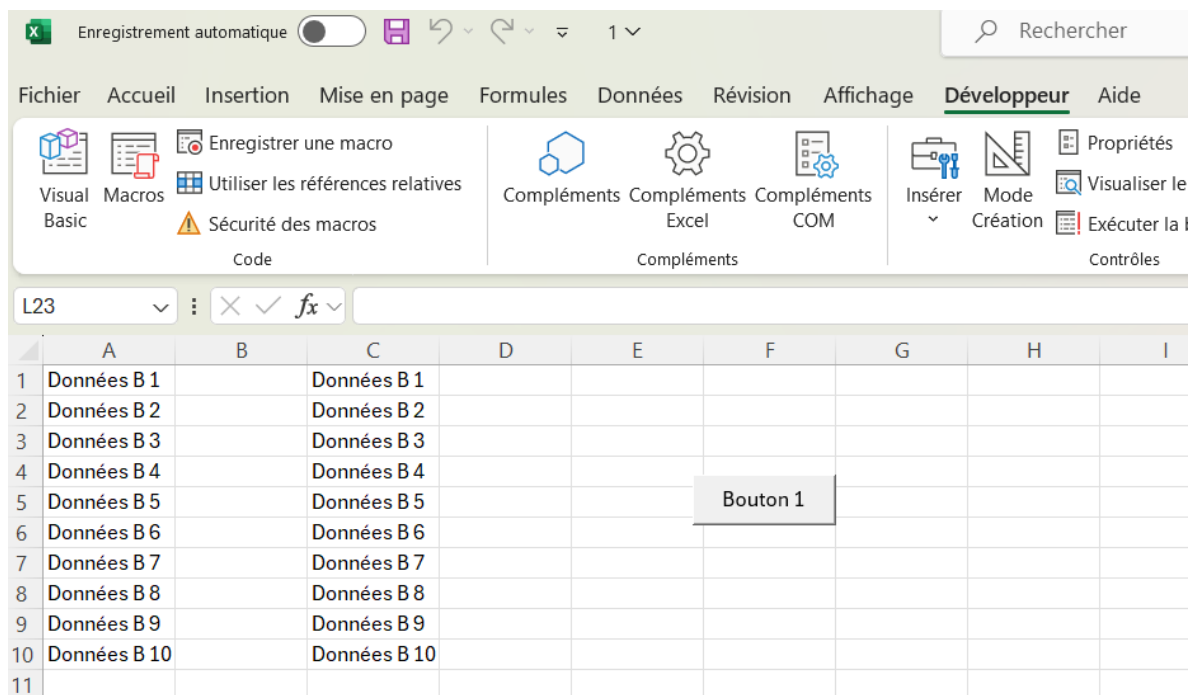
Nous souhaitons maintenant que cette macro s'exécute en cliquant sur un bouton.

Annuler les actions précédentes pour revenir à l'étape des 04 colonnes renseignées comme précédemment.

Pour insérer un bouton, aller dans *l'onglet Développeur* > *groupe Contrôles* > *Insérer* > cliquer sur *Bouton* (Contrôles de formulaires) > *Tracer le bouton* comme ci-dessous > *sélectionnez ensuite simplement votre macro* > *cliquer sur Ok* :



Lorsque vous cliquerez sur le bouton, la macro sera exécutée :

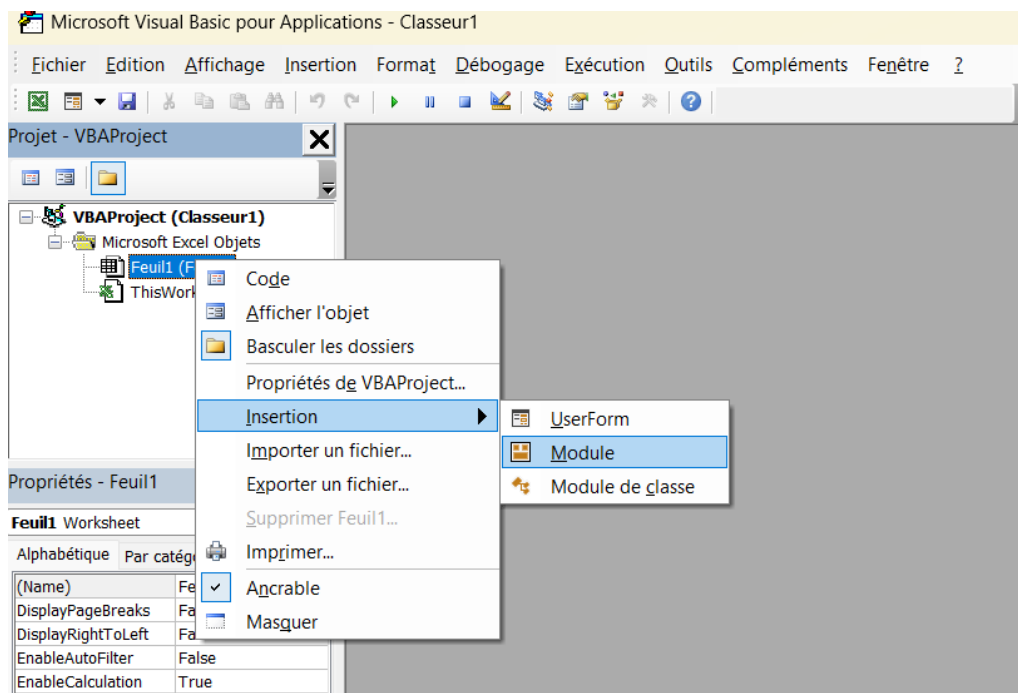


II. Les feuilles et cellules

II.1 Les sélections

Nous allons créer une macro qui sélectionnera une cellule de notre choix.

Ouvrez l'éditeur et ajoutez-y un module :

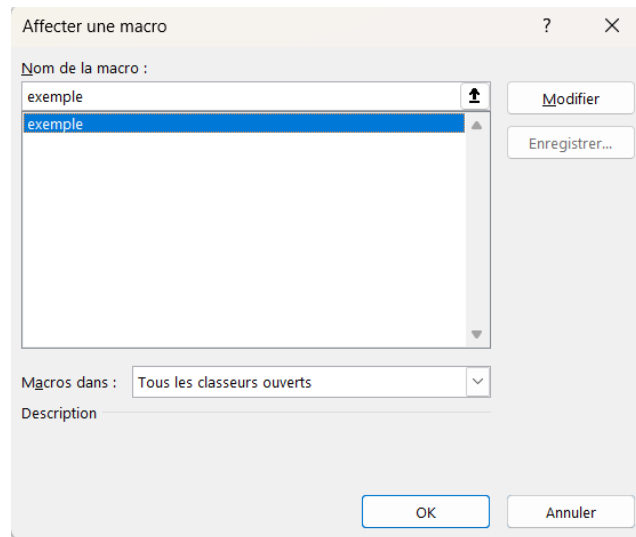


Dans le module, tapez `sub exemple` et appuyez sur Entrée.

Vous remarquerez qu'Excel a automatiquement ajouté la fin de cette nouvelle procédure :

```
Sub exemple()  
  
End Sub
```

Créez maintenant un bouton de formulaire auquel vous allez associer cette macro (vide pour le moment) :



Nous allons compléter notre macro avec ceci :

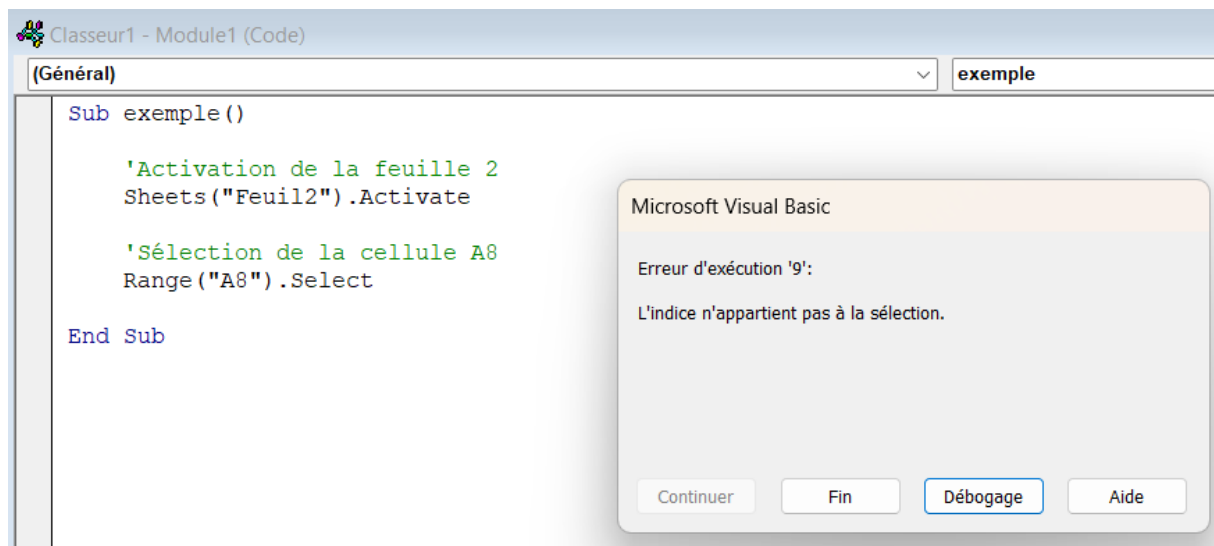
```
Sub exemple()  
  
    'Sélection de la cellule A8  
    Range("A8").Select  
  
End Sub
```

Testons cette macro en cliquant sur notre bouton de formulaire, la cellule A8 est alors sélectionnée.

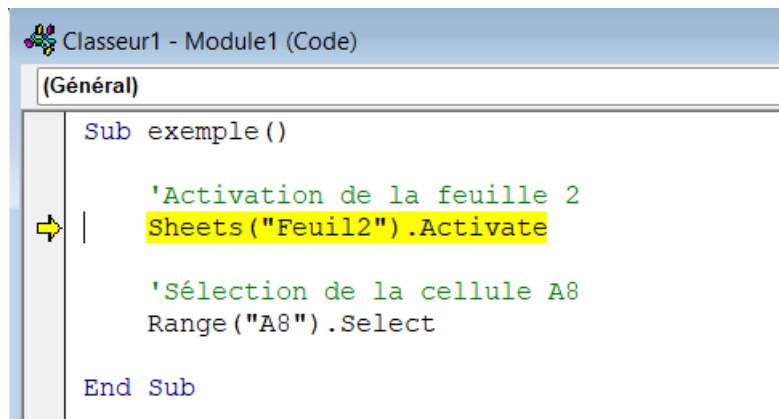
Nous allons maintenant modifier cette macro pour sélectionner la cellule A8 de la seconde feuille :

```
Sub exemple()  
  
    'Activation de la feuille 2  
    Sheets("Feuil2").Activate  
  
    'Sélection de la cellule A8  
    Range("A8").Select  
  
End Sub
```

Si Excel nous renvoie l'erreur ci-dessous, c'est que la feuille 2 n'existe pas dans notre classeur.



En cliquant sur **Débogage**, on obtient un surlignement au niveau de la syntaxe de l'activation de la Feuil2 :



Fermer l'éditeur pour arrêter le débogage puis relancer l'éditeur. Pour apporter la correction nous allons créer la Feuil2 et retester la macro.

Excel active alors la feuille 2 avant de sélectionner la cellule A8.

Il est à rappeler que nous pouvons nous aider des commentaires (texte en vert) pour bien comprendre les macros de ce cours.

SÉLECTION D'UNE PLAGE DE CELLULES

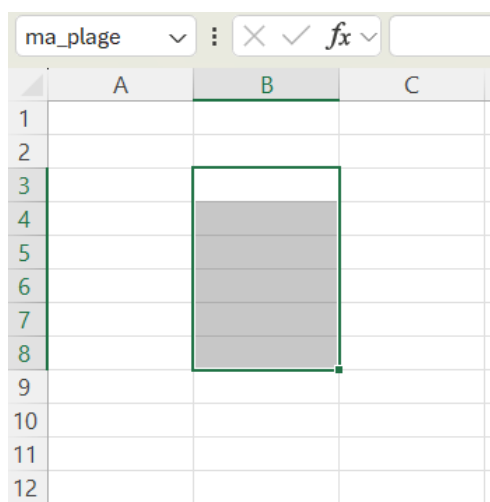
```
Sub exemple()  
  
    'Sélection des cellules A1 à A8  
    Range("A1:A8").Select  
  
End Sub
```

SÉLECTION DE CELLULES DISTINCTES

```
Sub exemple()  
  
    'Sélection des cellule A8 et C5  
    Range("A8, C5").Select  
  
End Sub
```

SÉLECTION D'UNE PLAGE DE CELLULES NOMMÉE

```
Sub exemple()  
  
    'Sélection des cellules de la plage "ma_plage"  
    Range("ma_plage").Select  
  
End Sub
```



	A	B	C
1			
2			
3			
4			
5			
6			
7			
8			
9			
10			
11			
12			

SÉLECTION D'UNE CELLULE EN FONCTION D'UN NUMÉRO DE LIGNE ET DE COLONNE :

```
Sub exemple()  
  
    'Sélection de la cellule de la ligne 8 et de la colonne 1  
    Cells(8, 1).Select  
  
End Sub
```

Cette autre manière de sélectionner permet des sélections plus dynamiques et sera bien utile par la suite.

En voici un petit exemple :

```
Sub exemple()  
  
    'Sélection aléatoire d'une cellule de la ligne 1 à 10 et de la colonne 1  
    Cells(Int(Rnd * 10) + 1, 1).Select  
  
    'Traduction :  
    'Cells([nombre_aléatoire_entre_1_et_10], 1).Select  
  
End Sub
```

Ici, le numéro de ligne est `Int(Rnd * 10) + 1`, autrement dit *un nombre entre 1 et 10* (inutile de retenir ce code pour le moment).

SÉLECTION DE LIGNES

Il est possible de sélectionner des lignes entières avec `Range` ou `Rows` (Rows étant spécifique aux lignes) :

```
Sub exemple()  
  
    'Sélection des lignes 2 à 6  
    Range("2:6").Select  
  
End Sub
```

```
Sub exemple()  
  
    'Sélection des lignes 2 à 6  
    Rows("2:6").Select  
  
End Sub
```

SÉLECTION DE COLONNES

Tout comme pour les lignes, il est possible de sélectionner des colonnes entières avec *Range* ou *Columns* (Columns étant spécifique aux colonnes) :

```
Sub exemple()  
  
    'Sélection des colonnes B à G  
    Range("B:G").Select  
  
End Sub
```

```
Sub exemple()  
  
    'Sélection des colonnes B à G  
    Columns("B:G").Select  
  
End Sub
```

II.2 Les propriétés

AUTOMATISATION DE LA SAISIE DU TEXTE

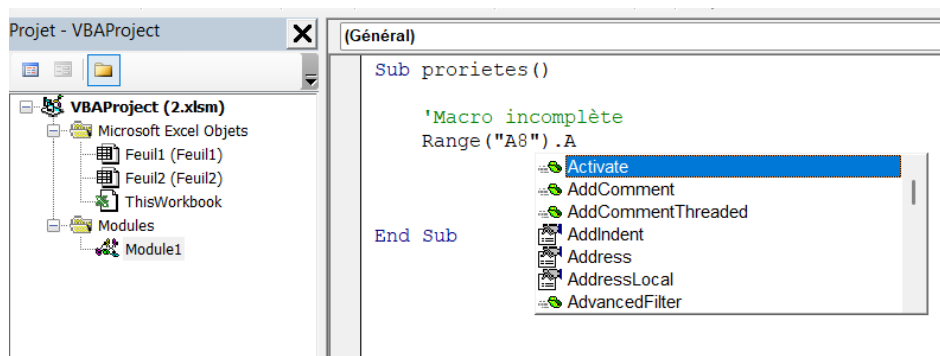
Nous allons maintenant agir sur le contenu et l'apparence des cellules et des feuilles.

Lancer Excel > l'enregistrer dans COURS BVA EXCEL sous le nom *2.xlsm* > *Ouvrir l'éditeur* > *Ajouter un module* > Saisir la *macro incomplète* ci-dessous : > *Retourner dans Excel puis ajouter un bouton* (Contrôle de formulaire) > Associer la macro portant le nom *propriétés* au bouton de formulaire :

```
Sub proprietes()  
  
    'Macro incomplète  
    Range("A8")  
  
End Sub
```

Nous voulons ensuite effectuer une action sur la cellule A8 avec le début de macro précédent.

Pour afficher la liste des possibilités que l'on peut associer à l'objet Range, ajoutez un `.` après *Range("A8")* :



L'éditeur affiche alors les différentes possibilités ...

Pour ce premier exemple, cliquez sur *Value* puis appuyez sur la touche Tab pour valider ce choix :

```
Sub proprietes()

    'Macro incomplète
    Range("A8").Value

End Sub
```

La propriété *Value* représente ici le contenu de la cellule.

Nous voulons maintenant donner la valeur 48 à la cellule A8 :

```
Sub proprietes()

    'Cellule A8 = 48
    Range("A8").Value = 48

    'Traduction :
    'La valeur de la cellule A8 est désormais : 48

End Sub
```

Puis, la valeur *Exemple de texte* à A8 (le texte doit être mis entre " ") :

```
Sub proprietes()

    'Cellule A8 = Exemple de texte
    Range("A8").Value = "Exemple de texte"

End Sub
```

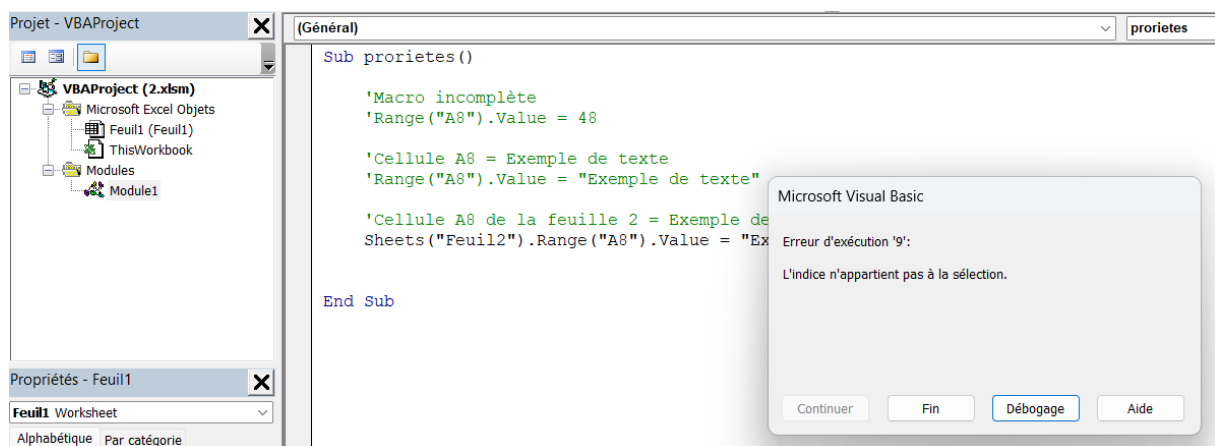
Dans ce cas, c'est bien la cellule A8 de la feuille où est lancée la procédure (ici, celle où se trouve le bouton formulaire) qui sera modifiée. Il s'agit en effet de la cellule A8 de la feuille actuelle nommée par défaut *"Feuil1"* dans Excel.

Si vous créez un second bouton sur la feuille 2, ce sera alors la cellule A8 de la feuille 2 qui sera modifiée.

Pour modifier la cellule A8 de la feuille 2 en cliquant sur le bouton de la feuille 1, il faut préciser le nom de la feuille en ajoutant `Sheets("Nom_de_la_feuille")` avant Range :

```
Sub proprietes()  
  
    'Cellule A8 de la feuille 2 = Exemple de texte  
    Sheets("Feuil2").Range("A8").Value = "Exemple de texte"  
  
End Sub
```

En exécutant le bouton, si vous obtenez l'erreur ci-dessous, c'est parce que la feuille "Feuil2" n'a pas été insérée dans le classeur. Le problème est alors résolu par l'insertion de cette feuille. En cliquant sur "Feuil2", on y trouve bien le texte `Exemple de texte` dans la cellule A8.



Il est important de cliquer sur "Feuil2" avant de voir le contenu de la cellule A8, puisque nous n'avons pas la macro automatiser l'ouverture de la "Feuil2" comme vue dans la partie sélection.

De même, si l'on souhaite modifier la cellule A8 de la feuille 2 d'un autre classeur ouvert, il faut préciser le nom du classeur en début de ligne à l'aide `de Workbooks("Nom_du_fichier")` :

Pour cela, ouvrir un nouveau classeur et l'enregistrer dans le même répertoire que le classeur 2.xlsm COURS BVA EXCEL sous le nom Classeur2.xlsx. Dans l'éditeur, saisir les lignes de macro ci-dessous :

```
Sub proprietes()  
  
    'Cellule A8 de la feuille 2 du classeur 2 = Exemple de texte  
    Workbooks("Classeur2.xlsx").Sheets("Feuil2").Range("A8").Value = "Exemple de texte"  
  
End Sub
```

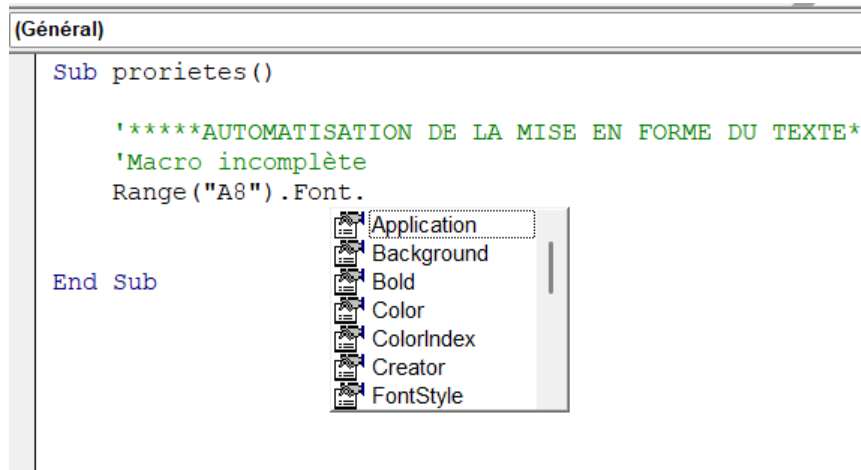
Bien que `Value` ait été utilisé pour illustrer ces différents exemples, il n'est pas nécessaire de l'indiquer, car c'est automatiquement la valeur de la cellule qui est modifiée si rien n'est précisé.

Ainsi, ces 2 lignes génèrent un résultat identique :

```
Range("A8").Value = 48
Range("A8") = 48
```

MISE EN FORME DU TEXTE

Après avoir sélectionné la propriété **Font** et ajouté un **.**, la liste des propriétés que l'on peut attribuer à la mise en forme du texte apparaît comme suit :



MISE EN FORME : TAILLE DU TEXTE

```
Sub proprietaes()  
  
    'Modifier la taille du texte des cellules A1 à A8  
    Range("A1:A8").Font.Size = 18  
  
End Sub
```

MISE EN FORME : TEXTE EN GRAS

```
Sub proprietaes()  
  
    'Mettre en gras les cellules A1 à A8  
    Range("A1:A8").Font.Bold = True  
  
End Sub
```

Bold = True signifie *Caractères en gras = Oui*.

Pour retirer la mise en forme **Bold** à un texte, il faut donc remplacer **Oui** par **Non**, autrement dit, **True** par **False** :

```

Sub proprietes()

    'Enlever la mise en forme "gras" des cellules A1 à A8
    Range("A1:A8").Font.Bold = False

End Sub

```

MISE EN FORME : TEXTE EN ITALIQUE

```

Sub proprietes()

    'Mettre en italique les cellules A1 à A8
    Range("A1:A8").Font.Italic = True

End Sub

```

MISE EN FORME : TEXTE SOULIGNÉ

```

Sub proprietes()

    'Souligner les cellules A1 à A8
    Range("A1:A8").Font.Underline = True

End Sub

```

MISE EN FORME : POLICE

```

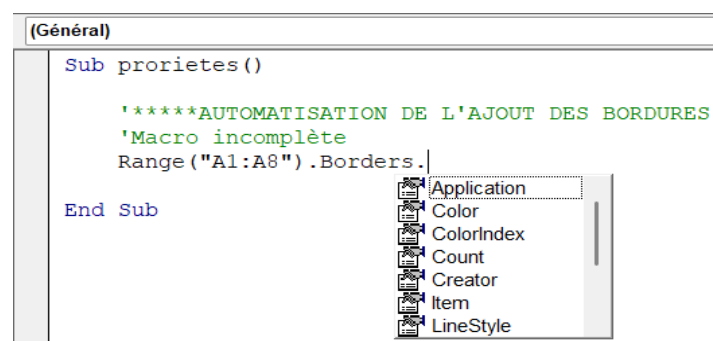
Sub proprietes()

    'Modifier la police de caractères des cellules A1 à A8
    Range("A1:A8").Font.Name = "Arial"

End Sub

```

AJOUTER DES BORDURES



```

Sub proprietes()

    'Ajouter une bordure aux cellules A1 à A8
    Range("A1:A8").Borders.Value = 1

    'Value = 0 : pas de bordure

End Sub

```

MODIFIER LES PROPRIÉTÉS D'UNE FEUILLE

```

Sub proprietes()

    'Masquer une feuille
    Sheets("Feuil3").Visible = 2

    'Visible = -1 : afficher la feuille

End Sub

```

MODIFIER LA VALEUR D'UNE CELLULE EN FONCTION D'UNE AUTRE

	A	B	C	D
1	Ma valeur			
2				
3				
4				
5				
6				
7	Cible			
8				
9				

L'objectif ici est que A7 prenne la valeur de A1, ce qui nous donne :

```

Sub proprietes()

    'A7 = A1
    Range("A7") = Range("A1")

    'Ou :
    'Range("A7").Value = Range("A1").Value

End Sub

```

Ou pour copier par exemple uniquement la taille du texte :

```
Sub proprietes()  
  
    Range("A7").Font.Size = Range("A1").Font.Size  
  
End Sub
```

Ce qui est à gauche du `=` prend la valeur de ce qui est à droite du `=`.

MODIFIER LA VALEUR D'UNE CELLULE EN FONCTION DE SA PROPRE VALEUR

Nous allons maintenant créer ici un compteur de clics.

A chaque clic, la valeur de A1 sera augmentée de 1 :

```
Sub proprietes()  
  
    'Compteur de clics en A1  
    Range("A1") = Range("A1") + 1  
  
End Sub
```

Cette ligne ne doit pas être interprétée comme une opération mathématique (rappelez-vous que ce qui est à gauche du `=` prend la valeur de ce qui est à droite du `=`).

Excel exécute le code ligne par ligne en respectant certaines priorités, ces commentaires devraient vous aider à mieux comprendre ce même code :

```
'Pour cet exemple : A1 vaut 10 avant l'exécution du code  
  
Sub proprietes()  
  
    'Un clic a été fait sur le bouton, nous entrons dans la procédure  
    'Pour le moment A1 vaut encore 10  
  
    'Pendant l'exécution de la ligne ci-dessous :  
    '- la valeur à droite du = est calculée en priorité (A1 vaut toujours 10, cela donne 10 + 1)  
    '- après calcul, la valeur à droite du = vaut donc 11  
    '- A1 prend ensuite la valeur à droite du = (soit la valeur 11)  
    Range("A1") = Range("A1") + 1  
  
    'A1 vaut alors 11 seulement après l'exécution de la ligne de code  
  
End Sub
```

WITH

Le code ci-dessous permet de définir différentes propriétés à la cellule A8 de la feuille 2 :

```
Sub proprietes()  
  
    Sheets("Feuil2").Range("A8").Borders.Weight = 3  
    Sheets("Feuil2").Range("A8").Font.Bold = True  
    Sheets("Feuil2").Range("A8").Font.Size = 18  
    Sheets("Feuil2").Range("A8").Font.Italic = True  
    Sheets("Feuil2").Range("A8").Font.Name = "Arial"  
  
End Sub
```

Nous pouvons utiliser **With** pour éviter les répétitions de `Sheets("Feuil2").Range("A8")` :

```
Sub proprietes()  
  
    'Début de l'instruction avec : With  
    With Sheets("Feuil2").Range("A8")  
        .Borders.Weight = 3  
        .Font.Bold = True  
        .Font.Size = 18  
        .Font.Italic = True  
        .Font.Name = "Arial"  
    'Fin de l'instruction avec : End With  
End With  
  
End Sub
```

`Sheets("Feuil2").Range("A8")` n'est donc plus répété.

Bien que ce ne soit pas indispensable dans ce cas, il est également possible de faire de même pour `.Font`, ce qui nous donnerait :

```
Sub proprietes()  
  
    With Sheets("Feuil2").Range("A8")  
        .Borders.Weight = 3  
        With .Font  
            .Bold = True  
            .Size = 18  
            .Italic = True  
            .Name = "Arial"  
        End With  
    End With  
  
End Sub
```

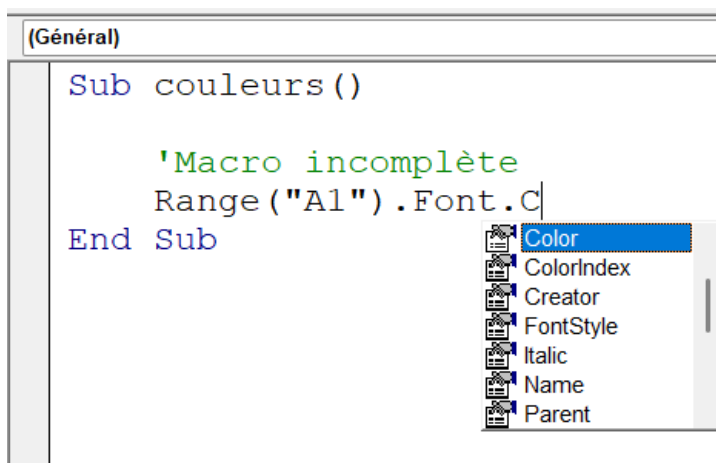
II.3 Les couleurs

Lancer Excel > l'enregistrer dans COURS BVA EXCEL sous le `nom 3.xlsm` > *Ouvrir l'éditeur* > *Ajouter un module* > Saisir la `macro incomplète` ci-dessous : > *Retourner dans Excel puis*

ajouter un bouton (Contrôle de formulaire) > Associer la macro portant le nom *couleurs* au bouton de formulaire.

Nous allons commencer par attribuer une couleur au texte en A1.

Après avoir ajouté *Font.*, nous obtenons :



Nous avons 2 possibilités pour définir la couleur : *ColorIndex* et ses 56 couleurs ou *Color* qui nous permettra d'utiliser n'importe quelle couleur.

COLORINDEX

Voici les 56 couleurs disponibles avec *ColorIndex* :

	A	B	C	D	E	F	G	H
1	1	2	3	4	5	6	7	8
2	9	10	11	12	13	14	15	16
3	17	18	19	20	21	22	23	24
4	25	26	27	28	29	30	31	32
5	33	34	35	36	37	38	39	40
6	41	42	43	44	45	46	47	48
7	49	50	51	52	53	54	55	56

Pour appliquer à notre texte l'une de ces 56 couleurs, nous écrirons :

```
Sub couleurs()  
  
    'Couleur du texte en A1 : vert (couleur 10)  
    Range("A1").Font.ColorIndex = 10  
  
End Sub
```

Ce qui nous donne :

	A	B	C	D
1	Exemple			
2				
3				
4				
5				
6				

Bouton 1

COLOR

Voici un exemple similaire avec Color :

```
Sub couleurs()  
  
    'Couleur du texte en A1 : RGB(0, 255, 0)  
    Range("A1").Font.Color = RGB(0, 255, 0)  
  
End Sub
```

La couleur ici est **RGB(0, 255, 0)**.

RGB en français signifie **R**ouge **V**ert **B**leu, les valeurs vont de 0 à 255 pour chaque couleur.

Quelques exemples de couleurs pour mieux comprendre :

- RGB(0, 0, 0) : noir
- RGB(255, 255, 255) : blanc
- RGB(255, 0, 0) : rouge
- RGB(0, 255, 0) : vert
- RGB(0, 0, 255) : bleu

Heureusement pour nous, il existe différentes solutions qui nous permettent de trouver facilement les valeurs RGB de la couleur qui nous intéresse.

Vous trouverez par exemple une liste de valeurs RGB sur le lien suivant :

<https://www.toutes-les-couleurs.com/code-couleur-rvb.php>

Pour donner une couleur violette à notre texte, nous pouvons donc rechercher les valeurs RGB de cette couleur sur la liste de couleurs dont le lien se trouve ci-dessus et entrer :

```
Sub couleurs()  
    'Couleur du texte en A1 : RGB(102, 0, 153)  
    Range("A1").Font.Color = RGB(102, 0, 153)  
End Sub
```

Ce qui nous donne :

	A	B	C	D
1	Exemple			
2				
3				
4				
5				

Bouton 1

CRÉER UNE BORDURE COLORÉE

Nous allons créer une macro qui va ajouter une bordure à la cellule active avec `ActiveCell`.

La bordure sera rouge et épaisse :

```
Sub couleurs()  
    'Epaisseur de la bordure  
    ActiveCell.Borders.Weight = 4  
    'Couleur de la bordure : rouge  
    ActiveCell.Borders.Color = RGB(255, 0, 0)  
End Sub
```

Aperçu :

	A	B	C	D
1				
2				
3				
4				

Bouton 1

COLORER LE FOND DES CELLULES SÉLECTIONNÉES

```
Sub couleurs()  
    'Colorer le fond des cellules sélectionnées  
    Selection.Interior.Color = RGB(176, 242, 182)  
End Sub
```

Aperçu :

	A	B	C	D
1				
2				
3				
4				
5				
6				
7				

Bouton 1

COLORER L'ONGLET D'UNE FEUILLE

```
Sub couleurs()  
    'Colorer l'onglet de la feuille "Feuil1"  
    Sheets("Feuil1").Tab.Color = RGB(255, 0, 0)  
End Sub
```

Aperçu :

21				
22				
23				
24				
25				
26				
27				
28				

< > Feuil1 Feuil2 +

Prêt   Accessibilité : consultez nos recommandations

III. Les variables

III.1 Les variables (Partie 1)

Lancer Excel > l'enregistrer dans COURS BVA EXCEL sous le nom *VariablesP1.xlsm* > *Ouvrir l'éditeur* > *Ajouter un module* > *Retourner dans Excel puis ajouter un bouton* (Contrôle de formulaire) > Associer la macro portant le nom *variablesP1* au bouton de formulaire.

Les variables permettent de stocker toutes sortes de données.

Voici un premier exemple :

```
'Affichage de la valeur de la variable dans une boîte de dialogue
Sub variables()

    'Déclaration de la variable
    Dim maVariable As Integer

    'Attribution d'une valeur à la variable
    maVariable = 12

    'Affichage de la valeur de maVariable dans une MsgBox
    MsgBox maVariable

End Sub
```

Cette première ligne de code est la déclaration de la variable (généralement placée en début de procédure).

```
Dim maVariable As Integer
```

- Dim : déclaration de la variable
- maVariable : nom choisi pour cette variable (sans espaces)
- As : déclaration du type de la variable
- Integer : type de la variable

Déclarer ses variables n'est pas obligatoire mais recommandé. Cela permet de s'y retrouver plus facilement, peut aider dans certains cas à résoudre plus facilement les problèmes, etc. Mieux vaut donc prendre l'habitude de déclarer correctement ses variables.

Le type de la variable indique la nature de son contenu (texte, nombres, date, etc.).

Une valeur est ensuite donnée à cette variable :

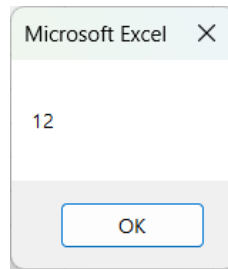
```
maVariable = 12
```

Et enfin, la valeur de la variable est affichée dans une boîte de dialogue :

```
MsgBox maVariable
```

MsgBox affiche une valeur dans une boîte de dialogue (les boîtes de dialogue seront détaillées dans quelques leçons).

Le résultat de ce code :



Si pour le moment vous ne comprenez pas bien l'intérêt d'utiliser des variables, soyez rassuré, les exemples abordés au cours des prochaines leçons vous en démontreront l'utilité.

LES TYPES DE VARIABLES

Nom	Type	Détails	Symbole
Byte	Numérique	Nombre entier de 0 à 255.	
Integer	Numérique	Nombre entier de -32'768 à 32'767.	%
Long	Numérique	Nombre entier de - 2'147'483'648 à 2'147'483'647.	&
Currency	Numérique	Nombre à décimale fixe de -922'337'203'685'477.5808 à 922'337'203'685'477.5807.	@
Single	Numérique	Nombre à virgule flottante de -3.402823E38 à 3.402823E38.	!
Double	Numérique	Nombre à virgule flottante de -1.79769313486232E308 à 1.79769313486232E308.	#
String	Texte	Texte.	\$
Date	Date	Date et heure.	
Boolean	Boolean	True (vrai) ou False (faux).	
Object	Objet	Objet.	
Variant	Tous	Tout type de données (type par défaut si la variable n'est pas déclarée).	

Quelques exemples avec différents types :

```

'Exemple : nombre entier
Dim nbEntier As Integer
nbEntier = 12345

'Exemple : nombre à virgule
Dim nbVirgule As Single
nbVirgule = 123.45

'Exemple : texte
Dim varTexte As String
varTexte = "Excel-Pratique.com"

'Exemple : date
Dim varDate As Date
varDate = "15/05/2025"

'Exemple : vrai/faux
Dim varBoolean As Boolean
varBoolean = True

'Exemple : objet (objet Worksheet pour cet exemple)
Dim varFeuille As Worksheet
Set varFeuille = Sheets("Feuil2") 'Set => attribution d'une valeur à une
variable objet

'Exemple d'utilisation de la variable objet : activation de la feuille
varFeuille.Activate

```

Les symboles indiqués dans le tableau ci-dessus permettent de raccourcir les déclarations de variables.

Par soucis de lisibilité, ils ne seront pas utilisés dans les leçons mais voici tout de même un exemple :

```

Dim exemple As Integer
Dim exemple%

```

Ces deux lignes sont identiques.

Il est possible de forcer les déclarations de variables en plaçant **Option Explicit** tout au début du module (une erreur sera ainsi générée en cas d'oubli de déclaration).

EXEMPLE PRATIQUE

Lancer Excel > l'enregistrer dans COURS BVA EXCEL sous le **nom exercice_variables.xlsm** > *Ouvrir l'éditeur* > **Ajouter un module** > *Retourner dans Excel puis ajouter un bouton* (Contrôle de formulaire) > Associer la macro portant le nom **variables** au bouton de formulaire.

Reproduire le tableau ci-dessous sur la feuille 1 :

	A	B	C	
1	Nom	Prénom	Age	
2	Smith	John	40	
3	Smith	John	41	
4	Smith	John	42	
5	Smith	John	43	
6	Smith	John	44	
7	Smith	John	45	
8	Smith	John	46	
9	Smith	John	47	
10	Smith	John	48	
11	Smith	John	49	
12	Smith	John	50	
13	Smith	John	51	
14	Smith	John	52	
15	Smith	John	53	
16	Smith	John	54	
17	Smith	John	55	
18				

Nous allons maintenant créer par étapes une macro qui va récupérer le nom dans la cellule A2, le prénom dans la cellule B2, l'âge dans la cellule C2 et qui va les afficher dans une boîte de dialogue.

Commençons par déclarer les variables (sur la même ligne, séparées par des virgules) :

```
Sub variables()  
    'Déclaration des variables  
    Dim nom As String, prenom As String, age As Integer  
End Sub
```

Attribuons ensuite les valeurs des cellules aux variables :

```
Sub variables()  
    'Déclaration des variables  
    Dim nom As String, prenom As String, age As Integer  
  
    'Valeurs des variables  
    nom = Cells(2, 1)  
    prenom = Cells(2, 2)  
    age = Cells(2, 3)  
End Sub
```

Et enfin, affichons le résultat dans la boîte de dialogue en concaténant les valeurs avec & (comme dans les formules Excel) :

```

Sub variables()

'Déclaration des variables
Dim nom As String, prenom As String, age As Integer

'Valeurs des variables
nom = Cells(2, 1)
prenom = Cells(2, 2)
age = Cells(2, 3)

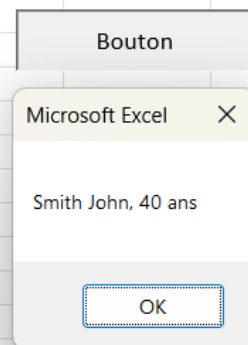
'Boîte de dialogue
MsgBox nom & " " & prenom & ", " & age & " ans"

End Sub

```

Ce qui nous donne :

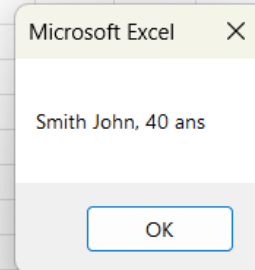
	A	B	C	D	E
1	Nom	Prénom	Age		
2	Smith	John	40		Bouton
3	Smith	John	41		
4	Smith	John	42		
5	Smith	John	43		
6	Smith	John	44		
7	Smith	John	45		
8	Smith	John	46		
9	Smith	John	47		
10	Smith	John	48		
11	Smith	John	49		
12	Smith	John	50		
13	Smith	John	51		
14	Smith	John	52		
15	Smith	John	53		
16	Smith	John	54		
17	Smith	John	55		



Nous allons maintenant chercher à afficher dans la boîte de dialogue la ligne du tableau correspondant au numéro indiqué dans la cellule F4.

Voici l'objectif :

	A	B	C	D	E	F	G
1	Nom	Prénom	Age				
2	Smith	John	40		Bouton		
3	Smith	John	41				
4	Smith	John	42		N° :	3	
5	Smith	John	43				
6	Smith	John	44				
7	Smith	John	45				
8	Smith	John	46				
9	Smith	John	47				
10	Smith	John	48				
11	Smith	John	49				
12	Smith	John	50				
13	Smith	John	51				
14	Smith	John	52				
15	Smith	John	53				
16	Smith	John	54				
17	Smith	John	55				



Prenez un moment pour effectuer cette modification vous-même avant de passer à la solution ci-dessous :

```
Sub variables()

'Déclaration des variables
Dim nom As String, prenom As String, age As Integer, numeroLigne As Integer

'Valeurs des variables
numeroLigne = Range("F4") + 1
nom = Cells(numeroLigne, 1)
prenom = Cells(numeroLigne, 2)
age = Cells(numeroLigne, 3)

'Boîte de dialogue
MsgBox nom & " " & prenom & ", " & age & " ans"

End Sub
```

Une variable **numeroLigne** a été ajoutée :

```
'Déclaration des variables
Dim nom As String, prenom As String, age As Integer, numeroLigne As Integer
```

La variable prend ensuite la valeur de la cellule F4 à laquelle nous ajoutons 1 (pour ne pas tenir compte de la première ligne qui contient les titres du tableau).

La variable **numeroLigne** aura donc pour valeur le numéro de ligne des cellules qui nous intéressent :

```
numeroLigne = Range("F4") + 1
```

Il ne reste plus qu'à remplacer les numéros de ligne dans `Cells` par notre variable :

```
nom = Cells(numeroLigne, 1)
prenom = Cells(numeroLigne, 2)
age = Cells(numeroLigne, 3)
```

Notre macro affiche maintenant la ligne du tableau qui nous intéresse.

	A	B	C	D	E	F	G
1	Nom	Prénom	Age				
2	Smith	John	40			Bouton	
3	Smith	John	41				
4	Smith	John	42				
5	Smith	John	43		N° :	7	
6	Smith	John	44				
7	Smith	John	45				
8	Smith	John	46				
9	Smith	John	47				
10	Smith	John	48				
11	Smith	John	49				
12	Smith	John	50				
13	Smith	John	51				
14	Smith	John	52				
15	Smith	John	53				
16	Smith	John	54				
17	Smith	John	55				

Microsoft Excel X
Smith John, 46 ans
OK

Notez au passage que nous pouvons réduire cette procédure entière sur une ligne :

```
Sub variables()  
    MsgBox Cells(Range("F4")+1,1) & " " & Cells(Range("F4")+1,2) & ", "  
    & Cells(Range("F4")+1,3) & " ans"  
End Sub
```

Le code fonctionne correctement, il est néanmoins beaucoup moins lisible que le précédent et plus difficile à entretenir par la suite (les codes ne seront donc pas réduits dans les leçons afin d'en faciliter la compréhension).

III.2 Les variables (Partie 2)

Dans le répertoire sur votre bureau nommé COURS BVA EXCEL, créer un classeur Excel et l'enregistrer sous le nom variablesP2.xlsm

LES TABLEAUX

Les variables permettent de stocker une seule valeur par variable, les tableaux permettent de stocker une multitude de valeurs par tableau (leur utilisation est proche de celle des variables).

Voici quelques exemples de déclarations : insérons pour cela une macro qu'on nommera tableaux.

LE TABLEAU À 1 DIMENSION

```
Sub tableaux1()  
    'Exemple de déclaration de tableau à 1 dimension  
    Dim tab1(4) As String  
End Sub
```

Dans cette déclaration, il n'y a qu'un chiffre entre parenthèses, il s'agit donc d'un tableau à une dimension.

Ce chiffre indique également le nombre de cases du tableau. Dans le cas présent, tab1(4) est un tableau dont les cases vont de 0 à 4, il s'agit donc d'un tableau comportant 5 cases :

0

1

2

3

4

Et voici comment attribuer des valeurs aux 5 cases de ce tableau :

```
Sub tableaux1()  
    'Exemple d'enregistrement de valeurs dans le tableau  
    tab1(0) = "Valeur de la case 0"  
    tab1(1) = "Valeur de la case 1"  
    tab1(2) = "Valeur de la case 2"  
    tab1(3) = "Valeur de la case 3"  
    tab1(4) = "Valeur de la case 4"  
End Sub
```

La première case d'un tableau est d'indice 0.

Ajoutons les lignes suivantes, pour afficher le tableau dans la colonne A (à partir de la ligne 1) :

```

Sub tableaux1()
    'Afficher le tableau dans la colonne A (à partir de la ligne 1)
    Dim i As String

    For i = 1 To 5
        Cells(i, 1).Value = tab1(i - 1)
    Next
End Sub

```

LE TABLEAU À 2 DIMENSIONS

```

Sub tableaux2()
    'Exemple de déclaration de tableau à 2 dimensions
    Dim tab2(4, 3) As String
End Sub

```

0, 0	0, 1	0, 2	0, 3
1, 0	1, 1	1, 2	1, 3
2, 0	2, 1	2, 2	2, 3
3, 0	3, 1	3, 2	3, 3
4, 0	4, 1	4, 2	4, 3

Et voici comment attribuer des valeurs aux cases d'un tableau à 2 dimensions :

```

Sub tableaux2()
    'Exemple d'enregistrement de valeurs dans le tableau à deux dimensions
    tab2(0, 0) = "Valeur de la case rouge"
    tab2(4, 1) = "Valeur de la case verte"
    tab2(2, 3) = "Valeur de la case bleue"
End Sub

```

Nous reviendrons sur les tableaux plus tard dans ce cours.

LES CONSTANTES

Les constantes permettent de stocker des valeurs comme les variables, à la différence près qu'on ne peut pas les modifier (d'où leur nom) après les avoir déclarées.

Par exemple, ces quelques lignes calculent le montant de la TVA en fonction d'un taux de TVA de 12.34% :

Pour cela insérer un nouveau module pour créer une macro à nommer constante

```
Sub constante()  
  
Cells(1, 1) = Cells(1, 2) * 0.1234  
Cells(2, 1) = Cells(2, 2) * 0.1234  
Cells(3, 1) = Cells(3, 2) * 0.1234  
Cells(4, 1) = Cells(4, 2) * 0.1234  
Cells(5, 1) = Cells(5, 2) * 0.1234  
  
End Sub
```

Pour éviter les répétitions et faciliter la lecture de ce code, il est possible de déclarer le taux de TVA sous forme de constante :

```
Sub constante()  
  
    'Déclaration de la constante + attribution de sa valeur  
    Const TAUX_TVA As Double = 0.1234  
  
    Cells(1, 1) = Cells(1, 2) * TAUX_TVA  
    Cells(2, 1) = Cells(2, 2) * TAUX_TVA  
    Cells(3, 1) = Cells(3, 2) * TAUX_TVA  
    Cells(4, 1) = Cells(4, 2) * TAUX_TVA  
    Cells(5, 1) = Cells(5, 2) * TAUX_TVA  
  
End Sub
```

En utilisant une constante, le jour où le taux de TVA changera, il vous suffira de modifier une seule fois la valeur de la constante dans le code (au lieu de rechercher et remplacer toutes les valeurs 0.1234 dans le code).

Par convention, une constante se nomme en majuscules en séparant les mots par un `_` (par exemple : `EXEMPLE_DE_NOM`).

LA PORTÉE DES VARIABLES

Si la variable est déclarée au début d'une procédure (Sub), elle ne peut être utilisée que dans cette même procédure. La valeur de la variable n'est pas conservée après l'exécution de la procédure.

```

Sub procedure1()

    Dim var1 As Integer

    '=> Utilisation de la variable dans la procédure uniquement

End Sub

Sub procedure2()

    '=> Impossible d'utiliser var1 ici

End Sub

```

Pour pouvoir utiliser une variable dans toutes les procédures d'un module, il suffit de la déclarer en début de module. De plus, cela permet de conserver la valeur de la variable jusqu'à la fermeture du classeur.

```

Dim var1 As Integer

Sub procedure1()

    '=> Utilisation de var1 possible

End Sub

Sub procedure2()

    '=> Utilisation de var1 possible

End Sub

```

Même principe pour utiliser une variable dans tous les modules, à la différence près que **Dim** est remplacé par **Public** :

```

Public var1 As Integer

```

Pour conserver la valeur d'une variable à la fin d'une procédure, remplacez Dim par Static :

```

Sub procedure1()

    Static var1 As Integer

End Sub

```

Pour conserver les valeurs de toutes les variables d'une procédure, ajoutez Static devant Sub :

```

Static Sub procedure1()

    Dim var1 As Integer

End Sub

```

CRÉER SON PROPRE TYPE DE VARIABLE

Nous n'allons pas nous attarder sur ce point, voici juste un exemple :

'Création d'un type de variable

```
Type Utilisateur
    Nom As String
    Prenom As String
End Type

Sub utilisationTypeVariables()

    'Déclaration
    Dim user1 As Utilisateur

    'Attributions des valeurs à user1
    user1.Nom = "Smith"
    user1.Prenom = "John"

    'Exemple d'utilisation
    MsgBox user1.Nom & " " & user1.Prenom

End Sub
```

IV. Les conditions

IV. 1 Les conditions (Partie 1)

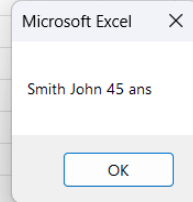
Les conditions sont très utiles en programmation, elles nous serviront à effectuer des actions en fonction de critères précis (même principe que la fonction SI).

La principale instruction est **If**, voici comment elle fonctionne :

```
If [CONDITION] Then '=> SI condition vraie ALORS
    'Instructions si vrai
Else '=> SINON (facultatif)
    'Instructions si faux
End If
```

Passons directement à la pratique et reprenons l'exemple développé à la leçon sur les variables. Il avait pour but d'afficher dans une boîte de dialogue la ligne du tableau correspondant au numéro indiqué dans la cellule F4.

	A	B	C	D	E	F
1	nom	Prenom	Age			
2	Smith	John	40			Bouton 1
3	Smith	John	41			
4	Smith	John	42		N°	6
5	Smith	John	43			
6	Smith	John	44			
7	Smith	John	45			
8	Smith	John	46			
9	Smith	John	47			
10	Smith	John	48			
11	Smith	John	49			
12	Smith	John	50			
13	Smith	John	51			
14	Smith	John	52			
15	Smith	John	53			
16	Smith	John	54			
17	Smith	John	55			



Si nous entrons une lettre en F4, cela génère un bug et nous voulons éviter cela. Le code initial était :

```
Sub variables()

'Déclaration des variables
Dim nom As String, prenom As String, age As Integer,
Dim numeroLigne As Integer

'Valeurs des variables
numeroLigne = Range("F4") + 1
nom = Cells(numeroLigne, 1)
prenom = Cells(numeroLigne, 2)
age = Cells(numeroLigne, 3)

'Boîte de dialogue
MsgBox nom & " " & prenom & ", " & age & " ans"

End Sub
```

Nous allons commencer par ajouter une condition pour vérifier si la valeur de la cellule F4 est bien numérique avant d'exécuter le code.

La fonction IsNumeric sera utilisée dans cette condition :

```
Sub variables()

'Si la valeur entre parenthèses (cellule F4) est numérique (donc si
'la condition est vraie) alors on exécute les instructions placées
'entre "Then" et "End If"
If IsNumeric(Range("F4")) Then

'Déclaration des variables
Dim nom As String, prenom As String, age As Integer,
Dim numeroLigne As Integer

'Valeurs des variables
numeroLigne = Range("F4") + 1
nom = Cells(numeroLigne, 1)
```



```

    prenom = Cells(numeroLigne, 2)
    age = Cells(numeroLigne, 3)

    'Boîte de dialogue
    MsgBox nom & " " & prenom & ", " & age & " ans"

End If

End Sub

```

Ajoutons également des instructions pour le cas où la condition n'est pas remplie :

```

Sub variables()

    'Si F4 est numérique
    If IsNumeric(Range("F4")) Then

        'Déclaration des variables
        Dim nom As String, prenom As String, age As Integer,
        Dim numeroLigne As Integer

        'Valeurs des variables
        numeroLigne = Range("F4") + 1
        nom = Cells(numeroLigne, 1)
        prenom = Cells(numeroLigne, 2)
        age = Cells(numeroLigne, 3)

        'Boîte de dialogue
        MsgBox nom & " " & prenom & ", " & age & " ans"

    'Si F4 n'est pas numérique
    Else

        'Boîte de dialogue : avertissement
        MsgBox "L'entrée " & Range("F4") & " n'est pas valide !"

        'Suppression du contenu de la cellule F4
        Range("F4") = ""

    End If

End Sub

```

Les valeurs non numériques ne sont désormais plus un problème.

Notre tableau contient 16 lignes de données (de la ligne 2 à la ligne 17), nous allons donc vérifier maintenant si la variable `numeroLigne` est *plus grande ou égale à 2* et *plus petite ou égale à 17*.

Mais avant, voici les opérateurs de comparaison :

=	Est égal à
<>	Est différent de
<	Est plus petit que
<=	Est plus petit ou égal à
>	Est plus grand que
>=	Est plus grand ou égal à

Ainsi que d'autres opérateurs utiles :

And	Et	[CONDITION 1] And [CONDITION 2] Les 2 conditions doivent être vraies
Or	Ou	[CONDITION 1] Or [CONDITION 2] Au moins 1 des 2 conditions doit être vraie
Not	Faux	Not [CONDITION] La condition doit être fausse
Mod	Modulo	[NOMBRE] Mod [DIVISEUR] Cet opérateur retourne le reste d'une division

Ajoutons maintenant les conditions indiquées un peu plus haut en utilisant And ainsi que les opérateurs de comparaison détaillés ci-dessus :

```
Sub variables()  
    'Si F4 est numérique  
    If IsNumeric(Range("F4")) Then  
  
        Dim nom As String, prenom As String, age As Integer,  
        Dim numeroLigne As Integer  
        numeroLigne = Range("F4") + 1  
  
        'Si le numéro est dans la bonne plage  
        If numeroLigne >= 2 And numeroLigne <= 17 Then  
            nom = Cells(numeroLigne, 1)  
            prenom = Cells(numeroLigne, 2)  
            age = Cells(numeroLigne, 3)  
            MsgBox nom & " " & prenom & ", " & age & " ans"
```

```

'Si le numéro est en dehors de la plage
Else
    MsgBox "L'entrée " & Range("F4") & " n'est pas un numéro »"
    MsgBox "valide !"
    Range("F4") = ""
End If

'Si F4 n'est pas numérique
Else
    'Boîte de dialogue : avertissement
    MsgBox "L'entrée " & Range("F4") & " n'est pas valide !"
    Range("F4") = ""

End If
End Sub

```

Pour rendre notre macro plus pratique, nous pouvons encore remplacer **17** par une variable contenant le nombre de lignes. Cela nous permettra d'ajouter/retirer des lignes à notre tableau sans avoir à modifier à chaque fois cette limite dans le code.

Pour cela, créons une variable **nbLignes** et ajoutons cette fonction : WorksheetFunction.CountA

Il s'agit en fait de la fonction NBVAL que vous connaissez probablement déjà.

Nous demandons à cette fonction de comptabiliser le nombre de cellules non vides de la première colonne et nous remplaçons ensuite **17** par **nbLignes** :

```

Sub variables()

'Si F4 est numérique
If IsNumeric(Range("F4")) Then

    Dim nom As String, prenom As String, age As Integer,
    Dim numeroLigne As Integer, nbLignes As Integer

    numeroLigne = Range("F4") + 1
    nbLignes = WorksheetFunction.CountA(Range("A:A")) 'Fonction NBVAL

'Si le numéro est dans la bonne plage
If numeroLigne >= 2 And numeroLigne <= nbLignes Then
    nom = Cells(numeroLigne, 1)
    prenom = Cells(numeroLigne, 2)
    age = Cells(numeroLigne, 3)
    MsgBox nom & " " & prenom & ", " & age & " ans"

'Si le numéro est en dehors de la plage
Else
    MsgBox "L'entrée " & Range("F4") & " n'est pas un numéro »"
    MsgBox "valide !"
    Range("F4") = ""
End If

```

```

'Si F4 n'est pas numérique
Else

    'Boîte de dialogue : avertissement
    MsgBox "L'entrée "" & Range("F4") & "" n'est pas valide !"
    Range("F4") = ""

End If

End Sub

```

ELSEIF

ElseIf permet d'ajouter plusieurs conditions à la suite :

```

If [CONDITION 1] Then '=> SI la condition 1 est vraie ALORS
    'Instructions 1
ElseIf [CONDITION 2] Then '=> SINON, SI la condition 2 est vraie ALORS
    'Instructions 2
Else '=> SINON
    'Instructions 3
End If

```

Si la condition 1 est vraie, les instructions 1 sont exécutées puis nous sortons de l'instruction **If** (qui débute avec **If** et se termine à **End If**). Si la condition 1 est fausse, nous passons à la condition 2. Si celle-ci est vraie les instructions 2 sont exécutées si ce n'est pas le cas les instructions 3 sont alors exécutées.

```

Sub commentaires()

    'Variables
    Dim note As Single, commentaire As String
    note = Range("A1")

    'Commentaire en fonction de la note
    If note = 6 Then
        commentaire = "Excellent résultat !"
    ElseIf note >= 5 Then
        commentaire = "Bon résultat"
    ElseIf note >= 4 Then
        commentaire = "Résultat satisfaisant"
    ElseIf note >= 3 Then
        commentaire = "Résultat insatisfaisant"
    ElseIf note >= 2 Then
        commentaire = "Mauvais résultat"
    ElseIf note >= 1 Then
        commentaire = "Résultat exécrable"
    Else
        commentaire = "Aucun résultat"
    End If

    'Commentaire en B1
    Range("B1") = commentaire

End Sub

```

SELECT

Une alternative aux instructions *If* contenant beaucoup de *ElseIf* existe, il s'agit de *Select* (cette instruction étant plus adaptée dans ce genre de cas).

Voici la même macro avec *Select* :

```
Sub commentaires()  
  
    'Variables  
    Dim note As Single, commentaire As String  
    note = Range("A1")  
  
    'Commentaire en fonction de la note  
    Select Case note '  
        Case Is = 6  
            commentaire = "Excellent résultat !"  
        Case Is >= 5  
            commentaire = "Bon résultat"  
        Case Is >= 4  
            commentaire = "Résultat satisfaisant"  
        Case Is >= 3  
            commentaire = "Résultat insatisfaisant"  
        Case Is >= 2  
            commentaire = "Mauvais résultat"  
        Case Is >= 1  
            commentaire = "Résultat exécrable"  
        Case Else  
            commentaire = "Aucun résultat"  
    End Select  
  
    'Commentaire en B1  
    Range("B1") = commentaire  
  
End Sub
```

```

Sub commentaires()

    'Variables
    Dim note As Single, commentaire As String
    note = Range("A1")

    'Commentaire en fonction de la note
    Select Case note '<= la valeur à tester (ici, la note)
        Case Is = 6
            commentaire = "Excellent résultat !"
        Case Is >= 5
            commentaire = "Bon résultat"
        Case Is >= 4
            commentaire = "Résultat satisfaisant"
        Case Is >= 3
            commentaire = "Résultat insatisfaisant"
        Case Is >= 2
            commentaire = "Mauvais résultat"
        Case Is >= 1
            commentaire = "Résultat exécration"
        Case Else
            commentaire = "Aucun résultat"
    End Select

    'Commentaire en B1
    Range("B1") = commentaire

End Sub

```

Notez que nous pouvons également entrer plusieurs valeurs :

```

Case Is = 6, 7 'Si la valeur = 6 ou 7
Case Is <> 6, 7 'Si la valeur est différente de 6 ou 7

```

Ou une plage de valeurs :

```

Case 6 To 10 'Si la valeur = de 6 à 10

```

IV.2 Les conditions (Partie 2)

La fonction *IsNumeric* (vue à la page précédente) renvoie True (vrai) si la valeur est numérique et False (faux) si ce n'est pas le cas :

```

If IsNumeric(Range("A1")) = True Then

```

```

If IsNumeric(Range("A1")) Then

```

Ces 2 lignes sont identiques (il n'est pas nécessaire d'entrer *= True* puisque que l'on cherche de toute manière à savoir si l'expression est vraie).

Dans le cas où nous voulons vérifier si la valeur n'est pas numérique, nous avons également deux possibilités :

```
If IsNumeric(Range("A1")) = False Then 'Si la valeur n'est pas numérique
```

```
If Not IsNumeric(Range("A1")) Then 'Si la valeur n'est pas numérique
```

Il existe de nombreuses autres fonctions que vous pouvez utiliser dans vos conditions (ou plus généralement dans vos codes VBA).

FONCTIONS DE DATES

Il existe de nombreuses fonctions de dates et d'heures pouvant être utilisées dans des conditions, en voici quelques exemples.

La fonction *IsDate* renvoie True si la valeur est une date ou False si ce n'est pas le cas :

```
If IsDate(Range("A1")) Then 'Si la valeur est une date
```

La fonction *Day* permet d'extraire le jour d'une date :

```
If Day(Range("A1")) = 1 Then 'Si c'est le premier jour du mois
```

La fonction *Year* permet d'extraire l'année d'une date :

```
If Year(Range("A1")) = 2025 Then 'Si c'est une date de l'année 2025
```

La fonction *Date* renvoie la date actuelle :

```
If Range("A1") < Date Then 'Si la date est passée
```

FONCTION ISEMPTY

La fonction *IsEmpty* renvoie False si la variable a été initialisée ou True si ce n'est pas le cas :

```
If IsEmpty(maVariable) Then 'Si la variable n'a pas été initialisée
```

Dans cet exemple, la condition est vraie car aucun type ni valeur n'ont été attribués à *maVariable* :

```
Sub fonctions()  
    Dim maVariable  
    If IsEmpty(maVariable) Then  
        MsgBox "Ma variable n'a pas été initialisée !"  
    Else  
        MsgBox "Ma variable contient : " & maVariable  
    End If  
End Sub
```

V. Les boucles

V.1 Boucles

Les boucles permettent de répéter des instructions un certain nombre de fois pour vous éviter de devoir écrire des macros d'une longueur interminable et vous faire gagner un temps considérable.

Le code suivant numérote les cellules de la colonne A (de la ligne 1 à 12) :

```
Sub numerotation()  
    Cells(1, 1) = 1  
    Cells(2, 1) = 2  
    Cells(3, 1) = 3  
    Cells(4, 1) = 4  
    Cells(5, 1) = 5  
    Cells(6, 1) = 6  
    Cells(7, 1) = 7  
    Cells(8, 1) = 8  
    Cells(9, 1) = 9  
    Cells(10, 1) = 10  
    Cells(11, 1) = 11  
    Cells(12, 1) = 12  
End Sub
```

Ce code est très répétitif ...

Maintenant, imaginez qu'il faille numéroter plusieurs milliers de lignes ... Vous comprenez donc probablement l'intérêt de créer des boucles.

Voici la boucle **Do** :

```
Sub syntaxe()  
    Do While [CONDITION]  
        'Instructions  
    Loop  
End Sub
```

Tant que la condition est vraie, les instructions sont exécutées en boucle (attention à ne pas créer une boucle infinie).

Voici la macro répétitive ci-dessus avec la boucle **Do** :

```
Sub numerotationDoWhile ()  
    Dim numero As Integer  
    numero = 1 'Numéro de départ  
    Do While numero <= 12 'Tant que la variable numero est <= 12, la boucle  
        'est répétée  
        Cells(numero, 1) = numero 'Numérotation  
        numero = numero + 1 'Le numéro est augmenté de 1 à chaque boucle  
End Sub
```


Avec cette boucle, si nous voulons numéroter 500 lignes, il suffit alors de remplacer 12 par 500 ...

DO LOOP

Dans le précédent exemple, vous avez pu voir la boucle **Do** sous la forme suivante :

```
Sub syntaxe()  
    Do While [CONDITION]  
        'Instructions  
    Loop  
End Sub
```

Avec **Do**, la condition peut également être placée en fin de boucle, ce qui implique que les instructions seront dans tous les cas exécutées au moins une fois :

```
Sub syntaxe()  
    Do  
        'Instructions  
    Loop While [CONDITION]  
End Sub
```

Plutôt que de répéter la boucle tant que la condition est vraie, il est possible de quitter la boucle lorsque la condition est vraie en remplaçant **While** par **Until** :

```
Sub syntaxe()  
    Do Until [CONDITION]  
        'Instructions  
    Loop  
End Sub
```

FOR NEXT

```
Sub syntaxe()  
    Dim i As Integer  
    For i = 1 To 5  
        'Instructions  
    Next  
End Sub
```

La boucle **For** est répétée ici 5 fois.

A chaque répétition de la boucle, la variable **i** est automatiquement incrémentée de 1 :

```

Sub boucleFor()

    Dim i As Integer

    For i = 1 To 5
        MsgBox i 'Renvoie les valeurs : 1 / 2 / 3 / 4 / 5
    Next

End Sub

```

Si nécessaire, vous pouvez modifier l'incrément (par défaut à 1) de la boucle en ajoutant **Step** :

```

Sub boucleFor()

    Dim i As Integer

    For i = 10 To 0 Step -2
        MsgBox i 'Renvoie les valeurs : 10 / 8 / 6 / 4 / 2 / 0
    Next

End Sub

```

FOR EACH NEXT

La boucle **For Each** permet de parcourir chaque élément d'un ensemble d'éléments, par exemple parcourir chaque cellule d'une plage de cellules :

```

Sub boucleForEach()

    Dim cellule As Range

    For Each cellule In Range("A1:B3")
        cellule = cellule.Address
    Next

End Sub

```

Parcourir chaque feuille du classeur :

```

Sub boucleForEach()

    Dim feuille As Worksheet

    For Each feuille In Worksheets
        MsgBox feuille.Name
    Next

End Sub

```

Parcourir chaque élément d'un tableau :

```
Sub boucleForEach()  
    Dim tableau(2) As String  
  
    tableau(0) = "A"  
    tableau(1) = "B"  
    tableau(2) = "C"  
  
    For Each valeur In tableau  
        MsgBox valeur  
    Next  
End Sub
```

QUITTER UNE BOUCLE PRÉMATURÉMENT

Il est possible de quitter une boucle **For** prématurément grâce à l'instruction suivante :

```
Exit For 'Quitter une boucle For
```

Dans cet exemple, l'objectif est de retourner le numéro de la première ligne contenant la valeur **1**. Lorsque cet objectif est atteint, le numéro est affiché et la boucle est interrompue (car il est dans ce cas inutile de parcourir les autres lignes) :

```
Sub quitterBoucleFor()  
    Dim i As Integer  
  
    'Boucle pour 100 lignes au maximum  
    For i = 1 To 100  
        'Si la cellule vaut 1  
        If Cells(i, 1) = 1 Then 'Si l'objectif est atteint  
            MsgBox "La cellule a été trouvée à la ligne " & i & " !"  
            Exit For 'On quitte la boucle For  
        End If  
    Next  
End Sub
```

Les autres instructions **Exit** :

```
Exit Do 'Quitter une boucle Do
```

```
Exit Sub 'Quitter une procédure
```

```
Exit Function 'Quitter une fonction
```

V.2 Boucles (exercice)

Pour mettre en pratique ce qui a été vu jusque-là, nous allons créer étape par étape une macro qui va numéroté de 1 à 100 une plage de cellules carrée de 10 par 10 et colorer une cellule sur 2, aperçu :

Voici le point de départ de l'exercice :

```
Sub exerciceBoucles()  
    ...  
End Sub
```

Pour commencer, ajoutez une boucle For qui va numéroté de 1 à 10 les cellules de la ligne 1, aperçu :

	A	B	C	D	E	F	G	H	I	J	K	L
1	1	2	3	4	5	6	7	8	9	10		
2												
3												
4												
5												
6												
7												

Bouton 1

Prenez quelques instants pour créer cette boucle avant de passer à la solution ci-dessous :

```
Sub exerciceBoucles()  
    Dim colonne As Integer  
    'Boucle des colonnes  
    For colonne = 1 To 10  
        Cells(1, colonne) = colonne  
    Next  
End Sub
```

Créez maintenant une seconde boucle qui va répéter la première boucle sur 10 lignes, aperçu :

	A	B	C	D	E	F	G	H	I	J	K	L
1	1	2	3	4	5	6	7	8	9	10		
2	1	2	3	4	5	6	7	8	9	10		
3	1	2	3	4	5	6	7	8	9	10		
4	1	2	3	4	5	6	7	8	9	10	Bouton 1	
5	1	2	3	4	5	6	7	8	9	10		
6	1	2	3	4	5	6	7	8	9	10		
7	1	2	3	4	5	6	7	8	9	10		
8	1	2	3	4	5	6	7	8	9	10		
9	1	2	3	4	5	6	7	8	9	10		
10	1	2	3	4	5	6	7	8	9	10		
11												
12												

Solution :

```
Sub exerciceBoucles()
    Dim colonne As Integer, ligne As Integer
    'Boucle pour parcourir les lignes de 1 à 10 en remplissant les
    colonnes de 1 à 10
    For ligne = 1 To 10
        'Boucle des colonnes
        For colonne = 1 To 10
            Cells(ligne, colonne) = colonne
        Next
    Next
End Sub
```

Les lignes sont pour le moment numérotées de 1 à 10.

Trouvez maintenant une solution pour obtenir une numérotation de 1 à 100, aperçu :

	A	B	C	D	E	F	G	H	I	J	K	L
1	1	2	3	4	5	6	7	8	9	10		
2	11	12	13	14	15	16	17	18	19	20		
3	21	22	23	24	25	26	27	28	29	30		
4	31	32	33	34	35	36	37	38	39	40	Bouton 1	
5	41	42	43	44	45	46	47	48	49	50		
6	51	52	53	54	55	56	57	58	59	60		
7	61	62	63	64	65	66	67	68	69	70		
8	71	72	73	74	75	76	77	78	79	80		
9	81	82	83	84	85	86	87	88	89	90		
10	91	92	93	94	95	96	97	98	99	100		
11												

Une solution "simple" consiste à utiliser une variable qui sera incrémentée de 1 après chaque entrée dans une cellule :

```
Sub exerciceBoucles()
    Dim colonne As Integer, ligne As Integer, numero As Integer
    'Valeur de la première cellule
    numero = 1
    'Boucle pour parcourir les lignes de 1 à 10 en remplissant les colonnes de 1 à 10
    For ligne = 1 To 10
        'Boucle des colonnes
        For colonne = 1 To 10
            Cells(ligne, colonne) = numero
            numero = numero + 1 'Valeur incrémentée de 1
        Next
    Next
End Sub
```

Une autre solution consiste à calculer la valeur à insérer dans la cellule à l'aide des numéros de colonne et de ligne :

```
Sub exerciceBoucles()
    Dim colonne As Integer, ligne As Integer
    'Boucle pour parcourir les lignes de 1 à 10 en remplissant les colonnes de 1 à 10
    For ligne = 1 To 10
        'Boucle des colonnes
        For colonne = 1 To 10
            Cells(ligne, colonne) = (ligne-1) * 10 + colonne
        Next
    Next
End Sub
```

```

Next
Next
End Sub

```

Pour terminer l'exercice, il reste encore à colorer le fond d'une cellule sur 2 à l'aide d'une instruction `If` et de l'opérateur `Mod` (qui retourne le reste d'une division), aperçu :

	A	B	C	D	E	F	G	H	I	J
1	1	2	3	4	5	6	7	8	9	10
2	11	12	13	14	15	16	17	18	19	20
3	21	22	23	24	25	26	27	28	29	30
4	31	32	33	34	35	36	37	38	39	40
5	41	42	43	44	45	46	47	48	49	50
6	51	52	53	54	55	56	57	58	59	60
7	61	62	63	64	65	66	67	68	69	70
8	71	72	73	74	75	76	77	78	79	80
9	81	82	83	84	85	86	87	88	89	90
10	91	92	93	94	95	96	97	98	99	100

Solution :

```

Sub exerciceBoucles()
    Dim colonne As Integer, ligne As Integer
    'Boucle pour parcourir les lignes de 1 à 10 en remplissant les
    colonnes de 1 à 10
    For ligne = 1 To 10
        'Boucle des colonnes
        For colonne = 1 To 10
            Cells(ligne, colonne) = (ligne-1) * 10 + colonne

            'Coloration d'une cellule sur 2
            If (ligne + colonne) Mod 2 = 0 Then 'Si le reste de la division
            'par 2 = 0
                Cells(ligne, colonne).Interior.Color = RGB(220, 220, 220)
            End If
        Next
    Next
End Sub

```

La condition $(\text{ligne} + \text{colonne}) \bmod 2 = 0$ est vraie si le reste de la division de (ligne + colonne) par 2 est égal à 0 (sachant que le reste de la division d'un nombre entier positif par 2 ne peut être que 0 ou 1).

Pour prendre un exemple plus simple, si l'objectif de l'exercice était de colorer les lignes paires (sans tenir compte des colonnes), la condition aurait été $\text{ligne} \bmod 2 = 0$.

VI. Les procédures et fonctions

Pour le moment, toutes les procédures créées sont de type *Public*, elles sont accessibles depuis tous les modules.

```
Sub procedure()  
    'Est identique à :  
Public Sub procedure()
```

Pour rendre une procédure inaccessible hors du module, ajoutez *Private* :

```
Private Sub procedure ()
```

VI.1 Lancer une procédure depuis une procédure

Pour exécuter une procédure depuis une autre procédure, entrez simplement son nom.

Un exemple simple :

```
Private Sub procedure1()  
    MsgBox "Attention !!!"  
End Sub  
  
Sub procedure()  
    If Range("A1") = "" Then  
        procedure1'<= exécute la procédure "procedure1"  
    End If  
End Sub
```

Ici, lorsque la procédure procedure est lancée et que A1 vaut "", la procédure procedure1 est exécutée et affiche la boîte de dialogue.

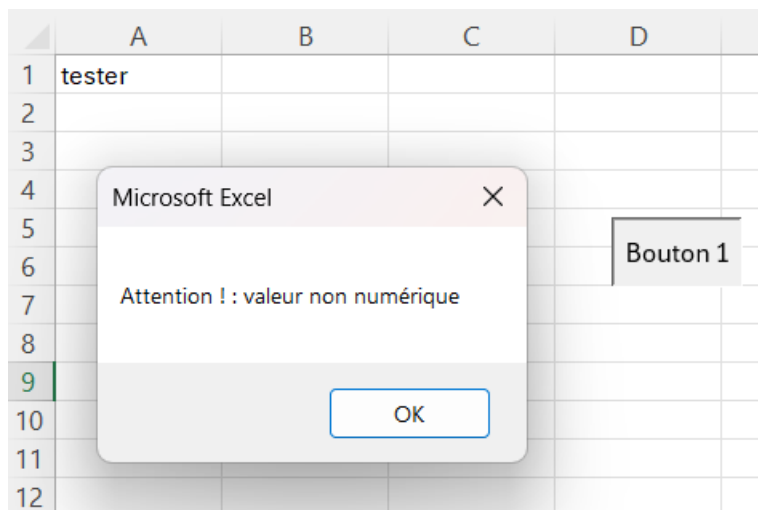
LES ARGUMENTS

Les arguments permettent de transmettre des valeurs d'une procédure à une autre (car rappelez-vous que par défaut les variables ne sont pas accessibles depuis les autres procédures).

Ajout d'un argument *texte* à la procédure *procedure1* :

```
Private Sub procedure1 (texte As String)
    MsgBox "Attention : " & texte & " !"
End Sub

Sub procedure()
    If Range("A1") = "" Then 'Si A1 est vide
        Procedure1 "cellule vide"
    ElseIf Not IsNumeric(Range("A1")) Then 'Si A1 est non numérique
        Procedure1 "valeur non numérique"
    End If
End Sub
```



L'argument ajouté à la procédure *procedure1* est de type *String* :

```
Private Sub procedure1 (texte As String)
```

Pour exécuter la procédure *procedure1*, il faudra donc entrer en argument une valeur de type *String* :

```
procedure1 "cellule vide"
```

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

LES ARGUMENTS OPTIONNELS

Par défaut, si une procédure requiert des arguments, ceux-ci sont obligatoires pour exécuter la procédure.

Des arguments optionnels peuvent toutefois être ajoutés après les arguments obligatoires avec *Optional*, par exemple :

```
Private Sub boiteDialogue(nom As String, Optional prenom, Optional age)
```

Cette procédure peut alors être lancée avec ou sans arguments optionnels, comme ceci :

```
'Exemple 1 : on affiche le nom
boiteDialogue nom

'Exemple 2 : on affiche le nom et le prénom
boiteDialogue nom, prenom

'Exemple 3 : on affiche le nom et l'âge
boiteDialogue nom, , age

'Exemple 4 : on affiche le nom, le prénom et l'âge
boiteDialogue nom, prenom, age
```

Les arguments doivent être indiqués dans l'ordre.

Pour vérifier si un argument optionnel est présent ou non, nous utiliserons la fonction *IsMissing*. Cette fonction n'étant compatible qu'avec certains types de variables (dont Variant), le type des arguments optionnels n'a pas été déclaré (type non déclaré = Variant).

Voici un exemple avec les 2 portions de code ci-dessus :

```
Sub procedureP()

    Dim nom As String, prenom As String, age As Integer

    nom = Range("A1")
    prenom = Range("B1")
    age = Range("C1")

    'Exemple 1 : on affiche le nom
    boiteDialogue nom

    'Exemple 2 : on affiche le nom et le prénom
    boiteDialogue nom, prenom

    'Exemple 3 : on affiche le nom et l'âge
    boiteDialogue nom, , age

    'Exemple 4 : on affiche le nom, le prénom et l'âge
    boiteDialogue nom, prenom, age

End Sub

Private Sub boiteDialogue(nom As String, Optional prenom, Optional age)
```

```

'Si l'âge est manquant
If IsMissing(age) Then

    If IsMissing(prenom) Then 'Si le prénom est manquant, on n'affiche
    'que le nom
        MsgBox nom
    Else 'Sinon, on affiche le nom et le prénom
        MsgBox nom & " " & prenom
    End If

'Si l'âge a été renseigné
Else

    If IsMissing(prenom) Then 'Si le prénom est manquant, on affiche
    '    le nom et l'âge
        MsgBox nom & ", " & age & " ans"
    Else 'Sinon on affiche le nom, le prénom et l'âge
        MsgBox nom & " " & prenom & ", " & age & " ans"
    End If

End If

End Sub

```