

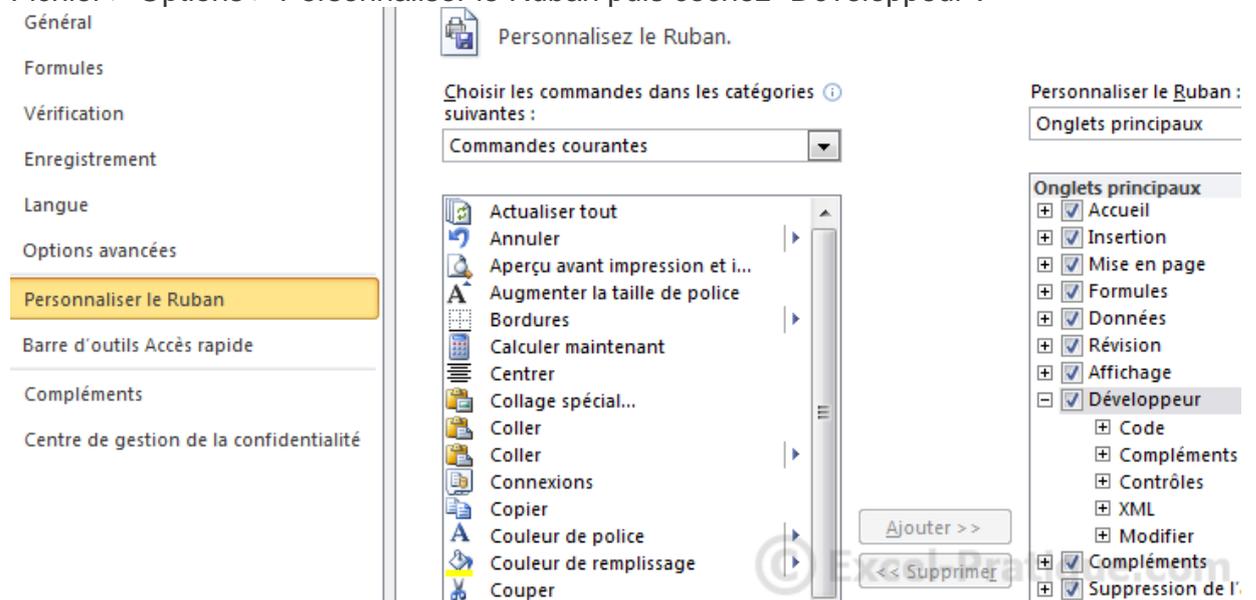
Cours VBA : introduction

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).

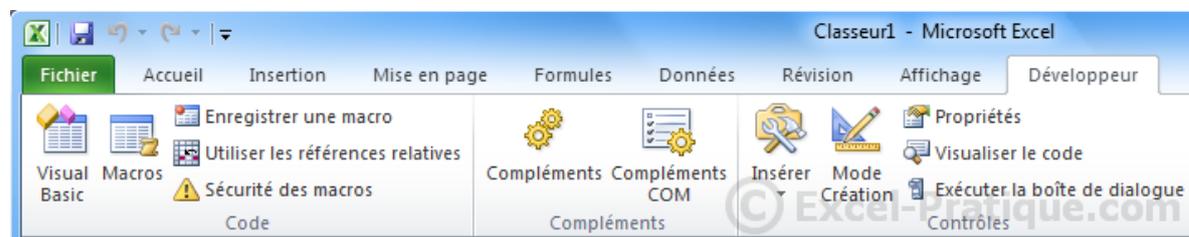
Grâce au VBA nous allons pouvoir réaliser à peu près tout ce que l'on souhaite avec Excel ...

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

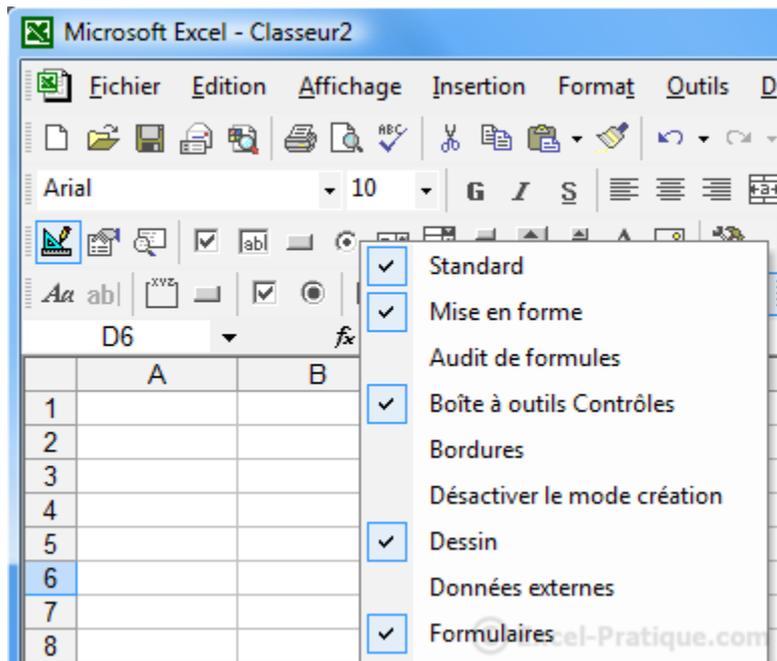
Si vous utilisez la version 2007 d'Excel (ou une version supérieure), cliquez sur Fichier > Options > Personnaliser le Ruban puis cochez "Développeur".



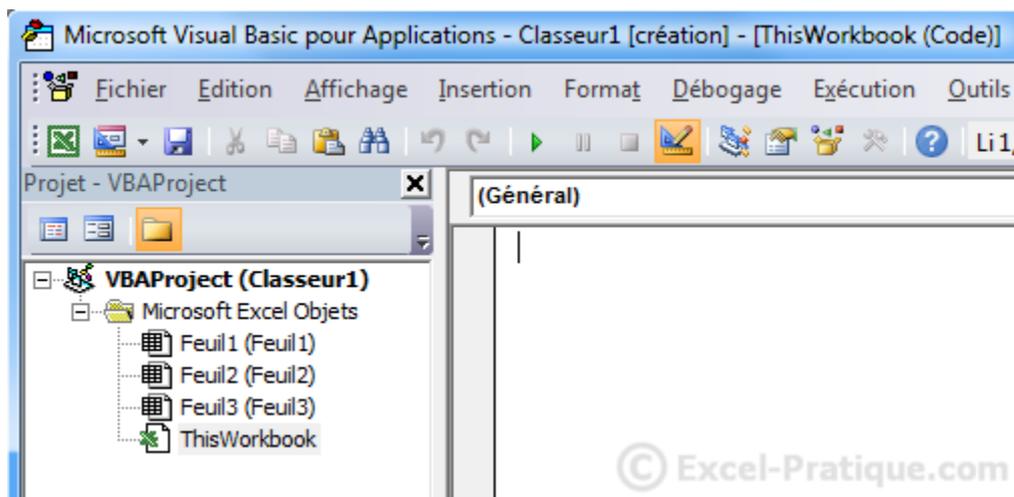
Un nouvel onglet sera ajouté :



Si vous utilisez une version d'Excel antérieure à 2007, ajoutez les barres "Boîtes à outils Contrôles" et "Formulaires" :



Pour travailler avec du code VBA, nous avons besoin d'un éditeur, celui-ci est déjà installé et vous pouvez l'ouvrir avec le raccourci "Alt F11" :

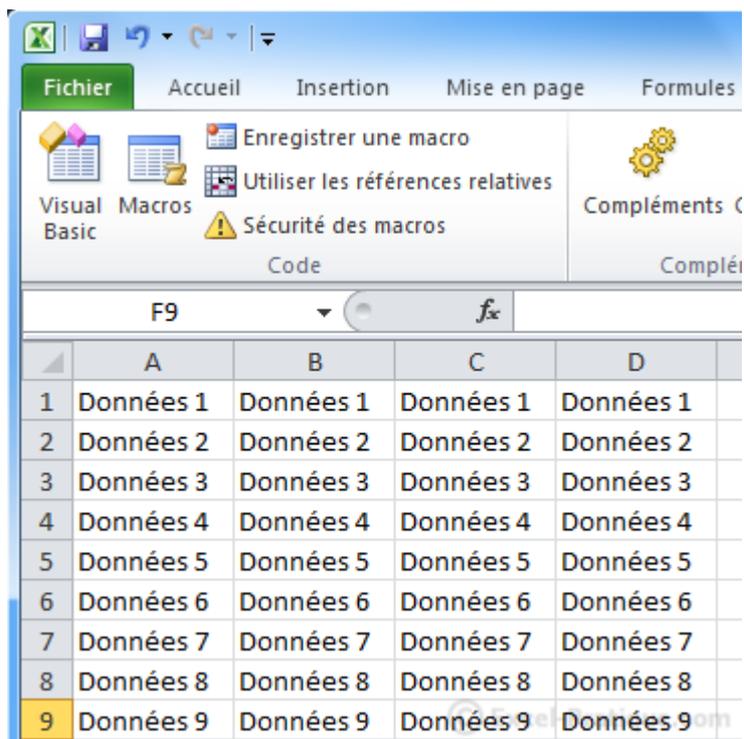


Cours VBA : première macro

Il est possible d'automatiser certaines tâches en toute simplicité grâce à l'enregistreur de macros.

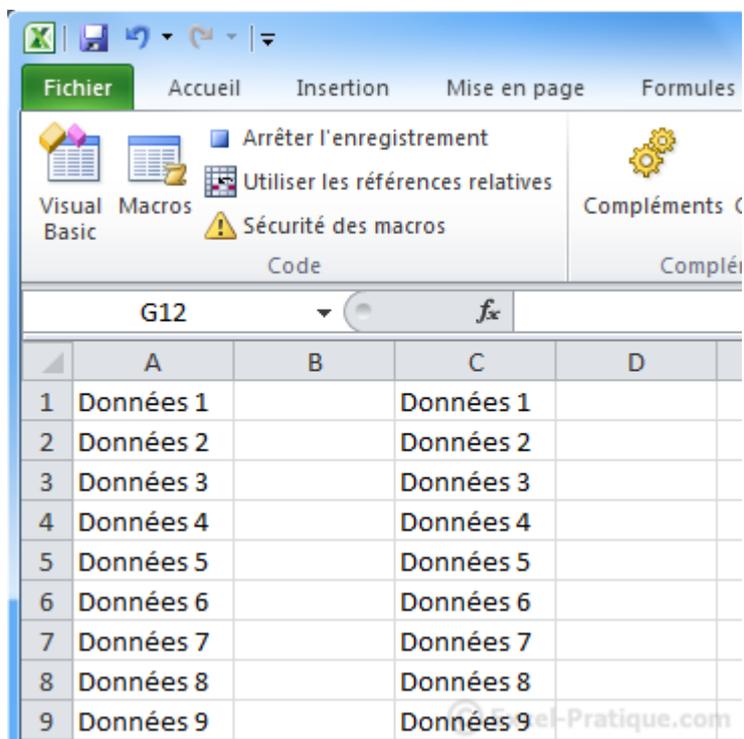
Pour prendre un exemple simple, nous allons automatiser les opérations suivantes :

- supprimer le contenu des colonnes A et C
- déplacer le contenu de la colonne B dans la colonne A
- déplacer le contenu de la colonne D dans la colonne C



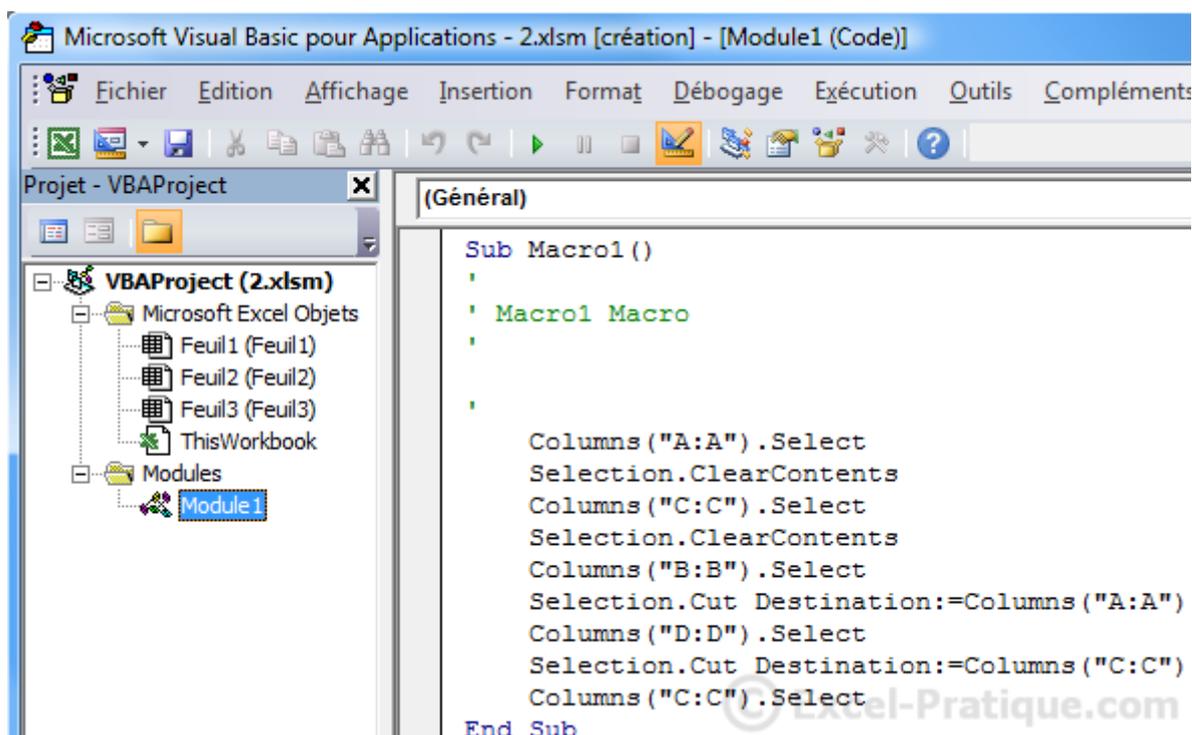
Pour ce faire, cliquez sur "Enregistrer une macro" puis "Ok", exécutez les opérations décrites ci-dessus sans interruption (car toutes les manipulations sont enregistrées) puis cliquez sur "Arrêter l'enregistrement".

Pour les versions d'Excel inférieures à 2007 : Outils > Macros > Nouvelle macro.



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

Pour voir votre macro, ouvrez l'éditeur (Alt F11) et cliquez sur "Module1" :



Ce code 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 modifier le nom de cette macro par quelque chose de plus parlant, remplacez simplement "Macro1" par "manipulations_des_colonnes" (le nom ne doit pas contenir d'espaces) :

```
Sub manipulations_des_colonnes()
```

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 a beaucoup de code ou pour ne pas exécuter certaines lignes de code sans pour autant les supprimer.

```
Sub manipulations_des_colonnes()  
'
```

```
' Mon premier commentaire !  
'
```

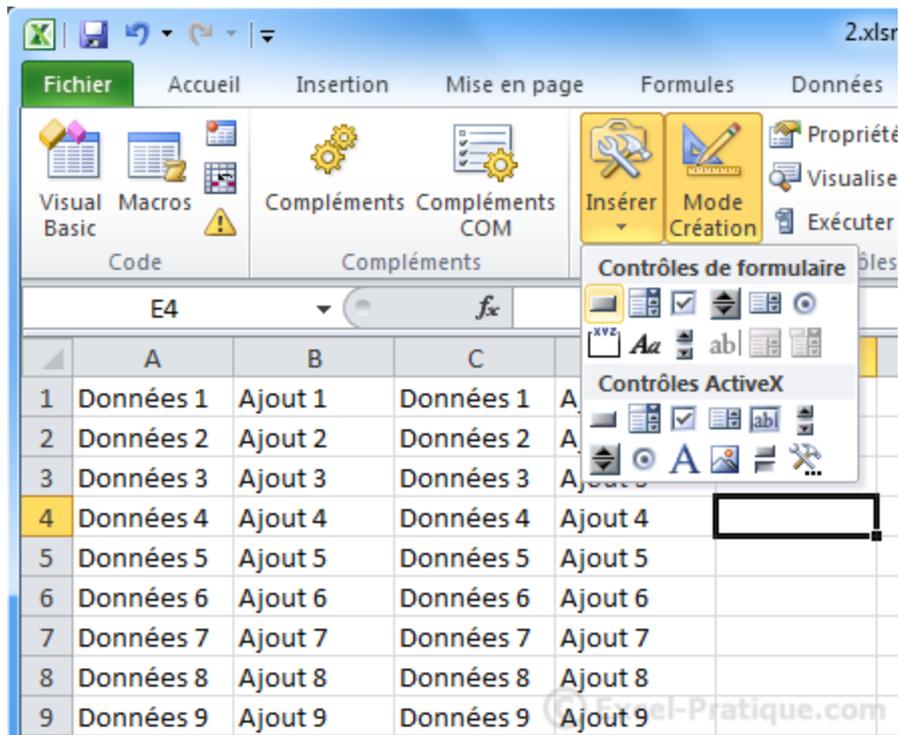
```
Columns("A:A").Select  
Selection.ClearContents  
Columns("C:C").Select  
Selection.ClearContents  
Columns("B:B").Select  
Selection.Cut Destination:=Columns("A:A")  
Columns("D:D").Select  
Selection.Cut Destination:=Columns("C:C")  
Columns("C:C").Select
```

```
End Sub
```

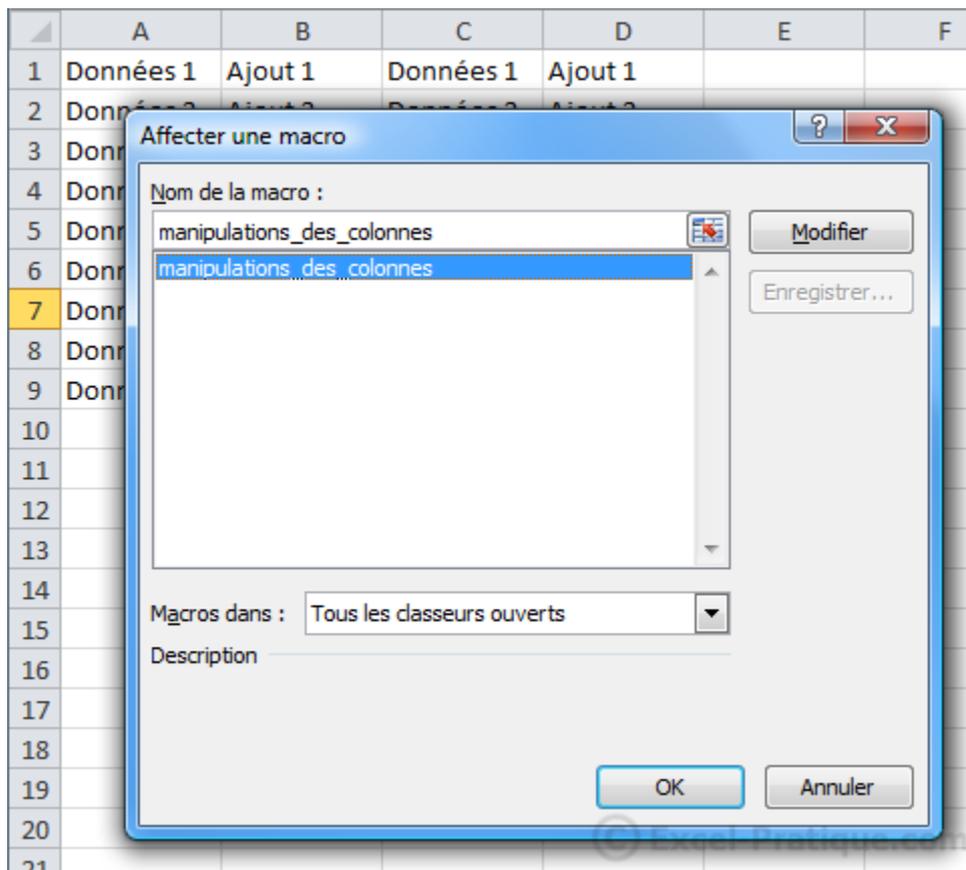
Maintenant, nous voulons que cette macro s'exécute en cliquant sur un bouton.

Cliquez sur Insérer > Bouton (Contrôles de formulaires) :

Pour les versions d'Excel inférieures à 2007 : "Bouton" de la barre "Formulaires".



Tracez votre bouton et sélectionnez ensuite simplement votre macro :



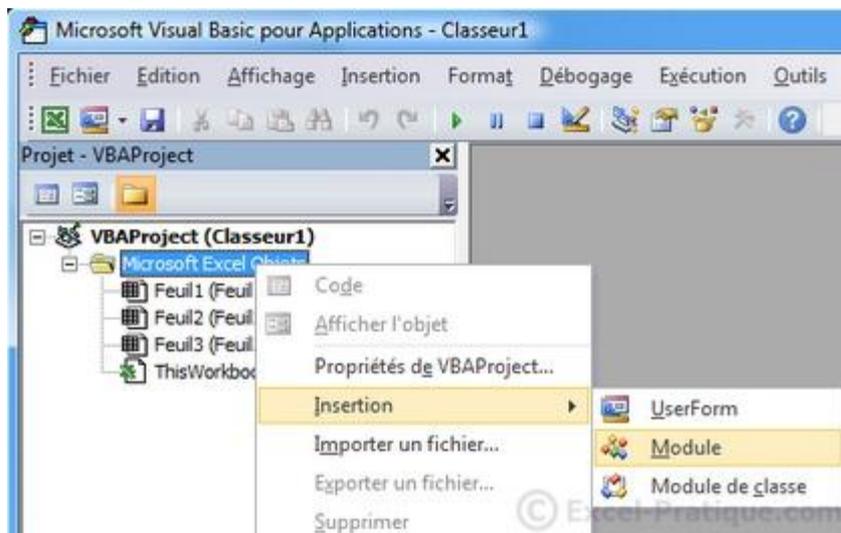
Lorsque vous cliquez sur le bouton, la macro sera exécutée :

	A	B	C	D	E
1	Ajout 1		Ajout 1		
2	Ajout 2		Ajout 2		
3	Ajout 3		Ajout 3		
4	Ajout 4		Ajout 4		
5	Ajout 5		Ajout 5		
6	Ajout 6		Ajout 6		Bouton 2
7	Ajout 7		Ajout 7		
8	Ajout 8		Ajout 8		
9	Ajout 9		Ajout 9		
10					
11					

Cours VBA : les sélections

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

Ouvrez l'éditeur et ajoutez-y un module :



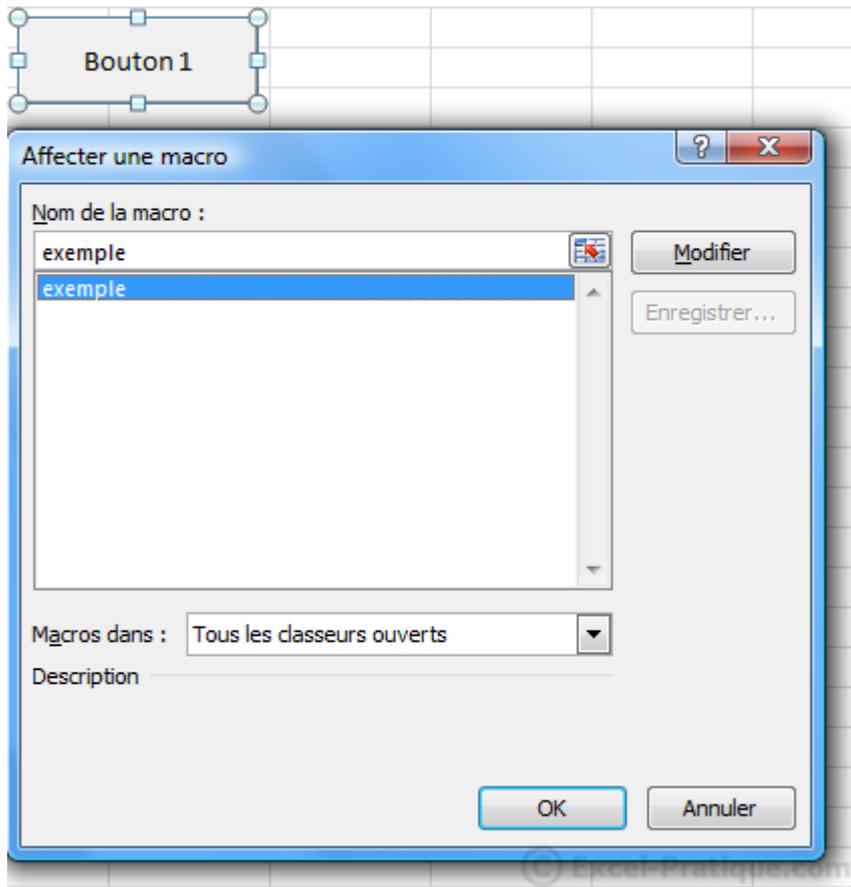
Dans le module, tapez "sub exemple" et appuyez sur Enter.

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) :



Complétez votre macro avec ceci :

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

Vous pouvez tester cette macro en cliquant sur votre 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
```

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

Remarque : aidez-vous des commentaires (texte en vert) pour bien comprendre les macros de ce cours.

Sélection de cellules distinctes

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

Sélection d'une plage de cellules

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

Sélection d'une plage de cellules renommée

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

	A	B
1		
2		
3		
4		
5		
6		
7		
8		

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).

Décaler une sélection

```

Sub exemple ()
'Sélection d'une cellule (calculée par rapport à la cellule active actuelle)
ActiveCell.Offset(2, 1).Select
End Sub

```

Décalage de 2 lignes vers le bas et 1 colonne vers la droite à partir de la cellule active, puis sélection :

	A2		fx	cellule
	A	B	C	
1				
2	cellule active avant clic			
3	+2 ↓			
4		+1 →	sélection après clic	
5				

© Excel-Pratique.com

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
```

Cours VBA : les propriétés

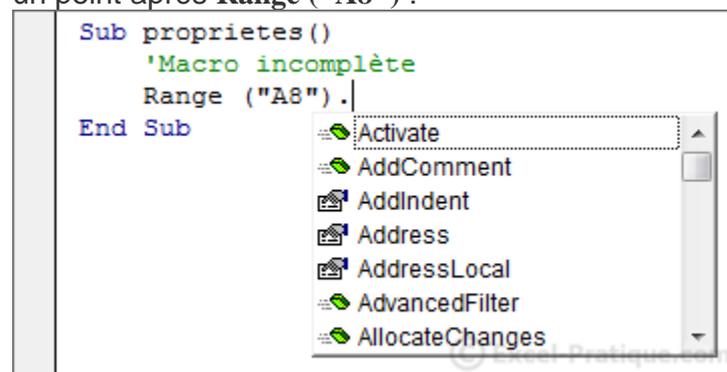
Nous allons maintenant agir sur le contenu et l'apparence des cellules et des feuilles.

Commencez par ouvrir l'éditeur, ajoutez-y un module, copiez la macro ci-dessous et associez-la à un bouton formulaire (*voir page "Sélections" en cas de besoin*) :

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

Nous voulons effectuer une action sur la cellule A8 avec ce début de macro.

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



L'éditeur affiche alors les différentes possibilités ...

Pour ce premier exemple, cliquez sur "Value" puis sur la touche Tab pour valider ce choix.

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

La propriété Value est ici le contenu de la cellule.

Nous voulons maintenant donner la valeur 48 à A8 :

```
Sub proprietes()  
    'A8 = 48  
    Range("A8").Value = 48  
    'Traduction :  
    'La valeur de la cellule A8 est égale à 48  
End Sub
```

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

```
Sub proprietes()  
    '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, grâce au bouton formulaire) qui sera modifiée. 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 ajouter avant Range : **Sheets("Nom_de_la_feuille")** ou **Sheets(Numéro_de_la_feuille)**.

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

De même, si l'on souhaite modifier la cellule A8 de la feuille 2 d'un autre classeur ouvert, il faut ajouter avant Sheets et Range : **Workbooks("Nom_du_fichier")**.

```
Sub proprietes()  
    '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.

Ces 2 lignes offrent un résultat identique :

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

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

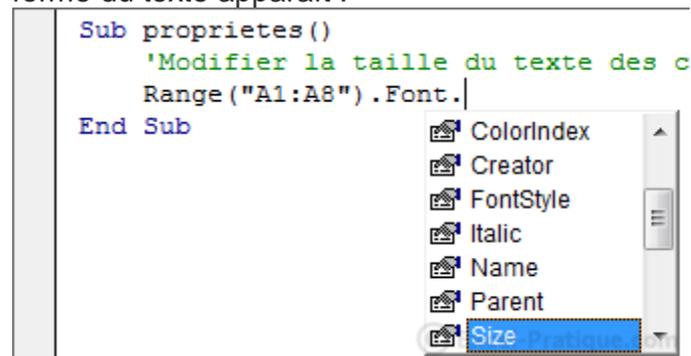
Effacer le contenu de cellules



```
Sub proprietes()  
    'Effacer le contenu de la colonne A  
    Range("A:A").ClearContents  
End Sub
```

Mise en forme du texte

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



La modification des couleurs sera détaillée à la page suivante ...

Mise en forme : taille du texte

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

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

Pour supprimer la mise en forme "caractères gras" 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

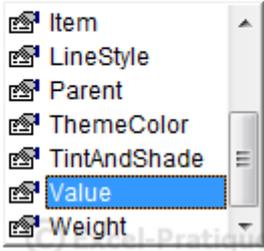
```
Sub proprietes()  
    'Souligner les cellules A1 à A8  
    Range("A1:A8").Font.Underline = True  
End Sub
```

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.  
End Sub
```



The screenshot shows a dropdown menu for the 'Borders' property of a range. The menu items are: Item, LineStyle, Parent, ThemeColor, TintAndShade, Value (highlighted in blue), and Weight. A watermark 'Excel-Pratique.com' is visible at the bottom right of the menu.

```
Sub proprietes()  
    'Ajouter une bordure aux cellules A1 à A8  
    Range("A1:A8").Borders.Value = 1  
    'Value = 0 => pas de bordure  
End Sub
```

Modifier la mise en forme de la sélection actuelle

```
Sub proprietes()  
    'Ajouter une bordure aux cellules sélectionnées  
    Selection.Borders.Value = 1  
End Sub
```

Modifier les propriétés d'une feuille

```
Sub proprietes()  
    'Masquer une feuille  
    Sheets("Feuil3").Visible = 0  
    'Visible = -1 => annule l'effet  
End Sub
```

N'oubliez pas que seule une toute petite minorité des possibilités de personnalisation sont indiquées ici. Si la propriété dont vous avez besoin n'est pas détaillée ici, n'ayez pas peur de la rechercher grâce à la liste de choix et l'aide Excel. L'enregistreur de macro peut également vous éviter de longues recherches. En enregistrant la manipulation dont vous avez besoin, vous pourrez retrouver plus facilement la propriété recherchée pour ensuite l'utiliser dans votre macro.

Modifier la valeur d'une cellule en fonction d'une autre

Nous voulons ici que A7 prenne la valeur de A1 :

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

Excel-Pratique.com

Nous allons donc demander 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
```

Si nous voulons copier la taille du texte uniquement, le code serait :

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

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

Modifier la valeur d'une cellule en fonction de sa propre valeur

Nous allons 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
```

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

Pour exemple : avant l'exécution du code, A1 vaut 0

```
Sub proprietes()  
  
    'Un clic a été fait sur le bouton, nous entrons dans la procédure  
    'Pour le moment A1 vaut encore 0  
  
    'PENDANT l'exécution de la ligne ci-dessous A1 vaut toujours 0  
    Range("A1") = Range("A1") + 1 'Le calcul est alors : Nouvelle_valeur_de_A1 = 0 + 1  
    'A1 vaut alors 1 seulement APRES l'exécution de la ligne de code  
  
End Sub
```

With

Ce code permet de définir différentes propriétés à la cellule active :

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

Nous pouvons utiliser **With** pour éviter les répétitions d'**ActiveCell** dans le cas présent.

Voici comment **With** fonctionne :

```
Sub proprietes()  
    'Début de l'instruction avec : WITH  
    With ActiveCell  
        .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
```

ActiveCell n'est donc plus répété.

Bien que ce ne soit pas indispensable dans ce cas, il est également possible d'éviter les répétitions de **.Font**, ce qui nous donnerait :

```
Sub proprietes()  
    With ActiveCell  
        .Borders.Weight = 3  
        With .Font  
            .Bold = True  
            .Size = 18  
        End With  
    End With
```

```

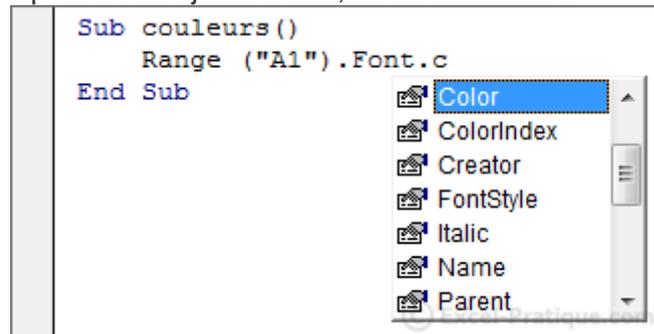
        .Italic = True
        .Name = "Arial"
    End With
End With
End Sub

```

Cours VBA : les couleurs

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	I
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 donner à notre texte l'une des 56 couleurs, nous écrivons :

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

Ce qui nous donne :

	A	B	C
1	Test		
2			
3		Bouton 1	
4			

Pour les versions d'Excel inférieures à 2007 : l'utilisation de `ColorIndex` est préférable à `Color`.

Color

Voici un exemple similaire avec `Color` :

```
Sub couleurs()  
    'Couleur du texte en A1 : RGB(50, 200, 100)  
    Range("A1").Font.Color = RGB(50, 200, 100)  
End Sub
```

La couleur ici est : **RGB(50, 200, 100)**

RGB en français signifie Rouge-Vert-Bleu (RVB), 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 la page suivante : [liste de valeurs RGB](#)

Pour donner une couleur violette à notre texte, nous pouvons donc rechercher les valeurs RGB de cette couleur sur la liste et écrire :

```

Sub couleurs()
    'Couleur du texte en A1 : RGB(192, 32, 255)
    Range("A1").Font.Color = RGB(192, 32, 255)
End Sub

```

Ce qui nous donne :

	A	B	C
1	Test		
2			
3		Bouton 1	
4			

Pour les versions d'Excel inférieures à 2007 : le nombre de couleurs est limité (la couleur disponible la plus proche de la valeur RGB sera utilisée).

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()
    'Épaisseur 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	E
1					
2					
3				Bouton 1	
4					

Pour appliquer cet effet à plusieurs cellules à la fois, nous pouvons utiliser **Selection** :

```

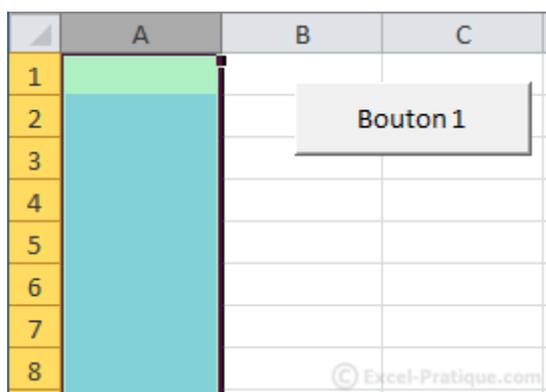
Sub couleurs()
    'Épaisseur de la bordure
    Selection.Borders.Weight = 4
    'Couleur de la bordure : rouge
    Selection.Borders.Color = RGB(255, 0, 0)
End Sub

```

Colorer le fond des cellules sélectionnées

```
Sub couleurs()  
    'Colorer le fond des cellules sélectionnées  
    Selection.Interior.Color = RGB(174, 240, 194)  
End Sub
```

Aperçu :



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 :



Cours VBA : les variables

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 ma_variable As Integer  
    'Attribution d'une valeur à la variable  
    ma_variable = 12
```

```
'Affichage de la valeur de ma_variable dans une MsgBox
```

```
MsgBox ma_variable
```

```
End Sub
```

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

```
Dim ma_variable As Integer
```

- **Dim** : déclaration de la variable
- **ma_variable** : 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 à résoudre plus facilement les problèmes, etc. Bref, mieux vaut prendre l'habitude de déclarer ses variables correctement.

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

Une valeur est ensuite donnée à cette variable :

```
ma_variable = 12
```

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

```
MsgBox ma_variable
```

MsgBox "valeur" affiche une valeur dans une boîte de dialogue de la façon la plus simple.

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
Byte	<i>Numérique</i>	Nombre entier de 0 à 255.
Integer	<i>Numérique</i>	Nombre entier de -32'768 à 32'767.
Long	<i>Numérique</i>	Nombre entier de - 2'147'483'648 à 2'147'483'647.
Currency	<i>Numérique</i>	Nombre à décimale fixe de -922'337'203'685'477.5808 à 922'337'203'685'477.5808.
Single	<i>Numérique</i>	Nombre à virgule flottante de -3.402823E38 à 3.402823E38.
Double	<i>Numérique</i>	Nombre à virgule flottante de -1.79769313486232D308 à 1.79769313486232D308.
String	<i>Texte</i>	Texte.
Date	<i>Date</i>	Date et heure.
Boolean	<i>Boolean</i>	True (vrai) ou False (faux).
Object	<i>Objet</i>	Objet Microsoft.
Variant	<i>Tous</i>	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 = "06.02.2011"
```

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.

Remarque : 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

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.

Fichier source : [exercice_variables.xls](#)

Nous 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
```

Nous attribuons ensuite les valeurs aux variables avec **Cells** :

```
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, nous affichons le résultat dans la boîte de dialogue en concaténant les valeurs avec & (comme sous 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				
5	Smith				
6	Smith				
7	Smith				
8	Smith				
9	Smith				
10	Smith				
11	Smith				



Nous allons maintenant rechercher à afficher dans la boîte de dialogue la ligne du tableau correspondant au numéro indiqué dans la cellule F5.

Voici l'objectif :

.
. .
. .
. .
. .
. .

La solution :

```
Sub variables()  
  'Déclaration des variables  
  Dim nom As String, prenom As String, age As Integer,  
numero_ligne As Integer  
  
  'Valeurs des variables  
  numero_ligne = Range("F5") + 1  
  nom = Cells(numero_ligne, 1)  
  prenom = Cells(numero_ligne, 2)  
  age = Cells(numero_ligne, 3)  
  
  'Boîte de dialogue  
  MsgBox nom & " " & prenom & ", " & age & " ans"  
End Sub
```

Ajout d'une variable :

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

La variable **numero_ligne** prend ensuite la valeur de la cellule F5 à laquelle nous ajoutons 1 (pour ne pas tenir compte de la première ligne qui contient les titres du tableau) ainsi, **numero_ligne** aura pour valeur le n° de ligne des cellules qui nous intéressent :

```
numero_ligne = Range("F5") + 1
```

Il ne reste plus qu'à remplacer les n° de ligne des **Cells** par notre variable :

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

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

	A	B	C	D	E	F
1	Nom	Prénom	Age			
2	Smith	John	40		Bouton	
3	Smith	John	41			
4	Smith	John	42			
5	Smith	John	43		N°:	2
6	Smith	John				
7	Smith	John				
8	Smith	John				
9	Smith	John				
10	Smith	John				
11	Smith	John				
12	Smith	John				
13	Smith	John				



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

```

Sub variables()
MsgBox Cells(Range("F5")+1,1) & " " & Cells(Range("F5")+1,2) & ", " &
Cells(Range("F5")+1,3) & " ans"
End Sub

```

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