

Certificat Informatique et Internet

Feuille 6

VBA : première partie

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.

Site Web du cours : <http://www.polytech.unice.fr/~hrenard/?page=enseignement#c2i>

1 Introduction

Ce cours pose les jalons de tout ce qui va suivre et nous donne une idée globale de ce qu'est la programmation VBA.

1.1 Que peut-on faire avec le VBA ?

Vous n'apprenez rien de nouveau lorsque je vous dirai qu'Excel sert à d'innombrables tâches dont voici un modeste aperçu :

- Gérer un budget et simuler des prévisions ;
- Analyser des données scientifiques ;
- Editer des factures et des formulaires ;
- Créer des graphiques à partir d'ensembles de données ;
- Gérer des listes de clients, de résultats scolaires ou d'idées de cadeaux, etc.

Tous les lecteurs ont en commun la nécessité d'automatiser certaines fonctions d'Excel. C'est là que VBA entre en jeu !

Vous pourriez par exemple créer un programme VBA qui imprime le rapport des ventes du mois. Après avoir développé et testé le programme, vous exécuterez la macro d'une seule commande qui effectuera à votre place les longues procédures.

1.1.1 Insérer une chaîne de caractères

Si vous devez systématiquement entrer le nom d'une société et/ou ses coordonnées dans une feuille de calcul, une macro pourra le faire à votre place. Vous pouvez étendre ce concept beaucoup plus loin. Par exemple, vous développerez une macro qui entre automatiquement les noms de tous les représentants travaillant pour la société.

1.1.2 Automatiser les tâches répétitives

Supposons que, responsable de ventes, vous deviez rédiger chaque mois le rapport qui calmera les angoisses de votre patron. Si cette tâche est toujours la même, vous le confierez à un programme VBA. Votre patron sera impressionné par la présentation toujours régulière de vos rapports. Un jour ou l'autre, il finira bien par vous proposer un poste plus gratifiant (on peut toujours rêver !)

1.1.3 Exécuter des actions à répétition

Si vous devez appliquer une même action à, disons, une bonne douzaine de classeurs Excel différents, vous avez intérêt à enregistrer une macro lorsque vous effectuez cette action la première fois, puis à laisser la macro la répéter sur les onze autres classeurs. Excel ne se plaindra jamais de ces ennuyeuses répétitions de tâches. L'enregistrement d'une macro n'est pas plus difficile qu'un enregistrement sonore, le micro en moins.

1.1.4 Créer une commande personnalisée

Vous avez souvent recours aux mêmes commandes d'un menu Excel ? Vous gagnerez un peu de temps en développant une macro qui les réunit toutes en une seule commande personnalisée, que vous lancerez d'une seule touche ou d'un seul clic sur un bouton.

1.1.5 Créer des boutons de barres d'outils personnalisés

Les barres d'outils d'Excel peuvent être personnalisées avec vos propres boutons qui exécuteront les macros que vous avez écrites. L'employé de bureau moyen est toujours très impressionné par ce genre de chose.

1.1.6 Créer des boutons personnalisés

La barre d'outils Accès rapide d'Excel peut être personnalisée avec vos propres boutons qui exécuteront les macros que vous avez écrites. Là, l'employé de bureau moyen tombe en pâmoison.

1.1.7 Développer des nouvelles fonctions de calcul

Bien qu'Excel soit généreusement pourvu de fonctions prédéfinies, vous pouvez créer vos propres fonctions personnalisées qui simplifieront considérablement vos formules. Vous serez étonné de constater combien c'est facile. Mieux encore, vos fonctions personnalisées apparaîtront dans la boîte de dialogue Coller une fonction, comme si elles avaient toujours fait partie d'Excel. Là, c'est le SAMU qu'il faut appeler pour l'employé de bureau !

1.1.8 Créer une application complète entièrement régie par la macro

Si vous disposez d'un peu de temps, vous pourrez recourir au VBA pour créer des applications complètes de grande envergure, avec des boîtes de dialogue, une aide en ligne et toutes sortes de gadgets (mortel pour l'employé de bureau).

1.1.9 Créer des macros complémentaires pour Excel

Vous connaissez probablement certaines macros complémentaires livrées avec Excel, comme l'Utilitaire d'analyse. Le langage VBA vous permettra de développer les vôtres.

1.2 Avantages et inconvénients du VBA

1.2.1 Les avantages du VBA

Quasiment tout ce qu'il est possible de faire dans Excel peut être automatiser. Il suffit pour cela d'écrire les instructions qu'Excel devra exécuter. L'automatisation des tâches présente de nombreux avantages :

- Excel exécute toujours les tâches de la même manière (dans la plupart des cas, cette régularité est une bonne chose).
- Excel exécute les tâches plus rapidement que vous ne le feriez manuellement.
- Si vous savez bien programmer les macros, Excel exécute toujours les tâches sans erreur.
- Les tâches peuvent être démarrées par quelqu'un qui n'y connaît rien à Excel.

- Il est possible de demander à Excel d'exécuter les tâches autrement impossibles à mettre en œuvre.
- Lorsque la tâche est longue et demande beaucoup de temps, vous n'avez plus à vous morfondre devant l'ordinateur. Allez plutôt faire un brin de causerie près de la machine à café.

1.2.2 Les inconvénients du VBA

Toute médaille ayant son revers, il est honnête que nous nous attardions sur les inconvénients réels ou potentiels du VBA :

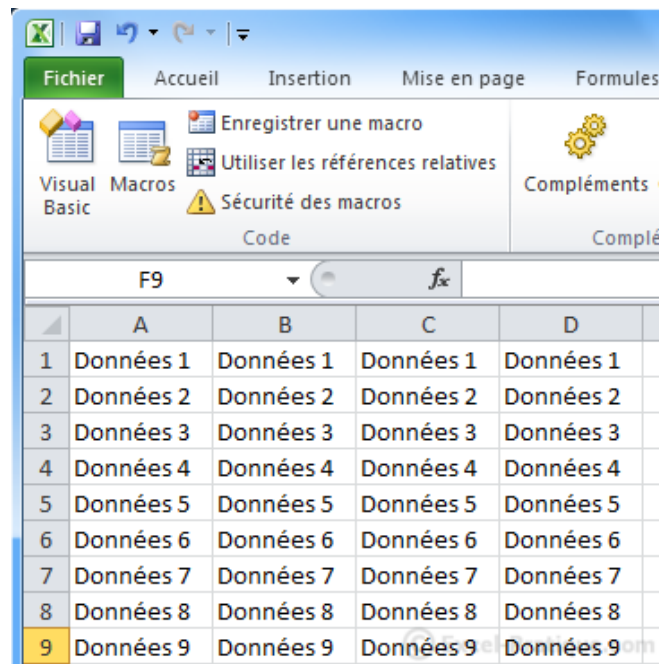
- Vous devez apprendre la programmation VBA.
- Les personnes qui désireraient utiliser vos programmes VBA doivent posséder Excel.
- Parfois, les choses tournent mal. Autrement dit, vous n'aurez jamais la certitude que votre programme fonctionnera dans tous les cas de figure. Bienvenue dans le monde enchanté de débogage.
- Le VBA n'est pas pérenne

2 Première macro

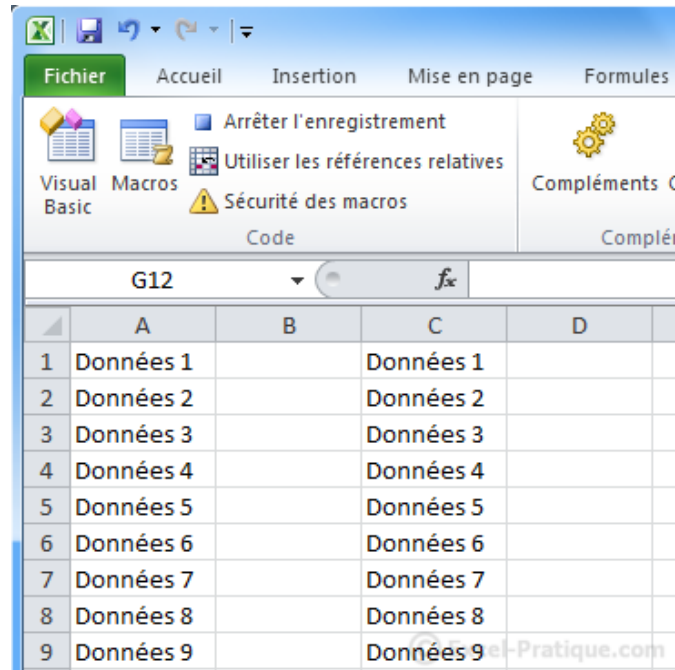
Il est possible d'automatiser certaines tâches en toute simplicité grâce à l'enregistreur de macros.

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

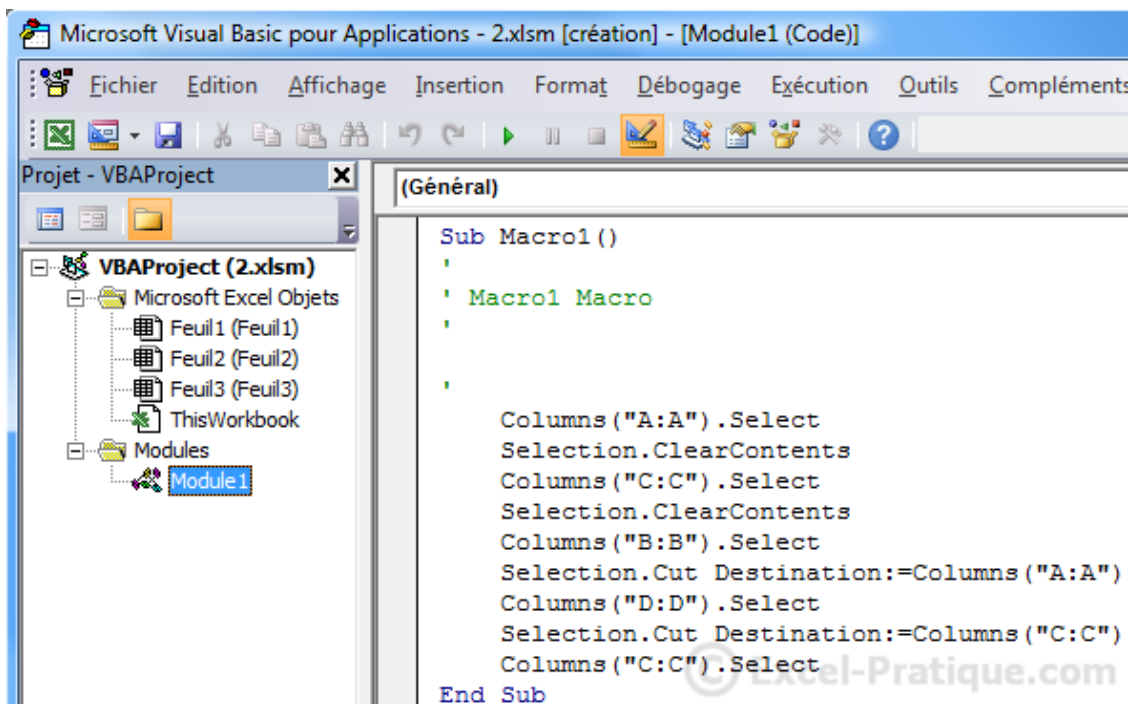


Pour ce faire, cliquez sur "Enregistrer une macro" puis "Ok", exécutez les opérations décrites ci-dessus sans interruption (car toutes les manipulations sont enregistrées) puis cliquez sur "Arrêter l'enregistrement".



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
```

BAT 3 / 2016-2017
H. RENARD-FERRY

```
Columns("C:C").Select
Selection.ClearContents

Columns("B:B").Select

Selection.Cut Destination:=Columns("A:A")

Columns("D:D").Select

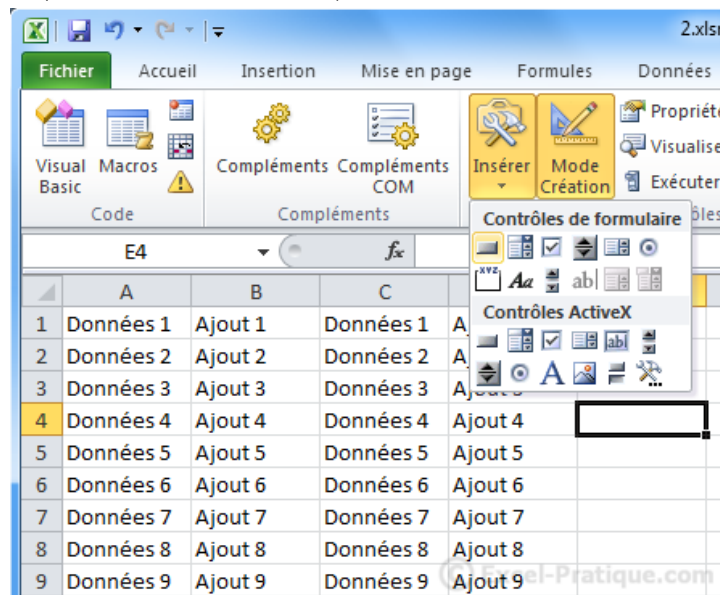
Selection.Cut Destination:=Columns("C:C")

Columns("C:C").Select

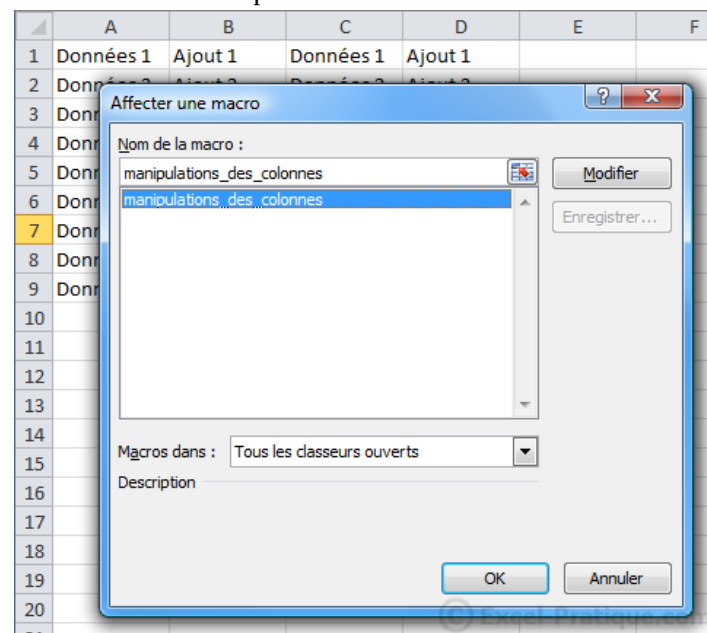
End Sub
```

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

Cliquez sur Insérer > Bouton (Contrôles de formulaires) :



Tracez votre bouton et sélectionnez ensuite simplement votre macro :



Lorsque vous cliquerez 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		
7	Ajout 7		Ajout 7		
8	Ajout 8		Ajout 8		
9	Ajout 9		Ajout 9		
10					
11					

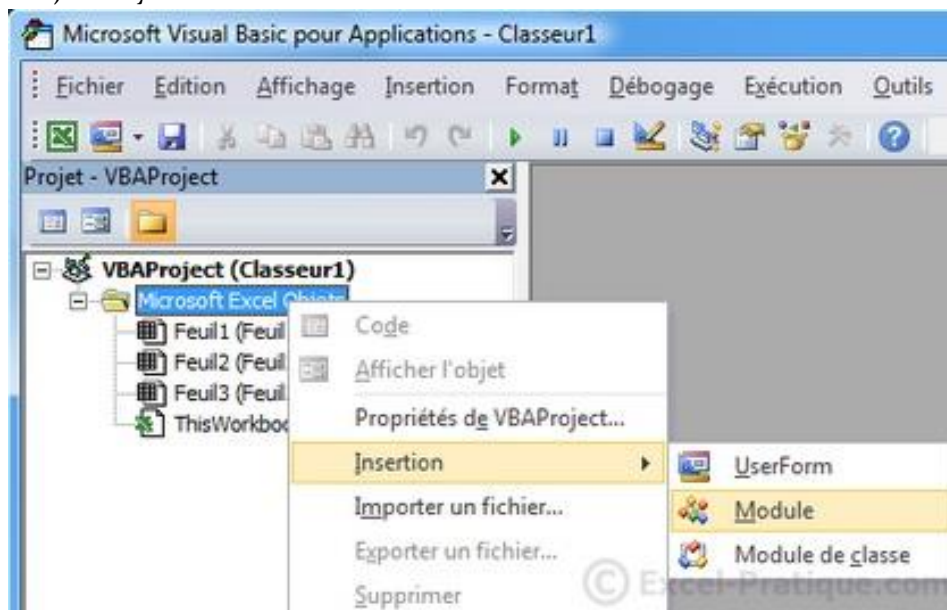
Bouton 2

© Excel-Pratique.com

3 Les feuilles et cellules

3.1 Sélections

Nous allons commencer par créer une macro qui sélectionnera une cellule de notre choix. Ouvrez l'éditeur et ajoutez-y un module :

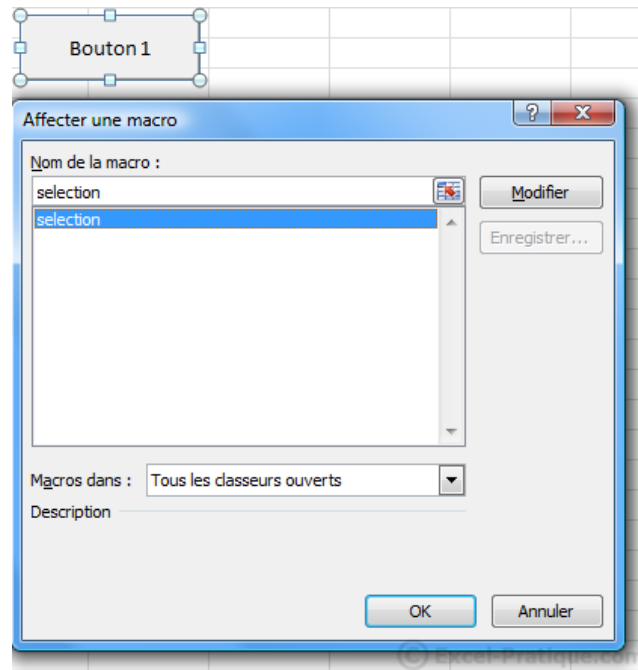


Dans le module, tapez "sub selection" et appuyez sur Enter.

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

```
Sub selection()  
  
End Sub
```

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



Complétez votre macro avec ceci :

```
Sub selection()
    '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 selection()
    '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.

3.1.1 Sélection de cellules distinctes

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

3.1.2 Sélection d'une plage de cellules

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

3.1.3 Sélection d'une plage de cellules renommée

```
Sub selection()
    'Sélection des cellules de la plage "ma_plage"
```

BAT 3 / 2016-2017
H. RENARD-FERRY

```
Range("ma_plage").Select
End Sub
```

ma_plage	
	B
1	
2	
3	
4	
5	
6	
7	
8	

3.1.4 Sélection d'une cellule en fonction d'un numéro de ligne et de colonne

```
Sub selection()
'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 selection()
'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 : $\text{Int}(\text{Rnd} * 10) + 1$, autrement dit : un nombre entre 1 et 10 (inutile de retenir ce code pour le moment).

3.1.5 Décaler une sélection

```
Sub selection()
'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			

BAT 3 / 2016-2017
H. RENARD-FERRY

3.1.6 Sélection de lignes

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

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

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

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

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

3.2 Propriétés

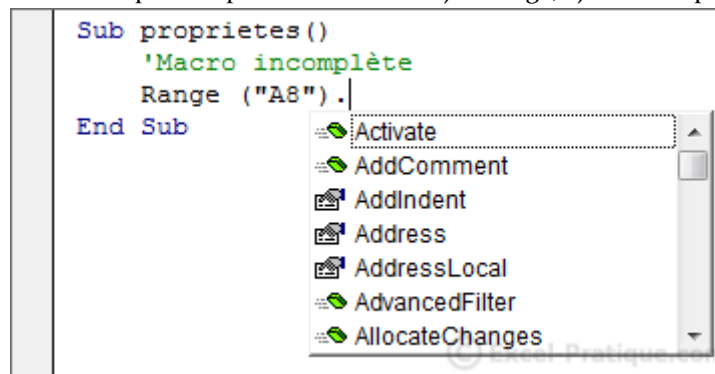
Nous allons maintenant agir sur le contenu et l'apparence des cellules et des feuilles.

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.

BAT 3 / 2016-2017
H. RENARD-FERRY

```
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").

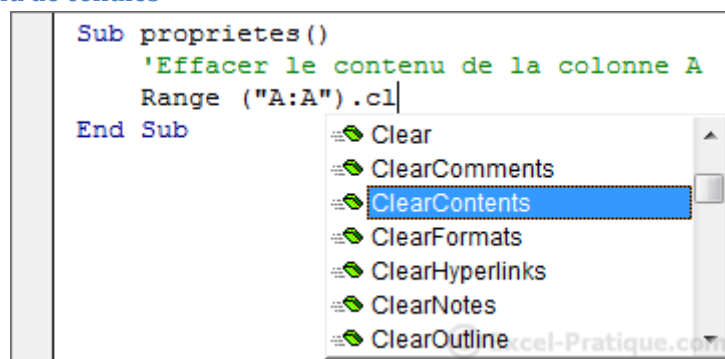
```
Sub proprietes()
    'A8 de la feuille 2 du classeur 2 = Exemple de texte
    Workbooks("Classeur2.xlsx").Sheets("Feuil2").Range("A8").Value = "Exemple de texte"
End Sub
```

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

3.2.1 Effacer le contenu de cellules

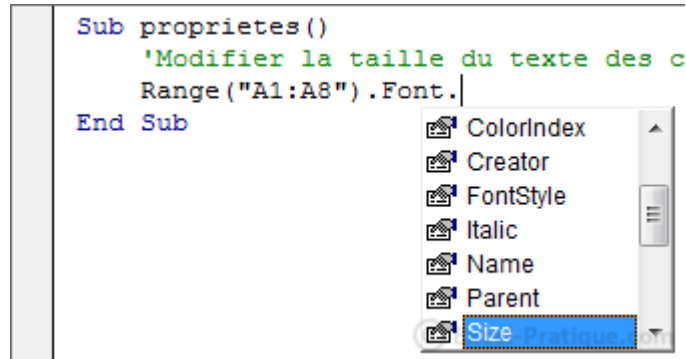


BAT 3 / 2016-2017
H. RENARD-FERRY

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

3.2.2 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 ...

3.2.3 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
```

3.2.4 Mise en forme : texte en gras

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

Font.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
```

3.2.5 Mise en forme : texte en italique

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

3.2.6 Mise en forme : texte souligné

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

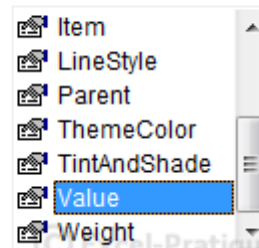
BAT 3 / 2016-2017
H. RENARD-FERRY

3.2.7 Mise en forme : police

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

3.2.8 Ajouter des bordures

```
Sub proprietes()
    'Ajouter une bordure aux cellules A1 à A8
    Range("A1:A8").Borders.
End Sub
```



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

3.2.9 Modifier la mise en forme de la sélection actuelle

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

3.2.10 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.

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

Nous voulons ici que A7 prenne la valeur de A1 :

BAT 3 / 2016-2017
H. RENARD-FERRY

	A	B	C
1	Ma valeur		
2			
3			
4		Bouton 1	
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 =.

3.2.12 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
```

3.2.13 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
```

BAT 3 / 2016-2017

H. RENARD-FERRY

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.

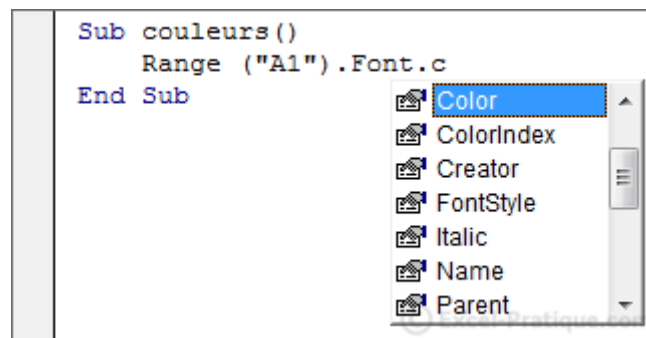
Bien que ce ne soit pas indispensable dans ce cas, il est également possible d'éviter les répétitions de .Font, ce qui nous donnerait :

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

3.3 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.

3.3.1 ColorIndex

Voici les 56 couleurs disponibles avec ColorIndex :

	A	B	C	D	E	F	G	H	I
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			
4			

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

3.3.2 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.

BAT 3 / 2016-2017
H. RENARD-FERRY

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			

3.3.3 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	E
1					
2				Bouton 1	
3					
4					

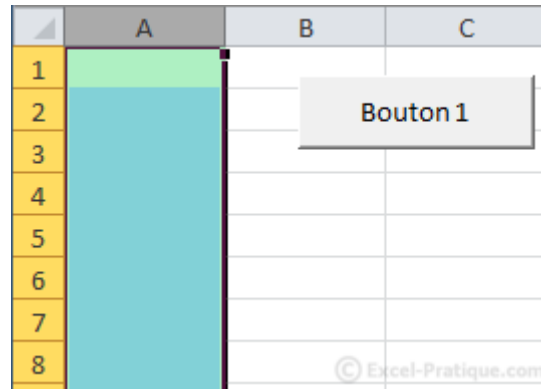
Pour appliquer cet effet à plusieurs cellules à la fois, nous pouvons utiliser Selection :

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

3.3.4 Colorer le fond des cellules sélectionnées

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

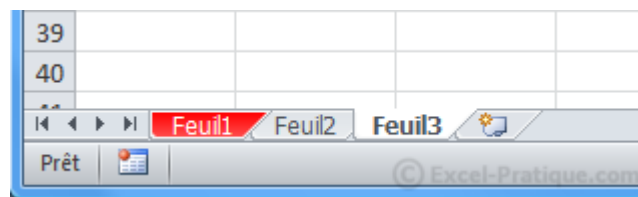
Aperçu :



3.3.5 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 :



4 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.

BAT 3 / 2016-2017
 H. RENARD-FERRY

Le type de la variable indique la nature de son contenu (texte, nombres, date, etc.).
 Une valeur est ensuite donnée à cette variable :

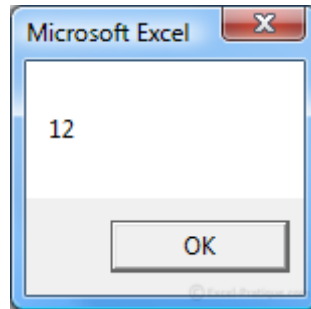
```
ma_variable = 12
```

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.

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 ...

4.1 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 -1.79769313486232D308 à 1.79769313486232D308.	#
String	Texte	Texte.	\$
Date	Date	Date et heure.	
Boolean	Boolean	True (vrai) ou False (faux).	
Object	Objet	Objet Microsoft.	
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
```

BAT 3 / 2016-2017

H. RENARD-FERRY

```
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).

4.2 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)
```

BAT 3 / 2016-2017

H. RENARD-FERRY

```
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 :

	A	B	C	D	E	F
1	Nom	Prénom	Age			
2	Smith	John	40			
3	Smith	John	41			
4	Smith	John	42			
5	Smith	John	43		N° :	5
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			

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

·
·
·
·
·
·
·

BAT 3 / 2016-2017
H. RENARD-FERRY

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).

4.3 Les tableaux

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

Voici quelques exemples de déclarations :

```
'Exemple de déclaration de variable
Dim var1 As String

'Exemple de déclaration de tableau à 1 dimension
Dim tab1(4) As String

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

'Exemple de déclaration de tableau à 3 dimensions
Dim tab3(4, 3, 2) As String
```

Le tableau à 1 dimension :

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

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 la dernière case du tableau. tab1(4) est un tableau dont les cases vont de 0 à 4, il s'agit donc d'un tableau de 5 cases :

0	'Attribution de valeurs aux 5 cases
1	tab1(0) = "Valeur de la case 0"
2	tab1(1) = "Valeur de la case 1"
3	tab1(2) = "Valeur de la case 2"
4	tab1(3) = "Valeur de la case 3"
	tab1(4) = "Valeur de la case 4"

Important : la première case d'un tableau est 0.

Autre exemple, le tableau à 2 dimensions :

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

0-0	0-1	0-2	0-3	'Attribution de valeurs aux 3 cases colorées tab2(0, 0) = "Valeur de la case rouge" tab2(4, 1) = "Valeur de la case verte" tab2(2, 3) = "Valeur de la case bleue"
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	

4.4 Les constantes

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

Par exemple, nous pouvons ajouter une constante pour éviter les répétitions de 6.87236476641 :

BAT 3 / 2016-2017
 H. RENARD-FERRY

```
Sub exemple_const()
  Cells(1, 1) = Cells(1, 2) * 6.87236476641
  Cells(2, 1) = Cells(2, 2) * 6.87236476641
  Cells(3, 1) = Cells(3, 2) * 6.87236476641
  Cells(4, 1) = Cells(4, 2) * 6.87236476641
  Cells(5, 1) = Cells(5, 2) * 6.87236476641
End Sub
```

Cela facilite la lecture du code (en particulier pour des codes importants) et facilite le changement (manuel) de la valeur de la constante en cas de besoin :

```
Sub exemple_const()
  'Déclaration de la constante + attribution de sa valeur
  Const TAUX_ANNUEL As Double = 6.87236476641

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

4.5 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 Global :

```
Global 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
```

BAT 3 / 2016-2017
H. RENARD-FERRY

4.6 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 invites
    nom As String
    prenom As String
End Type

Sub variables ()
    'Déclaration
    Dim p1 As invites

    'Attributions des valeurs à p1
    p1.nom = "Smith"
    p1.prenom = "John"

    'Exemple d'utilisation
    MsgBox p1.nom & " " & p1.prenom
End Sub
```

5 Exercices

5.1 Exercice n°1

Enregistrez une macro nommée Absolu dans le classeur o6-VBA-1.xlsx. Avant de commencer l'enregistrement, cliquez dans la cellule A3. Pendant l'enregistrement

- Tapez une valeur dans cette cellule
- Validez
- Cliquez dans la cellule A1 et tapez une autre valeur

Testez

5.2 Exercice n°2

Enregistrer une macro nommée Relatif dans le classeur o6-VBA-1.xlsx. Même exercice que le précédent mais cette fois, activez le bouton relatif avant de commencer les actions.

Testez

5.3 Exercice n°3

Enregistrement d'une macro nommée GrasItalique dans le classeur o6-VBA-1.xlsx. Pendant l'enregistrement

- Utiliser Format/cellule ...

Regardez le code

Nettoyez l'inutile

5.4 Exercice n°4

- Affectez un raccourci clavier à la macro « Absolu »
- Créez un bouton sur la feuille et affectez-lui la macro « relatif »
- Affectez la macro « GrasItalique » à un nouveau bouton dans la barre d'outils ou dans le menu

5.5 Exercice n°5

Écrivez une procédure qui :

BAT 3 / 2016-2017
H. RENARD-FERRY

- affiche une boîte de dialogue Bonjour avec un bouton unique et le titre Message VBA

5.6 Exercice n°6

Dans la procédure ci-dessous, trouvez l'erreur :

```
Sub ChercherErreur
    MsgBox Voici un nouveau message
End sub
```

5.7 Exercice n°7

Que donnera cette procédure comme résultat ?

```
Sub SurfaceCarré()
    côté = 10
    MsgBox côté * côté
End Sub
```

5.8 Exercice n°8

Créez une procédure qui

- Demandra à l'utilisateur de taper un nombre compris entre 1 et 10 (bornes comprises)
- Lorsque l'utilisateur validera son entrée, le programme doit afficher une boîte de dialogue affichant le nombre tapé augmenté de l'entrée précédente.

Exemple :

1ere utilisation de la procédure, l'utilisateur tape 10, le programme affiche 10

2e utilisation de la procédure, l'utilisateur tape 25, le programme affiche 35 (25+10)

et ainsi de suite

5.9 Exercice n°9 (affectation d'une valeur à une propriété)

Créez une procédure qui :

- Affichera la valeur 12 dans la cellule active.

5.10 Exercice n°10 (Récupération de la valeur d'une propriété)

Créez une procédure qui :

- Affichera dans une boîte de message la valeur contenue dans la cellule active

5.11 Exercice n°11 (Syntaxe de l'utilisation d'une méthode)

Créez une procédure qui fera dans l'ordre les actions suivantes :

- Mettre la valeur 'Hello World' dans la cellule active
- Afficher dans une boîte de message le contenu de la cellule active
- Effacer le contenu de la cellule active (la méthode)
- Pour vérifier, afficher de nouveau dans une boîte de message, le contenu de la cellule active