A. Introduction

Comme nous l'avons vu dans le chapitre précédent, l'Enregistreur de macros convertit chacune des sélections de cellule, de plage de cellules, **en références fixes** :

```
Range("A3:B5").Select
Range("C8").Select
Columns("D").Select
```

Écrire un code de cette manière est à proscrire car les références resteront invariablement les mêmes. Or, l'intérêt de faire un programme c'est d'appliquer une instruction à un élément (une cellule, une feuille de calcul, un classeur, un graphique...) puis de reproduire cette instruction à d'autres éléments.

Voilà pourquoi, il ne faut jamais écrire directement les références d'une cellule directement dans le code mais il faut utiliser une **variable** qui va, comme son nom l'indique, permettre de faire varier les coordonnées de la cellule.

B. Qu'est-ce qu'une variable ?

Une variable peut être vue comme une boîte dans laquelle vous allez stocker une information : un chiffre, une date, du texte...

Alors bien sûr, il ne s'agit pas d'une boîte au sens littéral du terme mais d'un espace réservé dans la mémoire de l'ordinateur. Comment l'ordinateur gère cet espace en mémoire, là n'est pas la question. Tout ce qu'il faut retenir c'est qu'en créant une variable (une boîte) avec un nom qui lui sera propre, le langage VBA saura toujours dans quelle boîte il faudra lire ou récupérer une donnée. En fait, il suffit de déclarer un nom de variable et de lui attribuer un type de donnée et Visual Basic s'occupera du reste.

Il n'y a pas vraiment de règles pour le nom que vous donnez à vos variables ; vous pouvez choisir n'importe quel mot à condition qu'il ne fasse pas partie des mots réservés (mots-clés) de VBA (comme Sheets, Cells, Worksheets, Columns, Rows...).

Une variable peut être composée d'un ou plusieurs mots mais ils ne doivent pas être séparés par le caractère **espace** ni par un **tiret (-)**. Le seul caractère de séparation autorisé est l'**underscore (_)**.

*Pour vos variables, il est préférable de privilégier des noms compréhensibles (**Numero_Ligne**, **Nom_Classeur**, **Nom_Feuille**...) plutôt que de choisir des noms composés d'une seule lettre (i, j, k...) comme c'est souvent malheureusement le cas.*

C. La méthode MsgBox

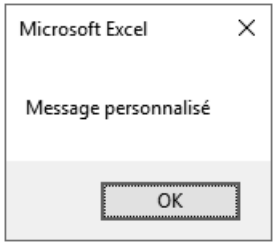
La méthode `MsgBox` est une fonctionnalité importante dans la compréhension d'un programme en Visual Basic car elle va permettre d'afficher à tout moment un message personnalisé. Non seulement `MsgBox` affiche un message personnalisé mais il bloque également l'exécution de la suite du code.

1. Affichage d'un message personnalisé

Par exemple, écrivez le programme suivant :

```
Sub Test_Affichage()
   MsgBox "Message personnalisé"
End Sub
```

Et lancez l'exécution de ce programme en appuyant sur la touche F5 ou en cliquant sur l'icône  . Le résultat de ce programme affiche la boîte de dialogue suivante :



Comme vous le constatez, l'instruction pour afficher un message est très simple à réaliser mais ce qu'il faut surtout comprendre c'est que tant que vous n'appuyez pas sur le bouton **OK**, l'exécution du code reste bloquée sur la ligne `MsgBox`.

Maintenant, il est possible d'afficher plus qu'un message personnalisé : il est aussi possible de regrouper en même temps du texte personnalisé avec le contenu d'une variable.

2. Afficher le résultat d'une variable

Tout texte saisi entre guillemets s'affichera tel quel dans la boîte de dialogue. Mais une variable ne doit jamais être écrite entre guillemets. Si tel était le cas, la boîte de dialogue afficherait le nom utilisé dans le programme pour qualifier la variable et non pas son contenu.

```
MsgBox "Ma_Variable"  'Affiche juste le mot Ma_Variable comme message.
```

Alors que l'instruction suivante affichera la valeur de la variable dans la boîte de dialogue.

```
MsgBox "Ma_Variable a pour valeur " & Ma_Variable
```

Selon le contenu de la variable (texte, nombre, date...), l'instruction précédente affichera à la fois le texte écrit entre guillemets et le contenu de la variable. Nous pouvons par exemple écrire les messages suivants :

- Ma_Variable a pour valeur 35
- Ma_Variable a pour valeur Paris
- Ma_Variable a pour valeur 25/12/2016

Nous verrons dans la section suivante les différents types de variables que vous pouvez utiliser.

Vous pouvez remarquer le symbole & entre le texte entre guillemets et le nom de la variable. Nous verrons plus en détails dans le chapitre Gérer les chaînes de caractères des techniques propres aux chaînes de caractères (et il y en a beaucoup), mais retenez à ce stade que le symbole & permet de lier du texte personnalisé avec le contenu d'une variable.

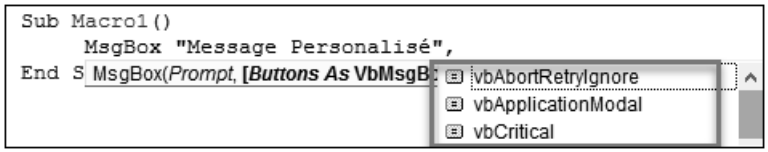
3. Personnaliser le message

Vous pouvez personnaliser tous les paramètres de la boîte de dialogue `MsgBox`, titre de la fenêtre, nombre de boutons, icônes. De cette façon, vos utilisateurs auront la perception que votre message est propre à votre programme et n'est pas un message d'Excel.

La méthode **MsgBox**, outre le message à faire passer, possède les options suivantes :

```
MsgBox Message, Bouton, Titre
```

L'option **Bouton** vous permet de changer le nombre de boutons qui s'affichent (comme **Annuler**, **Oui**, **Non**...) et aussi de changer l'icône qui s'affiche sur la gauche de la boîte de dialogue. Pour cela, il faut choisir l'une des constantes qui s'affiche après avoir saisi une virgule après votre message personnalisé.

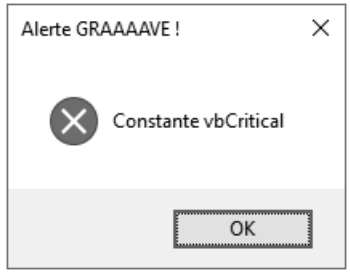


En troisième paramètre, vous pouvez également mettre un titre personnalisé.

Ainsi, en écrivant la ligne suivante :

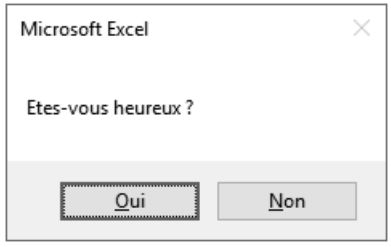
```
MsgBox "Constante vbCritical", vbCritical, "Alerte GRAAAAVE !"
```

vous affichez le message suivant :



Maintenant, si vous choisissez la constante `vbYesNo`, la boîte de dialogue va afficher les boutons **Oui** et **Non**, mais l'écriture de la ligne sera légèrement différente car il faut indiquer que la réponse de l'utilisateur (**Oui** ou **Non**) sera mise dans une variable.

```
Sub Macro1()
Dim LaReponse As Long
     LaReponse = MsgBox("Etes-vous heureux ?", vbYesNo)
End Sub
```



Nous étudierons plus tard comment traiter la réponse dans le chapitre Les conditions et aussi comment connaître la valeur de la réponse dans le chapitre Débogage.

D. Déclaration d'une variable

1. Déclaration obligatoire ou pas ?

Visual Basic permet de travailler avec des variables sans les déclarer. Cependant, un langage informatique quel qu'il soit, ne peut pas en même temps travailler avec des données numériques et des données textuelles. En effet, si vous essayez d'additionner des chiffres et du texte, votre programme s'arrêtera net et occasionnera un plantage, c'est-à-dire un arrêt total du programme sans terminer son traitement.

Déclarer vos variables vous prémunit d'erreurs de manipulation des données dans un programme.

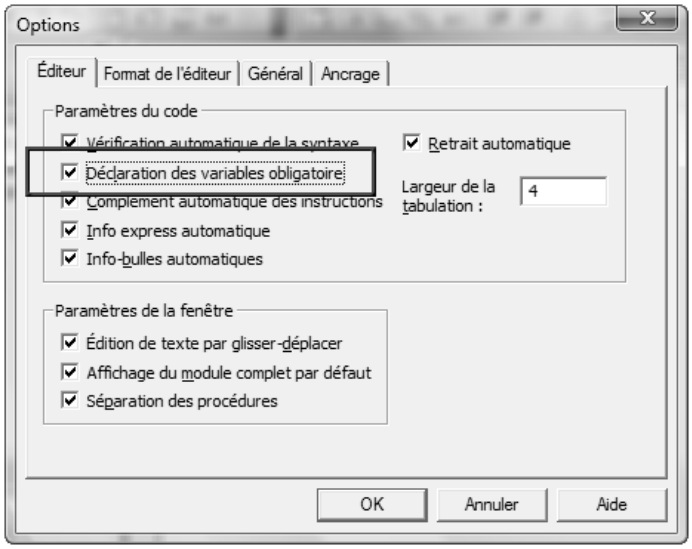
Dans tous les exemples qui suivent, les variables seront systématiquement déclarées.

De plus, quand une variable est déclarée, il est très facile de rappeler le nom de vos variables grâce à la combinaison de touches Ctrl Espace.

Il suffit de saisir les premiers caractères du nom d'une variable déclarée puis d'utiliser le raccourci Ctrl Espace pour que le nom complet de la variable soit complété instantanément.

Vous pouvez rendre la saisie des variables obligatoire en modifiant une option dans les paramètres de Visual Basic Editor.

- Ouvrez le menu **Outils - Options**.
- Dans l'onglet **Éditeur**, cochez l'option **Déclaration des variables obligatoire**.



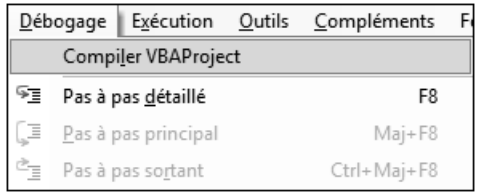
✎ Cliquez sur **OK**.

Maintenant, l'ajout de tout nouveau module commencera systématiquement par l'instruction `Option Explicit` qui impose la déclaration de toutes vos variables.

2. Vérification des déclarations

Quand vous êtes en mode `Explicit`, c'est-à-dire que vous travaillez en déclarant toutes vos variables, vous ne pourrez pas lancer votre programme si une seule variable n'a pas été déclarée. En effet le programme sera dans l'incapacité de comprendre comment interpréter cette information.

Ceci peut paraître très bloquant mais il existe un moyen simple de vérifier si toutes vos variables ont bien été déclarées avant de lancer votre programme. Il vous suffit d'activer le menu **Débogage**, puis le sous-menu **Compiler VBAProject**.



L'outil va parcourir tout votre projet (toutes les procédures et/ou toutes fonctions) et s'il détecte qu'une variable est utilisée sans avoir été déclarée, il vous le signale par un message d'erreur.