Présentation de VBA pour Excel

Excel permet d'automatiser des **traitements**, d'ajouter de **nouvelles fonctions** pour répondre à vos besoins spécifiques, de **personnaliser votre espace de travail** et de permettre à Excel de **communiquer avec d'autres applications Microsoft et Internet**. Cela vous permettra de développer des **applications personnalisées** de gestion, de reporting ou d'analyse en utilisant le **langage VBA**. Ce langage VBA permet **d'exploiter** toutes les potentialités du tableur Excel.

L'objectif de ce cours est de :

- se **familiariser** avec le code VBA ;
- d'exploiter le code VBA pour **manipuler les objets d'Excel** (classeurs, feuilles de calcul, plages de cellules...) afin notamment d'automatiser des traitements.

Présentation du langage VBA

Visual Basic pour Application (**VBA**) est le langage de programmation commun à toutes les applications de la suite Microsoft Office (Word, Access, Excel, Outlook et PowerPoint).

Quelques définitions

Le projet

Quand vous ouvrez un **classeur dans Excel**, un **projet** est associé à ce classeur et va contenir tous les **modules de codes VBA** regroupés par catégories.

Les modules

Les modules contiennent les **macros** enregistrées et vos propres **procédures** et **fonctions** écrites en VBA. Ces modules pourront être **exportés** sous forme de fichiers indépendants afin d'être **réutilisés** en les important dans d'autres classeurs. De cette manière il est possible de réutiliser du code VBA existant sans devoir tout réécrire.

Excel - VBA : exporter / importer un module VBA

Les procédures

Un ensemble de code VBA doit être regroupé dans des procédures. Ce sont donc des **mini programmes** écrits en VBA. Quand vous enregistrez une macro, une procédure est générée avec comme nom le nom de la macro. Pour créer vos propres procédures, vous utiliserez l'instruction **Sub**.

Les fonctions

Les fonctions sont des procédures particulières car elles retourne un résultat, une valeur. Pour créer une fonction vous utiliserez l'instruction Function.

Création des procédures

Il y a deux manières de créer des procédures VBA :

- générer automatiquement du code VBA à partir de l'enregistreur de macros ;
- saisir directement le code de votre procédure dans l'environnement Visual Basic Editor (ou environnement VBE).

La création de macros, bien que très utile et facile, a cependant des limites. Les macros ne peuvent permettre que d'automatiser des actions répétitives sous Excel.

Pour avoir des traitements plus évolués, il est nécessaire de créer ses propres procédures.

Il est bien sur possible de créer des macros et d'améliorer, d'enrichir le code VBA obtenu.

Pour écrire et exécuter des macros, exécuter des macros enregistrées ou créer des applications Excel, vous devez afficher l'onglet Développeur de la façon suivante :

- Cliquez sur l'onglet **Fichier**, puis sur **Options** ;
- Sélectionnez la catégorie **Personnaliser le Ruban** ;
- Sous **Personnaliser le Ruban**, dans la liste **Onglets principaux**, sélectionnez la case à cocher **Développeur** ;
- Cliquez sur le bouton **OK** ; l'onglet **Développeur** a été ajouté au ruban d'Excel, à droite de l'onglet **Affichage**.

Pour à l'éditeur de Visual Basic, Visual Basic Editor (VBE), cliquez sur l'**icône Visual Basic** ou utilisez le raccourci-clavier **[Alt][F11]**.

Le Groupe « Code »

Ce groupe permet :

- d'ouvrir **l'environnement de développement** Visual Basic (VBA) ;
- **d'enregistrer** des macros en utilisant les références **absolues** ou les références **relatives** à la première cellule sélectionnée ;
- **d'exécuter** des macros.

L'environnement de développement Visual Basic vous permet de visualiser le code VBA de vos macros.
Voici l'exemple de deux macros utilisant pour déplacer une plage de cellules :

- soit l'option référence relative - procédure RefRelative()
  * soit l'option référence absolue - procédure RefAbsolue()

Cliquez sur le bouton de l'onglet Développeur pour accéder au code d'une macro.

```
Sub RefAbsolue()
'  Référence absolue
   Range("B2:C8").Select
   Selection.Cut Destination:=Range("C4:D10")
   Range("C4:D10").Select
End Sub

Sub RefRelative()
'  Référence relative
   ActiveCell.Range("A1:B7").Select
   Selection.Cut Destination:=ActiveCell.Offset(2, 1).Range("A1:B7")
   ActiveCell.Offset(2, 1).Range("A1:B7").Select
End Sub
```

Il y a une troisième manière d'accéder à une cellule en utilisant la fonction **Cells(i,j)**.

L'indentation du code

Quand vous écrivez du code VBA, il est conseillé de mettre en évidence les blocs d'instruction en utilisant **l'indentation** :

- d'une **procédure** ou d'une **fonction**,
- d'un bloc logique de **prise de décision (alternative)** ou de **structures répétitives**.

Cela rend le code est **plus lisible** et **plus facile à corriger** en cas d'erreur.

Pour **indenter** une ligne ou plusieurs lignes, appuyez :

- sur la touche **[Tab]** pour une tabulation,
- sur les touches **[Shift][Tab]** pour effectuer un retrait.

Vous pouvez préciser la largeur de la tabulation par le menu Outil - Options.

Les macros et la sécurité

Les paramètres de sécurité des macros permettent de contrôler ce qui se produit lorsque vous ouvrez un classeur contenant une macro. Le bouton de l'onglet **Développeur** permet de de définir les paramètres de sécurité.

Gardez l'option **Désactiver toutes les macros avec notification** afin de pouvoir décider d'activer ou pas les macros d'un classeur.

Retour au menu de Excel & VBA

Excel et VBA (Visual Basic pour Application)