TP1 Découverte Macros et VBA (1h30) sous WPS

L'objectif de ce TP est de découvrir l'environnement de programmation Visual Basic (VBE). Nous allons également établir des programmes en Visual Basic pour application (VBA). Nous commençons par la création, la modification et l'exécution d'une macro avant d'aller plus loin dans la programmation.

Ressources complémentaires dans Moodle :

https://www.excel-pratique.com/fr/vba
A noter que la syntaxe VBA sous WPS est la même que sous Excel.

Définitions

Macro :

Macro est le diminutif de macro-commande qui désigne à l'origine une commande regroupant plusieurs commandes. Les macros sont très utiles pour exécuter de façon répétitive les mêmes suites d'opérations. Par la suite, on a pris l'habitude d'appeler macro tout programme introduit dans un logiciel applicatif pour en étendre les commandes.

Macro sous WPS:

Pour programmer des macros sous WPS, on utilise le langage VBA (Visual Basic pour Applications). C'est un langage très proche de Visual-Basic et commun à un grand nombre d'applications en particulier celles du pack Office comme Word ou Access.

Partie 1 : Création d'une macro commande par enregistrement

1. Nous allons travailler sous l'onglet « DEVELOPPEUR » qui est affiché par défaut dans WPS Speadsheets

Les opérations liées à la programmation sont principalement incluses dans l'onglet Développeur. Si les commandes sont grisées, il peut y avoir des soucis de droit.

Pour définir temporairement le niveau de sécurité de manière à activer toutes les macros :

1. Sous l'onglet **Développeur**, icône « sécurité », cliquez sur **Niveau de sécurité MOYEN**.
2. A l'ouverture d'un fichier contenant des macros un message d'avertissement s'affichera, veillez à cliquer sur OK (si vous connaissez l'origine des macros !!! en l'occurrence les fichiers de TPS d'EG de votre prof !)

1. Créer une macro

- Sous l'onglet **Développeur**, cliquez sur **Enregistrer une macro**. Tapez le nom de la macro (l'appeler « *MaPremiereMacro »* sans espace) dans la zone "Nom de la macro" puis cliquer sur **OK**.

A partir de là, tous ce que vous faites est enregistré. Faire ensuite les modifications suivantes :

1. Aller dans la cellule A1 et taper "Nombre"
2. Aller dans la cellule B1 et taper " Carré".
3. Aller dans la cellule B2 et écrire la formule "= A2* A2".
4. Sélectionner les cellules "A1:B1", les mettre en gras, les centrer et faire un remplissage en gris.

Sous l'onglet **Développeur**, cliquez sur **arrêter l'enregistrement**. Vous venez de créer votre première macro.

2. Exécuter la macro

1. Sélectionner une autre feuille ou prenez soin d'effacer vos précédentes actions
2. Aller dans **VB Macros**,
3. Sélectionner la macro que vous venez de créer *MaPremiereMacro* et cliquer sur **Exécuter.**

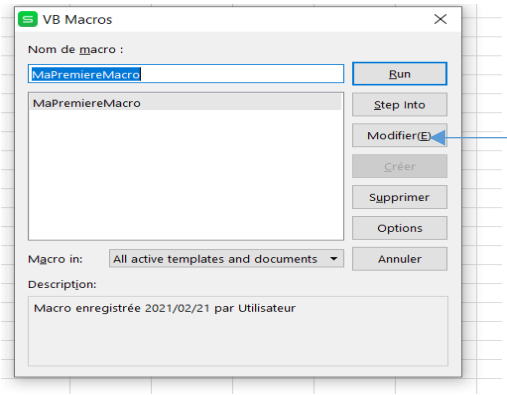
Lancer une macro via un bouton de contrôle

Pour lier une macro à un bouton de commande dans Excel il suffit de créer un bouton et de lui affecter une macro. Sous WPS ce n'est pas possible, il faut passer par l'éditeur VBA, nous verrons donc cela plus tard.

3. Visualiser le code de la macro

Pour visualiser le code VBA créé lors de l'enregistrement de votre macro *MaPremiereMacro,* procéder comme suit:

- Procéder de la même manière que pour afficher une macro (onglet DEVELOPPEUR /Macros) en sélectionnant la macro que vous venez de créer *MaPremiereMacro* et cliquer sur *Modifier*. Le programme de la macro est alors affiché par l'éditeur VBE.



Partie 2 : Création d'une macro commande par programmation

1. Généralités

Nous avons vu jusque-là comment créer des macros en enregistrant les opérations à réaliser. Les macros enregistrées sont simples à faire et ne nécessitent aucune connaissance en programmation. Cependant, elles peuvent s'avérer limitées dès qu'il s'agit d'utiliser des opérations complexes, faisant appel à des notions pointues de programmation. Dans ce cas, on doit écrire nous-mêmes le code des opérations sans passer par l'enregistreur de macros. Cette méthode suppose des connaissances minimales en algorithmique et en langage VBA.

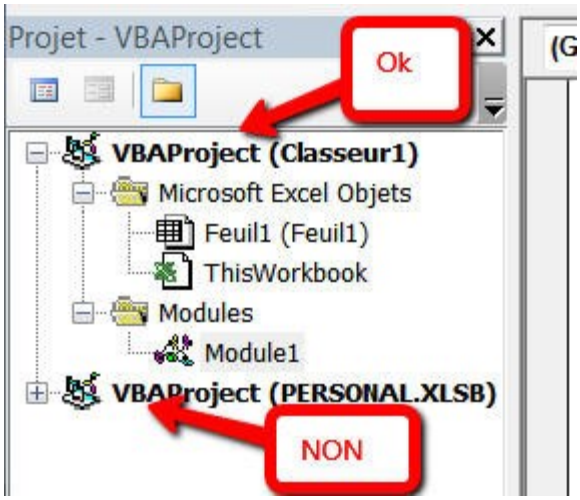
Vous ne devriez donc rencontrer aucune difficulté !!!

2. L'éditeur de code en VBA

L'explorateur de projets permet de naviguer dans les différents endroits pouvant contenir du code VBA Excel, c 'est à dire les différents emplacements où il vous est possible de sauvegarder le code que vous allez écrire. Vous le lancer en cliquant sur l'icône Visual Basic Editor. L'explorateur de fichier est le même que dans Excel (Vous aurez Sheets1 au lieu de Feuil1).

Vos choix:

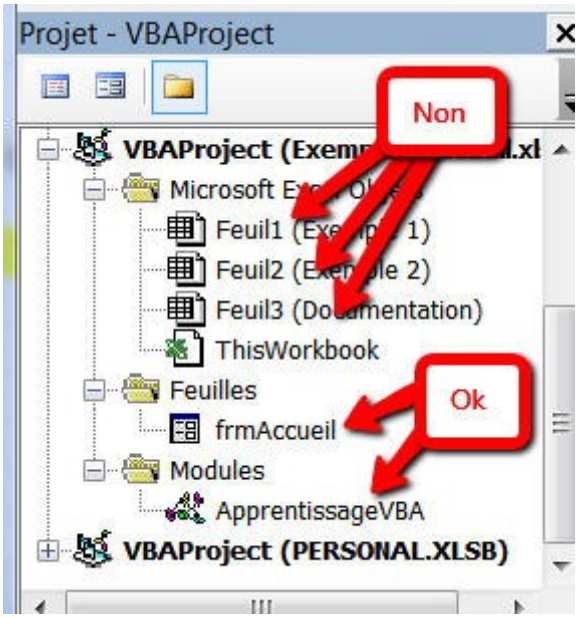
- Un classeur Excel: le code VBA est sauvegardé avec le classeur. Si vous copiez le classeur, il est copié avec. Si vous supprimez le classeur, il est supprimé.
- Le classeur de macros personnelles: le code VBA est sauvegardé dans le classeur personal.xlsb de l'ordinateur, **ET NON DANS LE CLASSEUR ACTIF**. Comme personal.xlsb est automatiquement chargé lors du démarrage d'Excel sur un ordinateur, les programmes VBA sauvegardés dans le classeur personal.xlsb de cet ordinateur sont disponibles dans tous les classeurs ouverts sur ce même ordinateur. Ce



choix est utile aux programmeurs expérimentés et aux administrateurs de parcs d'ordinateurs. **Il est fortement déconseillé aux programmeurs amateurs.**

Dans un classeur Excel, vous devez choisir un des endroits suivants:

- Une feuille de ce classeur. Cet emplacement est fortement déconseillé. Les programmes qui y sont cachés seront difficiles à retrouver plus tard.
- Le classeur lui-même (**ThisWorkbook**). Il est utilisé pour enregistrer les programmes déclenchés par des événements de classeur (ouverture, fermeture, modification de cellule...).
- Un module. Il est utilisé pour enregistrer les programmes utilisés dans le classeur ou par les autres programmes du classeur.
- Un formulaire. Il est utilisé pour enregistrer les programmes déclenchés par les événements du formulaire (sera vu dans le TP gestion des stocks)



Tous ces emplacements peuvent contenir du code VBA Excel.

En pratique, vous utiliserez le plus souvent les endroits suivants:

- Un module dans un classeur Excel.
- Un formulaire dans un classeur Excel.

3. Utiliser les fonctions d'entrées/sortie VBA

Afin d'utiliser les fonctions d'entrées/sorties dans des programmes VBA, vous devez d'abord lancer l'éditeur VBE). Pour ce faire, Aller dans Developpeur/Visual basic Editor. Il existe principalement deux fonctions qui peuvent être utilisées pour saisir/afficher des données par l'intermédiaire des boites de dialogue. Ces deux fonctions sont :

InputBox : *Cette méthode affiche une boîte de dialogue dans laquelle l'utilisateur peut saisir des données. Elle renvoie les informations que l'utilisateur a saisies dans la boîte de dialogue.*

Exemple : Pour demander à l'utilisateur de taper un nombre:

```
nombre =InputBox("Entrer un nombre")
```

MsgBox : *Affiche un message dans une boîte de dialogue, attend que l'utilisateur clique sur un bouton, puis peut renvoyer une valeur de type* ***Integer*** *qui indique le bouton choisi par l'utilisateur.*

NB : Plus tard nous verrons comment récupérer ou écrire des données directement depuis une feuille du classeur.

Exemple :

```
Reponse = MsgBox ("Souhaitez-vous continuer ? »,VbYesNo )
 If Reponse = vbYes Then                                                    '
            MyString = "Oui"
Else
      MyString = "Non"
End If
```

Exercice 1

1. Copier le code de « Que fais-je » ci-dessous dans l'onglet développeur
2. Corriger les erreurs de syntaxe en utilisant les rubriques d'aide de VBA : la touche **F1** du clavier.
3. Une fois les erreurs corrigées, exécuter le programme à l'aide du bouton ► de la barre d'outils de VBE.
4. Que fait ce programme?
5. Quelles sont les 2 structures de contrôle utilisées ? Quelle autre structure peut-on utiliser en VBA pour écrire une boucle ? Utiliser les ressources complémentaires pour répondre.
6. Quel est le rôle des variables *Compteur* et *Commentaire* dans le programme ?

```
Sub Que_fait_je()   ' début de la procédure

Dim Commentaire As String  ' pour déclarer la variable (Dim) Commentaire comme chaîne de
`caractères (String)
Dim nb_aleat As Integer ' un nombre aléatoire généré par le système
Dim nb_lu As Integer
Dim score As Integer
Dim compteur As Integer
Dim VbOk As Boolean
' Tester si l'utilisateur a cliqué sur le bouton OK
VbOk = MsgBox("Bienvenue dans le programme de calcul mental", vbOKCancel)
If VbOk Then
 ' Si oui initialiser un générateur de nombres aléatoires

nb_aleat = Int((100 * Rnd) + 1)    ' générer un nombre aléatoire entre 1 et 101
Commentaire = ""
InputBox("Le nombre à trouver est entre 1 et 100 " & " - Tentez votre chance : ")

While nb_lu <> nb_aleat
If nb_lu < nb_aleat Then
   Commentaire = "Trop petit"

   Else
   Commentaire = "Trop grand"


nb_lu = InputBox("Le nombre à trouver est entre 1 et 100 " & " - Tentez votre chance : " &
Commentaire)
' tant que ce nombre est différent de celui tapé par l'utilisateur,
              'boucler en incrémentant le compteur
Wend
MsgBox "Bravo ! Vous avez trouvé après " & compteur & " tentatives"
End If
End Sub
```

4. Manipulation des feuilles et cellules Excel

a. Manipulation de feuilles avec VBA : l'objet Worksheet

L'objet VBA **Worksheet,** représente une feuille de calcul Excel.

L'exemple suivant montre comment masquer la feuille de calcul 1 dans le classeur actif.

```
Worksheets(1).Visible = False
```

Ou encore

```
Worksheets("Feuille1").Visible = False
```

Le numéro d'index de la feuille de calcul indique sa position dans la barre d'onglets du classeur. ***Worksheets(1)*** est la première feuille du classeur (la plus à gauche) et **Worksheets** (***Worksheets.Count***) la dernière.
Le nom de la feuille de calcul apparaît sur l'onglet correspondant. Utilisez la propriété **Name**

pour définir ou renvoyer le nom de la feuille de calcul.
L'exemple suivant montre comment renommer la feuille "Feuil1" par " Nouveau_nom".

```
Worksheets("Feuil1").Name ="Nouveau nom"
```

L'objet **Worksheet** est également un membre de la collection **Sheets**. La collection **Sheets** contient toutes les feuilles du classeur (feuilles graphiques et feuilles de calcul).
La propriété **ActiveSheet** permet de faire référence à la feuille active. L'exemple suivant montre comment utiliser la méthode **Activate** pour activer la feuille de calcul Sheet1, définir l'orientation de la page en mode paysage, puis imprimer la feuille de calcul.

```
Worksheets ("Sheet1").Activate

ActiveSheet.PageSetup.Orientation = xlLandscape

ActiveSheet.PrintOut
```

b. Manipulation de cellules avec VBA : l'objet Range

L'objet **Range** peut représenter une cellule, une ligne, une colonne ou une sélection de cellules contenant un ou plusieurs blocs contigus de cellules ou une plage 3D. Les propriétés et méthodes suivantes sont utilisées lors de l'utilisation d'un objet **Range** :

- Propriété **Range**
- Propriété **Cells**
- **Range** et **Cells**
- Propriété **Offset**
- Méthode **Union**

Propriété Range

Pour renvoyer un objet **Range** représentant une cellule unique ou une plage de cellules, spécifiez Range (arg), *arg* désignant la plage. L'exemple suivant montre comment placer la valeur de la cellule A1 dans la cellule A5.

```
Worksheets("Feuil1").Range("A5").Value = Worksheets("Feuil1").Range("A1").Value
```

On peut aussi écrire des formules dans des cellules:

```
Worksheets(2).Range("A1") = 12          'Ecrire dans la cellule A1 de la feuille 2 le nombre 12
Worksheets(2).Range( "B1:D4") = 15  'Ecrire dans les cellules de la plage A1 à D4 le nombre 15
Worksheets(2).Range ("A2") = "=(A1+B1)/2" 'Ecrire dans la cellule A2 de la feuille2 la
formule (A1+B1)/2
```

La propriété **Offset** permet de donner une adresse relative. Par exemple:

```
ActiveCell.Offset(-1, 2)
```

Cela permet de référencer la cellule située une ligne avant et deux colonnes après la cellule active. La méthode **Union** permet de spécifier plusieurs plages de cellules :

```
Union (Range («A2: B3"), Range ("D1:D4") ) . Select
```

Cela permet de sélectionner les deux plages de cellules (A2:B3 et D1:D4).

Remarques :

- Pour affecter une fonction avec le nom « francisé », il faut écrire la commande ci-dessous :

```
Worksheets(1).Range ("A1").FormulaLocal = "=MOYENNE (A5:A10)"
```

- Pour mettre des guillemets dans une formule, il faut les « échapper » en les doublant : Worksheets(1).Range ("A1").FormulaLocal="=SI (A1=""AES2"";""ok"";""no"")"

Exercice 2 (Utiliser Exercice2.XLSM dans les ressources)

1. Ecrire une procédure VBA (à appeler « Calculer »), permettant de calculer et de remplir les cellules grisées.
2. Affecter cette procédure à un bouton à mettre sur le fichier des notes. Mettre le texte « Calculer » sur le texte de bouton.
3. Créer un autre bouton effaçant les différentes cases grisées. Vérifier le bon fonctionnement des 2 boutons.
4. Utiliser la méthode UNION pour effacer les cellules.

	A	B	C	D	E	F	G	H
1	**Nom**	**Prénom**	**Promotion**	**Interrogation**	**Examen**	**Moyenne**	**Coefficient**	
2	Largo	Winch	GIM	12	5	7,8	Interrogation	2
3	Cover	Harry	INFO	7	14	11,2	Examen	3
4	Dubosc	Franck	INFO	8	12	10,4		
5	Hochon	Paul	GIM	12,75	13	12,9		
6	Vador	Darck	GIM	11	12	11,6		
7	You	thu	GIM	12,5	8	9,8		
8	Gamejie	Sam	INFO	19	10	13,6		
9	Loch	Ness	INFO	14	12	12,8		
10	Reineudet	Neige	INFO	15,5	2	7,4		
11	Jedusor	Tom	GIM	7,75	14	11,5		
12	Hanzequine	Christine	INFO	4	16,5	11,5		
13	Bros	Mario	INFO	16,5	11,75	13,65		
14	Lovgood	Luna	INFO	2	8	5,6		
15	Tard	Guy	GIM	17,5	13,25	14,95		
16	Hafeu	Pierre	INFO	1,75	9	6,1		
17	Victor	Hugo	GIM	18,75	12	14,7		
18								
19	NB Etudiants GIM	7	Moyenne GIM	11,89285714				
20	NB Etudiants INFO	9	Moyenne INFO	10,25				
21								
22								
23								
24		CALCULER			EFFACER			
25								
26								
27								