

FONCTIONS APPROFONDIES AVEC LES MACROS

GESTION DES CELLULES ET DES CLASSEURS

GESTION DES CELLULES

RÉCUPÉRATION DE VALEURS D'UNE AUTRE FEUILLE

Stocke dans les variables *Montant*, *Remise* le contenu des cellules B18 et B19 situées dans la feuille « Gestion » :

```
With Sheets("Gestion")
    Montant = .[B18]
    Remise = .[B19]
End With
```

- ✓ Pour les coordonnées des cellules, il est également possible d'écrire :
.Cells(i,1)
.Range("B2 :B20")

GESTION D'UN CLASSEUR

ENREGISTREMENT D'UN CLASSEUR DANS UN DOSSIER ET UN NOM DE FICHIER

```
Dim ActuFichier As String
ActuFichier = ThisWorkbook.Path & "\Suivi\Gestion.xlsx"
ThisWorkbook.SaveAs ActuFichier
```

ENREGISTREMENT D'UN CLASSEUR AUTOMATIQUEMENT

```
ActiveWorkbook.Save
```

OUVERTURE D'UN CLASSEUR EN LIGNE

Exemple : récupération d'un tableau en ligne réalisé sur le site de Jotform :

```
Workbooks.Open Filename:="http://eu.jotform.com/excel/8779906063"
```

GESTION DES DOSSIERS

ACCÈS À UN DOSSIER PARTICULIER

```
ChDir ("C:\Users\Foubert\Documents\Messagerie")
```

ACCÈS À UN SOUS-DOSSIER DU DOSSIER ACTUEL

```
ChDir (ThisWorkbook.Path & "\Moranges")
```

ACCÈS AU DOSSIER PRÉCÉDENT LE DOSSIER COURANT

```
ChDir ".."
```

AFFICHAGE DU DOSSIER CONTENANT LE CLASSEUR OUVERT

```
MsgBox ThisWorkbook.Path
```

CRÉATION D'UN DOSSIER SI CELUI-CI N'EXISTE PAS

```
Dim ActuDossier As String
ActuDossier = ThisWorkbook.Path & "\Messagerie"
' Création du dossier si celui-ci n'existe pas
If Dir(ActuDossier, vbDirectory) = "" Then
    ' Création du dossier
    MkDir (ActuDossier)
Else
    ' Accès au dossier
    ChDir (ActuDossier)
End If
```

OUVERTURE D'UN LIEN HYPERTEXTE CONTENU DANS UNE CELLULE

Ouvre un lien hypertexte en fonction du contenu de la cellule choisie :

```
ActiveWorkbook.FollowHyperlink Address:="http://" & ActiveCell, NewWindow:=True
```

PROCÉDURES ET SOUS-PROCÉDURES

PROCÉDURES PUBLIQUES ET PRIVÉES

Une procédure peut appeler une sous-procédure. L'intérêt d'une sous-procédure est que c'est un code unique qui peut être appelée de plusieurs procédures afin d'effectuer des calculs, modifier des variables, etc.

Les procédures publiques sont accessibles de l'ensemble des modules alors que les procédures privées ne sont accessibles que du module en cours. Ainsi, si la procédure est privée, il pourra y avoir plusieurs procédures du même nom mais avec des codes différents dans des modules différents.

- La procédure publique commence par *Public* (qui est facultatif) :
Public Sub NomMacro(arguments)
- La procédure privée commence par *Private* :
Private Sub NomMacro(arguments)

De plus, il est possible de définir des arguments afin d'utiliser une valeur d'une variable de procédure vers une sous-procédure.

PROCÉDURE AVEC UNE SOUS-PROCÉDURE

Procédure *ChoixPays* qui vérifie si la cellule active contient bien du texte. Si c'est le cas, elle lance la sous-procédure *AffichagePays* avec la cellule active comme valeur obligatoire et la cellule à droite de la cellule active comme valeur facultative : `AffichagePays ActiveCell, ActiveCell.Offset(0, 1)`

Sous-procédure :

```
Private Sub AffichagePays(Pays As String, Optional Capital As String)
    ' Sous-procédure qui affiche le pays et, éventuellement, sa capitale
    MsgBox "Pays choisi : " & Pays & Chr(10) & Capital
End Sub
```

Procédure principale (à exécuter) :

```
Sub ChoixPays()
    If VarType(ActiveCell) = vbString Then
        AffichagePays ActiveCell, ActiveCell.Offset(0, 1)
    End If
End Sub
```

TYPE D'APPEL DE LA VARIABLE

La sous-procédure peut modifier ou non la variable d'origine :

- ByRef : fait des calculs qui changent la variable d'origine (valeur par défaut)
- ByVal : fait des calculs qui ne modifient pas la variable d'origine

La procédure et la sous-procédures qui suivent augmentent ou diminuent la valeur de base de 10% mais avec *ByVal*, la valeur de la procédure d'origine sera modifiée ce qui ne sera pas le cas avec *ByRef* :

Sous-procédures :

```
Private Sub Diminution(ByVal Montant As Integer)
    ' Diminue la valeur de 10% mais ne change pas la variable Montant d'origine
    Montant = Montant * 0.9
End Sub
Private Sub Augmentation(ByRef Montant As Integer)
    ' Augmente la valeur de 10% mais modifie la variable Montant d'origine
    Montant = Montant * 1.1
End Sub
```

- ✓ Dans ces deux procédures, *Montant as Integer* va recevoir le contenu de la variable (ci-dessous *Ventes*), mentionné après le nom de la macro appelée

Procédure principale (à exécuter) :

```
Sub Progression()
    Dim Ventes As Integer
    Ventes = 10000
    Diminution Ventes ' la valeur Ventes restera à 1000 (ByVal)
    MsgBox "la valeur Ventes reste à 1000 et non 9000 (ByVal) : " & Ventes
    Augmentation Ventes ' la valeur Ventes passera à 11000 (ByRef)
    MsgBox "la valeur Ventes passe à 11000 (ByRef) : " & Ventes
End Sub
```

POUR APPELER UNE PROCÉDURE

Dans le même module (ou si ce nom est unique) :

Il suffit de mettre le nom de la procédure, exemple :

```
Export ' Lance la procédure export située dans le même module
```

D'un module à un autre module :

Si le nom de la procédure existe dans plusieurs modules, il faut préciser le nom du module :

```
Suivi.MAJ OU Call Suivi.MAJ ' Nom du module.Nom de la procédure
```

La procédure *MAJ* est appelée dans le module ayant pour nom *Suivi*.

Pour appeler une macro d'un classeur vers un autre classeur :

Écrire `Application.Run` suivi du classeur, du module et de la procédure :

```
Application.Run "'Tableau des contacts.xlsm'!Formulaire.Ouverture"
```

- ✓ Ouvre la procédure *Ouverture*, dans le module *Formulaire* située dans le classeur *Tableau des contacts.xlsm*

LIENS SUR LES PROCÉDURES ET SOUS-PROCÉDURES

Lien vers les procédures publiques et privées :

http://www.excel-pratique.com/fr/vba/procedures_fonctions.php

Lien vers les procédures publiques et privées ainsi que les variables :

<https://openclassrooms.com/courses/analysez-des-donnees-avec-excel/modules-fonctions-et-sous-routines>

UTILISATION DES VARIABLES

VARIABLES PRIVÉES ET PUBLIQUES

Les variables privées ne sont utilisables que dans la procédure en cours d'exécution alors que les variables publiques peuvent être accessibles de n'importe où dans le classeur.

POUR DÉFINIR UNE VARIABLE ACCESSIBLE QUE DANS UNE PROCÉDURE

Placer la définition de la variable à l'intérieur de la procédure, exemple :

```
Sub Essor()  
    Dim Adresse As Integer  
    Adresse=25  
    Suite du code  
End Sub
```

POUR DÉFINIR UNE VARIABLE ACCESSIBLE À PARTIR DE TOUTES LES PROCÉDURES D'UN MODULE

Définir la variable avant toutes les procédures, exemple :

```
Dim Adresse As String  
' Le contenu de Adresse sera accessible  
' à partir de toutes les procédures de ce module uniquement  
Sub Essor()  
    Adresse=25  
    Envoi ' Appel de la macro Envoi dans le même module  
End Sub
```

- ✓ `Private Adresse As String` est équivalent à `Dim` et montre bien que la variable `Adresse` est réservée à ce module.

POUR DÉFINIR UNE VARIABLE ACCESSIBLE À PARTIR DE TOUS LES MODULES D'UN CLASSEUR

Définir la variable avant toutes les procédures avec `Public` et non `Dim`, exemple :

```
Public Adresse As String  
Sub Essor()  
    Adresse=25  
    Call Module2.Suite ' Appel la macro Suite dans le module Module2  
End Sub
```

POUR RÉCUPÉRER UNE VARIABLE DANS UN AUTRE CLASSEUR OUVERT :

Cela nécessite l'utilisation d'une sous-procédure dans le deuxième classeur.

Procédure dans le premier classeur (macro à exécuter) :

```
Public Adresse As String  
Sub Essor()  
    Adresse = 25  
    ' Envoi du contenu de la variable Adresse et  
    ' appel de la macro Import dans le module Module1 du classeur Classeur1.xlsm  
    Application.Run "Classeur1.xlsm!Module1.Import", Adresse  
End Sub
```

Sous-procédure dans le second classeur :

```
Sub Import(ByVal Afficher As String)  
    MsgBox "Valeur de la variable du premier classeur : " & Afficher  
End Sub
```

- ✓ La variable `Afficher` va recevoir le contenu de la variable (ici `Adresse`) mentionnée à la fin de l'appel de la procédure

POUR VIDER LE CONTENU D'UNE VARIABLE PRIVÉE OU PUBLIQUE

```
Adresse = Empty ' La variable Adresse perd son contenu
```

CHRONOMÉTRAGE DE LA DURÉE D'UNE MACRO

```
Dim Chrono As Date
' Début du chronométrage de la durée de la macro
Chrono = Now()
Code de la macro...
' Fin du chronométrage de la durée de la macro
MsgBox "La macro a duré " & Chr(10) & Format(Now() - Chrono, "hh:mm:ss")
```

VARIABLE EN TABLEAU

Il est possible de stocker plusieurs valeurs dans une même variable — sous forme de tableau à plusieurs dimensions — dans une seule variable ayant le même nom mais des numéros différents.

Exemple : Stocke dans une variable appelée *Contenu* la valeur des cellules de la ligne 1 à la ligne 5 et de la colonne 1 à la colonne 7 pour remettre ces valeurs dans une seconde feuille

```
Sub Stockage()
' Dimensionne la variable contenu de 1 à 5 pour la première dimension
' et de 1 à 7 pour la seconde dimension
Dim I As Integer, J As Integer
Dim Contenu(1 To 5, 1 To 7) As String
' Ubound : compte jusqu'à la limite choisie (1 ou 2) de la dimension de Contenu
For I = 1 To UBound(Contenu, 1)
    For J = 1 To UBound(Contenu, 2)
        Contenu(I, J) = Cells(I, J)
        Worksheets("feuille2").Cells(I, J) = Contenu(I, J)
    Next J
Next I
End Sub
```

COPIER-COLLER DE CELLULES

EN PASSANT PAR UN COPIER-COLLER

Copie le contenu des cellules de la 1^{ère} à la 8^e colonne de la ligne courante après la dernière cellule non vide de la colonne A

```
Sub Copie()
Range(Cells(ActiveCell.Row, 1), Cells(ActiveCell.Row, 8)).Select
Selection.Copy ' Copie les cellules sélectionnées
' Sélection de la cellule après la dernière cellule non vide de la colonne A
Range("A2").End(xlDown).Offset(1, 0).Select
ActiveSheet.Paste ' Colle les cellules copiées dans la dernière cellule
ActiveCell.Offset(0, 1).Select
End Sub
```

- ✓ Pour ne copier-coller que les valeurs (résultats des calculs) sans la mise en forme, remplacer *ActiveSheet.Paste* par *Selection.PasteSpecial Paste:=xlPasteValues*

EN PASSANT PAR UNE VARIABLE

Copie le contenu des cellules de la 1^{ère} à la 8^e colonne de la ligne courante vers la cellule A30 en stockant leur contenu dans la variable *ZoneOrigine*

```
Sub Transfert()
Dim ZoneOrigine As Range
Set ZoneOrigine = Range(Cells(ActiveCell.Row, 1), Cells(ActiveCell.Row, 8))
' Sélection de la cellule après la dernière cellule non vide de la colonne A
Range("A2").End(xlDown).Offset(1, 0).Select
' Copie le contenu de la variable ZoneOrigine à partir de la cellule A30 :
ZoneOrigine.Copy Destination:=ActiveCell
End Sub
```

CODES DE FONCTIONS, D'ERREURS ET DE LIENS AVEC OUTLOOK

UTILISER DES FONCTIONS EXCEL EN VBA

RECHERCHE D'UNE VALEUR DANS DES CELLULES

Stocke dans la variable *Lig* le numéro de la première ligne de la zone allant de A2 à A30 de la feuille de calcul *Cours* contenant le mot « Intérêt », stocké dans la variable *ActCh*

```
AcCh="Intérêt"  
Lig = Application.Match("*" & AcCh & "*", Worksheets("Cours").Range("A2:A30"), 0)
```

RECHERCHE D'UNE VALEUR CORRESPONDANTE

Affiche le courriel trouvé dans les cellules F2:F20 à la même ligne où se situe le nom *Nivat* trouvé dans les cellules A2:A20.

```
Courriel = Application.Index(Range("F2:F20"), _  
    Application.Match("Nivat", Range("A2:A20"), 0), 1)
```

- ✓ Le 0, à la fin, indique une recherche exacte de la fonction *Match* (permettant de retrouver l'information quel que soit le tri)
- ✓ Le 1 indique que l'index se fait dans la première colonne de la sélection (même si, ici, la sélection n'a qu'une seule colonne).

GESTION DES ERREURS

NE DÉCLENCHE L'ACTION QUE SI CE N'EST PAS UNE ERREUR

Si la variable *AcCh* ne contient pas un message d'erreur, alors un message est affiché.

```
If Not IsError(AcCh) Then  
    MsgBox "Aucune erreur détectée!"  
End If
```

GESTION DES ERREURS

Si le classeur *gestion.xlsx* n'est pas ouvert, la macro ouvre le classeur avant de l'activer :

```
Sub AtteindreClassuer()  
    ' En cas d'erreur (le classeur n'est pas ouvert lors de l'activation)  
    ' la macro va à l'étiquette ErreurOuverture: pour ouvrir le classeur  
    On Error GoTo ErreurOuverture  
    Workbooks("gestion.xlsx").Activate ' Activation du classeur  
    On Error GoTo 0 ' Arrêt de la gestion des erreurs  
    MsgBox "Le classeur a été activé"  
    Exit Sub ' arrêt de la macro  
    ' Ouverture du classeur en cas d'erreur (s'il n'est pas ouvert)  
  
ErreurOuverture:  
    Workbooks.Open ThisWorkbook.Path & "\gestion.xlsx" ' Ouverture du classeur  
    MsgBox "Le classeur a été ouvert"  
    Resume ' Retour à la ligne erronée  
End Sub
```

ENVOI D'UN COURRIEL

Macro qui ouvre un courriel dans Outlook avec des destinataires, un objet, le corps du texte et une pièce jointe

- ☞ Avant : Dans menu Outils, Références, cocher Microsoft Outlook [N° de version] Object Library

```
Sub Courriel()  
    Dim ObjOutlook As New Outlook.Application  
    Dim ObjOutlookmail As MailItem  
    Dim CourrielAdresseA As String  
    Dim CourrielAdresseCC As String  
    Dim CourrielAdresseCCi As String  
    Dim CourrielAdresseEx As String  
    Dim CourrielObjet As String, CourrielMessage As String, FichierJoint As String  
  
    ' Paramètres pour Outlook  
    Set ObjOutlook = New Outlook.Application  
  
    ' Nom de la pièce jointe  
    FichierJoint = ThisWorkbook.Path & "\Envoi\Liste.pdf"  
  
    ' Éléments du courriel  
    CourrielAdresseEx = "heulluy@outlook.com"  
    CourrielAdresseA = "foubert@gmail.com"  
    CourrielAdresseCC = "nivat@gmail.com;emont@live.fr"  
    CourrielAdresseCCi = "comborn@yahoo.com"  
    CourrielObjet = "Envoi de la liste"  
    CourrielMessage = "Bonjour," & vbCrLf & _  
        "Voici la liste demandée." & vbCrLf & "Bien cordialement,"  
  
    Set oBjMail = ObjOutlook.CreateItem(olMailItem) ' Création du courriel  
    With oBjMail  
        .SentOnBehalfOfName = CourrielAdresseEx ' Expéditeur  
        .To = CourrielAdresseA ' destinataire principal  
        .CC = CourrielAdresseCC ' destinataire en copie  
        .BCC = CourrielAdresseCCi ' destinataire en copie invisible  
        .Subject = CourrielObjet ' l'objet du mail  
        .Attachments.Add FichierJoint ' Nom du fichier joint  
        .Body = CourrielMessage ' Corps du message  
        .Display ' ou écrire « send » pour l'envoi direct du message  
    End With  
End Sub
```

- ☞ Au lieu de mettre .body, écrire .HTMLBody si le message est en HTML. ■