E.S.L.S.C.A.

Programmation VBA

Thème 4 :
La Programmation avancée

Les Fonctions.

Les fonctions sont des procédures qui renvoient une valeur. Elles peuvent posséder des arguments qui représentent des variables définies dans l'appel de la fonction. Dans l'exemple suivant, une fonction calculant la somme de deux entiers est appelée d'une procédure.

```
'Déclaration de la fonction de type Integer
Function MaSomme(Nbre1 As Integer, Nbre2 As Integer)As Integer
    MaSomme = Nbre1 + Nbre2
End Function

'Procédure appelant la fonction
Sub Calcul ()
    Dim Resultat As Integer
    Dim Val1 As Integer, Val2 As Integer
    Val1 = 2
    Val2 = 3
    'Appel de la fonction avec ses arguments
    Resultat = MaSomme(Val1, Val2) 'Resultat = 5
End Sub
```

Reprenons le tableau des élèves ainsi que la correspondance des mentions par rapport aux notes.

	A	B
1	ELEVE	NOTE
2	PIERRE	5
3	JACQUES	15
4	JEAN	10
5	VALERIE	12
6	PAUL	18
7	DELPHINE	13
8	ANTOINE	0
9	JEANNE	6
10	ANDRE	19
11	JOCELYNE	8
12	FELIX	12
13	JUSTIN	15
14	ETIENNE	6
15	MARIE	5

Notes :	*Mention* :
0	Nul
1 à 5	Moyen
6 à 10	Passable
11 à 15	Bien
16 à 19	Très bien
20	Excellent

La fonction suivante va définir la mention par rapport aux notes.

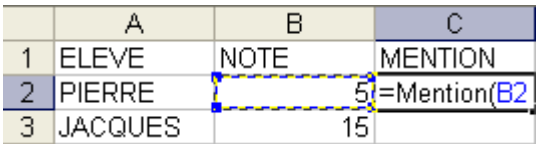
```
'Déclaration de la fonction de type String
Function Mention(Note As Integer)As String
    Select Case Note
        Case 0
            Mention = "Nul"
        Case 1 To 5
            Mention = "Moyen"
        Case 6 To 10
            Mention = "Passable"
        Case 11 To 15
            Mention = "Bien"
        Case 16 To 19
            Mention = "Très Bien"
        Case Else
            Mention = "Excellent"
    End Select
End Function
```

La procédure suivante va appeler la fonction "Mention" pour chaque élève et inscrire sa valeur dans la colonne C.

```
Sub Calcul_Mention ()
    Dim i As Integer 'Nbre d'élève
    Dim ValNote As Integer
    For i = 1 To 14
        ValNote = Range("A1").Offset(i, 1)
        'L'instruction suivante va placer dans la colonne C
        'la valeur de la fonction "Mention"
        Range("C1").Offset(i) = Mention(ValNote)
    Next i
End Sub
```

	A	B	C
1	ELEVE	NOTE	MENTION
2	PIERRE	5	Moyen
3	JACQUES	15	Bien
4	JEAN	10	Passable
5	VALERIE	12	Bien
6	PAUL	18	Très Bien
7	DELPHINE	13	Bien
8	ANTOINE	0	Nul
9	JEANNE	6	Passable
10	ANDRE	19	Très Bien
11	JOCELYNE	8	Passable
12	FELIX	12	Bien
13	JUSTIN	15	Bien
14	ETIENNE	6	Passable
15	MARIE	5	Moyen

Les fonctions peuvent également être appelées d'une feuille de calcul comme toutes les fonctions intégrées d'Excel. Dans notre exemple, on aurait pu saisir "=Mention(B2)" dans la cellule "C2" puis recopier cette formule sur la plage "C3:C15" pour obtenir le même résultat. En saisissant directement la fonction dans la feuille de calcul, la valeur de la mention change si la note est modifiée.

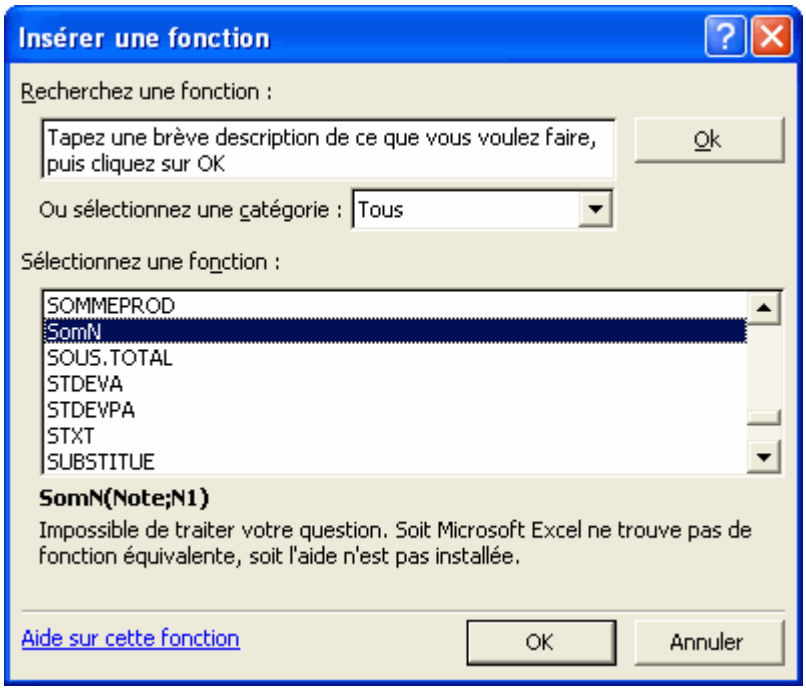


	A	B	C
1	ELEVE	NOTE	MENTION
2	PIERRE	5	=Mention(B2
3	JACQUES	15	

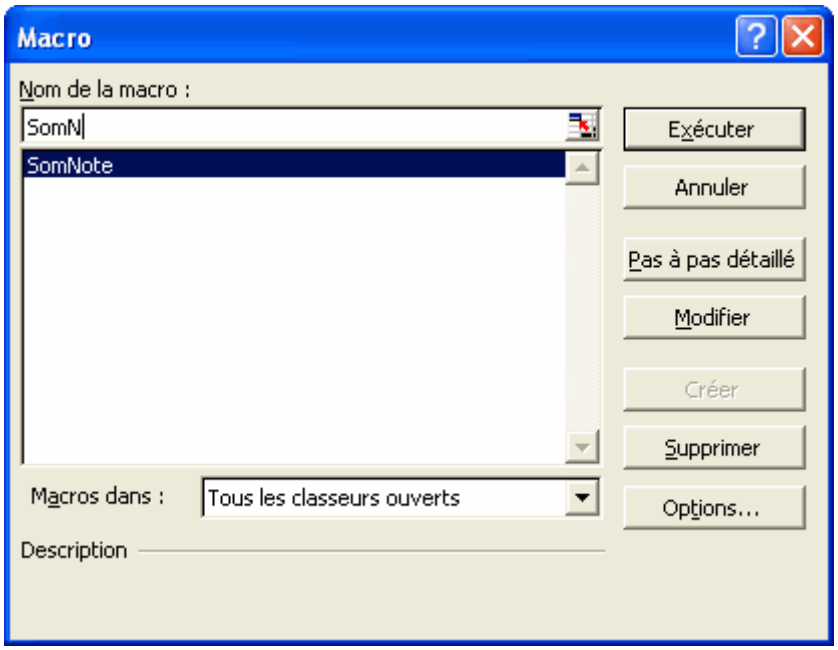
Dans les exemples précédents, les arguments des fonctions sont obligatoires. Pour rendre un argument facultatif, il faut le déclarer par le mot Optional. La fonction suivant va calculer la somme des notes avec comme option, une note minimale.

```
'Déclaration de la fonction de type Integer
Function SomN(Note As Range, Optional N1 As Integer)As Integer
    Dim Item As Variant
    'Item prend la valeur de chaque cellule de la plage de
    'cellule définie dans l'argument Note
    For Each Item In Note
        'si l'argument N1 est défini
        If N1 > 0 Then
            'test si la valeur de la cellule est > ou = à N1
            If Item >= N1 Then
                SomN = SomN + Item
            End If
        Else 'l'argument N1 n'est pas défini ou est = 0
            SomN = SomN + Item
        End If
    Next Item
End Function

'Appel de la fonction sans argument N1
Sub SommeNote ()
    Dim Resultat As Integer
    Resultat = SomN(Range("B2:B15")) 'Resultat = 144
End Sub

'Appel de la fonction avec argument N1(note minimale)
Sub SommeNote ()
    Dim Resultat As Integer
    Resultat = SomN(Range("B2:B15"), 10) 'Resultat = 114
End Sub
```

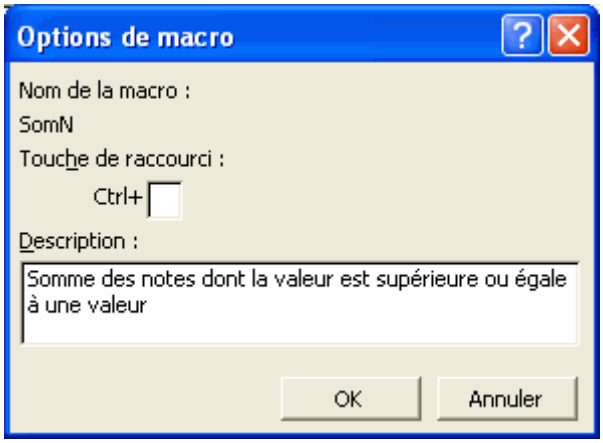
Vos fonctions créées apparaissent dans la liste des fonctions d'Excel accessible par le menu "Insertion-Fonctions".



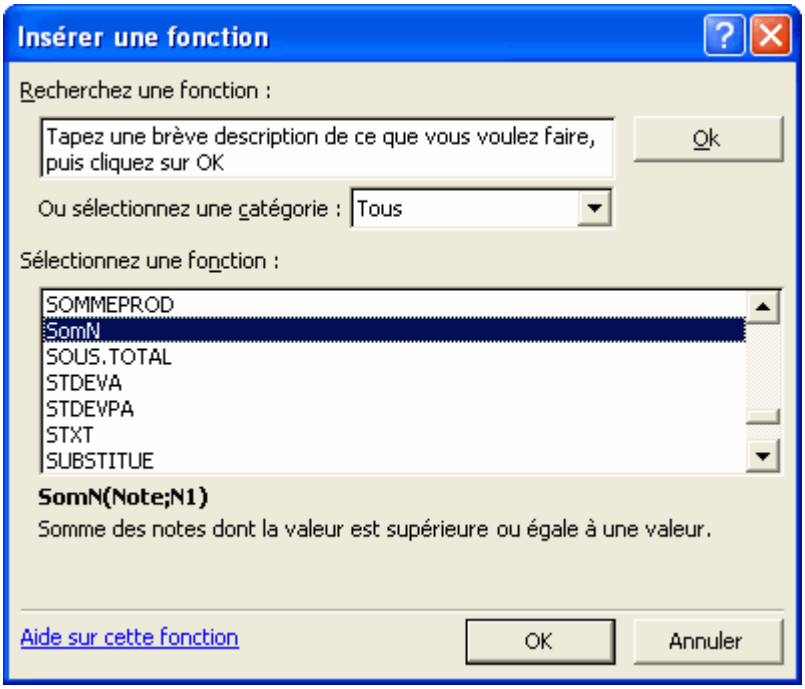
Pour ajouter une zone d'aide à vos fonctions, ouvrez la boite de dialogue Macro par le menu "Outils-Macro-Macros". Dans cette boite, seuls les procédures apparaissent. Saisissez le nom de votre fonction puis cliquez sur "Options".



Saisissez votre texte dans la zone description puis validez.



Voila comment apparaît votre fonction dans la liste des fonctions :



Les fonctions de texte.

VBA possède des fonctions permettant d'extraire une chaîne de caractères d'un texte.
La fonction **Len** renvoie le nombre de caractères d'un texte.

```
Dim Message As String,  Longueur As Integer
Message =  "Fonctions de texte"
Longueur = Len(Message)     'Longueur renvoie 18
```

La fonction **Left** renvoie un nombre de caractères en partant de la gauche. La syntaxe est Left(Chaîne de caractères, Nombre de caractères).

```
Dim Message As String, MTexte As String
Message =  "Fonctions de texte"
MTexte = Left(Message, 1)     'Renvoie "F"
MTexte = Left(Message, 9)     'Renvoie "Fonctions"
```

La fonction **Right** renvoie un nombre de caractères en partant de la droite. La syntaxe est Right(Chaîne de caractères, Nombre de caractères).

```
Dim Message As String, MTexte As String
Message =  "Fonctions de texte"
MTexte = Right(Message, 1)     'Renvoie "e"
MTexte = Right(Message, 8)     'Renvoie "de texte"
```

La fonction **Mid** renvoie un nombre de caractères en partant d'un caractère défini. La syntaxe est Mid(Chaîne de caractères, Départ, Nombre de caractères). Si le Nombre de caractères n'est pas indiqué, la fonction renvoie tous les caractères à partir de la position de départ.

```
Dim Message As String, MTexte As String
Message =  "Fonctions de texte"
MTexte = Mid(Message, 2, 5)     'Renvoie "oncti"
MTexte = Mid(Message, 11, 2)     'Renvoie "de"
MTexte = Mid(Message, 11)     'Renvoie "de texte"
```

La fonction **LTrim** supprime les espaces se trouvant avant la chaîne de caractères.

```
Dim Message As String, MTexte As String
Message =  "   Fonctions   "
MTexte = LTrim(Message)     'Renvoie "Fonctions   "
```

La fonction **RTrim** supprime les espaces se trouvant après la chaîne de caractères.

```
Dim Message As String, MTexte As String
Message =  "   Fonctions   "
MTexte = RTrim(Message)     'Renvoie "   Fonctions"
```

La fonction **Trim** supprime les espaces se trouvant avant et après la chaîne de caractères.

```
Dim Message As String, MTexte As String
Message =  "   Fonctions   "
MTexte = Trim(Message)    'Renvoie "Fonctions"
```

La fonction **Ucase** convertie la Chaîne de caractères en majuscules.

```
Dim Message As String, MTexte As String
Message =  "Fonctions de texte"
MTexte = Ucase(Message)
'Renvoie "FONCTIONS DE TEXTE"
```

La fonction **Lcase** convertie la Chaîne de caractères en minuscules.

```
Dim Message As String, MTexte As String
Message =  "Fonctions de texte"
MTexte = Lcase(Message)
'Renvoie "fonctions de texte"
```

La fonction **Application.Proper** convertie la Chaîne de caractères en nom propre.

```
Dim Message As String, MTexte As String
Message =  "Fonctions de texte"
MTexte = Application.Proper(Message)
'Renvoie "Fonctions De Texte"
```

La fonction **Replace** permet de remplacer une chaîne de caractères par une autre.

```
Dim Message As String, MTexte As String
Message =  "Fonctions de texte"
MTexte = Replace(Message, "te", "et")
'Renvoie "Fonctions de etxet"
```

Cette fonction possède des arguments facultatifs permettant de déterminer la position du premier remplacement et le nombre de remplacement à effectuer. La syntaxe est **Replace**(Texte, Chaîne à remplacer, chaîne de remplacement, Départ, Nombre de remplacement).

```
Dim Message As String, MTexte As String
Message =  "Fonctions de texte"
MTexte = Replace(Message, "t", "WW", 3, 1)
'Renvoie "ncWWions De texte"

MTexte = Replace(Message, "t", "WW", , 2)
'Renvoie "FoncWWions de WWexte"
```

La fonction **Val** renvoie la valeur numérique d'une chaîne de caractères. Si la chaîne de caractères est composée de chiffres et de lettres, la valeur s'arrête au premier caractère non numérique.

```
Dim Message As String, MTexte As Double
Message =  "2003"
MTexte = Val(Message)
'Renvoie 2003

Message =  "a 2003"
MTexte = Val(Message)
'Renvoie 0

Message =  "   2003 2004"
MTexte = Val(Message)
'Renvoie 20032004

Message =  "2003 et 2004"
MTexte = Val(Message)
'Renvoie 2003
```

La fonction **IsNumeric** permet de tester si une chaîne de caractères est numérique. Elle renvoie une valeur de type Boolean.

```
Dim Message As String, MTexte As Integer
Message =  100
If IsNumeric(Message) = True Then
    MTexte = Message + 10 'MTexte prend la valeur 110
End If
```

La fonction **IsDate** permet de tester si une chaîne de caractères est une date. Elle renvoie une valeur de type Boolean.

```
Dim Message As String, MTexte As Integer
Message =  "1 MARS 2000"
If IsDate(Message) = True Then
    MTexte = Month(Message) 'MTexte prend la valeur
                             3(3ème mois de l'année)
End If
```

Certaines fonctions permettent de convertir des données en d'un type défini. Par exemple, la fonction **CDate** va convertir des données en date.
Tableau de fonctions de conversions de données :

Fonctions :	*Type* :
CBool	Boolean
CByte	Byte
CCur	Currency
CDate	Date
CDbl	Double
CDec	Decimal
CInt	Integer
CLng	Long
CSng	Single
CStr	String
CVar	Variant

```
Dim Message As Double, MTexte As Integer
Message =  325.25
MTexte = CInt(Message) 'MTexte prend la valeur 325
```

Le format des dates et heures est défini par la fonction **Format**. La syntaxe est **Format**(MaDate, Format).

```
Dim MaDate As Date, MDate As String
MaDate =  date 'date du jour
MDate = Format(M, "dd mmmm yyyy") 'MDate prend la valeur
                                  '"05 Octobre 2003"
```

La fonction **Format** permet également de formater les nombres.

```
Dim MonNombre As String
MonNombre = Format(1500, "0 000") 'MonNombre prend la 'valeur "1 500"
MonNombre = Format(1500, "0 000.00 Euros") 'MonNombre 'prend la valeur
"1 500.00 Euros"
```

Classeur, Feuilles, Cellules.

Les classeurs.
Les classeurs sont désignés par le mot "Workbook". Ils peuvent être ouverts, fermés, enregistrés, activés, masqués supprimés ... par une instruction VB.
Quelques exemples d'instructions sur les classeurs :

```
'Ajouter un nouveau classeur
Workbooks.Add

'Fermer un classeur. Le nom du classeur ou son index peut être indiqué.
Workbooks("NomDuClasseur.xls").Close
'ou
Workbooks(1).Close

'Fermer le classeur actif.
ActiveWorkbook.Close

'Ouvrir un classeur.
Workbooks.Open "c:\Chemin\NomDuFichier.xls"

'Activer un classeur.
Workbooks("NomDuClasseur.xls").Activate
```

Certaines méthodes de l'objet Workbook possèdent des arguments.
Quelques exemples :

```
'Fermer un classeur sans l'enregistrer
Workbooks("NomDuClasseur.xls").Close False

'Ouvrir un classeur en lecture seule.
Workbooks.Open "c:\Chemin\NomDuFichier.xls", , True

'Enregistrer un classeur sous "Test.xls" avec comme mot de passe
"testpass"
Workbooks(1).SaveAs "test.xls", , "testpass"
```

Les feuilles de calcul.
Les feuilles de calcul sont désignées par le mot "Worksheet". Comme les Workbook, ces objets possèdent de nombreuses propriétés et méthodes.
Quelques exemples d'instructions sur les feuilles :

```
'Selectionner une feuille
Worksheets("Feuil1").Select

'Récupérer le nom de la feuille active dans une variable.
MaFeuille = ActiveSheet.Name

'Masquer une feuille.
Worksheets("Feuil1").Visible = False

'Supprimer une Feuille.
Worksheets("Feuil1").Delete
```

Les exemples précédents font référence aux feuilles du classeur actif. Vous pouvez également faire référence aux feuilles des autres classeurs ouverts :

```
'Copier la Feuil2 de Classeur.xls dans un nouveau classeur
Workbooks("Classeur.xls").Worsheets("Feuil2").Copy
```

Les cellules.
Une plage de cellules est désignée par l'objet "Range". Pour faire référence à la plage de cellule "A1:B10", on utilisera Range("A1:B10").

```
'Effacer les données et la mise en forme de la plage de cellule "A1:B10"
Range("A1:B10").Clear
```

L'objet Range permet également de faire référence à plusieurs plages de cellules non contiguës.

```
'Sélectionner les plages de cellule "A1:B5" et "D2:F10"
Range("A1:B5,D2:F10").Select
```

Pour faire référence à une seule cellule, on utilisera l'objet Range("Référence de la cellule) ou Cells(Numéro de ligne, Numéro de colonne).

```
'Ecrire 5 dans la cellule "A3"
Range("A3").Value = 5
'ou
Cells(3, 1).Value = 5
```

Dans l'exemple suivant, nous allons recopier la plage de cellules "A1:B10" de la "Feuil1" du classeur actif dans la cellule "D5" de la "Feuil2" du classeur "Classeur2". Voici à ce que l'enregistreur de macro produirait comme code :

```
Range("A1:B10").Select
Selection.Copy
Windows("Classeur2").Activate
Sheets("Feuil2").Select
Range("D5").Select
ActiveSheet.Paste
Sheets("Feuil1").Select
Application.CutCopyMode = False
Windows("Classeur1").Activate
```

Voici maintenant le code tel qu'il pourrait être écrit sur une seule ligne de code:

```
Range("A1:B10").Copy Workbooks("Classeur2"). _
Worksheets("Feuil2").Range("D5")
```

On peut utiliser une autre syntaxe pour faire référence à une cellule :

```
'la ligne
Workbooks("Classeur2").Worksheets("Feuil2").Range("D5")
'peut être remplacée par:
Range("[Classeur2]Feuil2!D5")
```

En utilisant des variables objets (très utiles lorsque votre programme fait souvent référence aux mêmes plages de cellules), le code pourrait devenir :

```
Dim Cel1 As Range, Cel2 As Range
Set Cel1 = Range("A1:B1")
Set Cel2 = Workbooks("Classeur2"). _ Worksheets("Feuil3").Range("D5")
Cel1.Copy Cel2
```

VB vous permet également de changer le format des cellules (polices, couleur, encadrement ...). L'exemple suivant applique la police "courrier" en taille 10, en gras, en italique et de couleur rouge. Notez l'utilisation du bloc d'instruction **With - End With** faisant référence à l'objet Font(police) de l'objet Cel1

```
Dim Cel1 As Range
Set Cel1 = Range("A1")
With Cel1.Font
    .Bold = True
    .Italic = True
    .Name = "Courier"
    .Size = 10
    .Color = RGB(255, 0, 0)
End With
```

A partir d'une cellule de référence, vous pouvez faire appel aux autres cellules par l'instruction "Offset". La syntaxe est Range(Cellule de référence).Offset(Nombre de lignes, Nombre de colonne).

```
'Pour écrire 5 dans la cellule "B2", on pourrait utiliser :
Range("A1").Offset(1, 1) = 5
'Ecrire une valeur à la suite d'une liste de valeur dans la colonne A:
Dim NbEnreg As Integer
'NbEnreg correspond au nombre d'enregistrement de la colonne A:
NbEnreg = Range("A1").End(xlDown).Row
Range("A1").Offset(NbEnreg, 0) = 10
```

Les arguments (Nombre de lignes, Nombre de colonnes) de l'instruction Offset sont facultatifs et leur valeur par défaut est 0. La dernière ligne de code de l'exemple précédent aurait pu s'écrire :

```
Range("A1").Offset(NbEnreg) = 10
```

Exercices :

Ecrire les programmes VBA qui permettent de :

1. Afficher le nom des feuilles d'un classeur
2. Ecrire une phrase à l'envers
3. Donner la valeur maximale d'une sélection
4. Extraire les consonnes et les voyelles d'une chaîne de caractères
5. Insérer une ligne entre chaque cellule
6. Mettre de couleur rouge les cellules d'une sélection dont le nombre est > 10
7. Afficher le nombre de cellules vides d'une sélection
8. Mettre l'heure du système dans une cellule
9. Afficher aléatoirement un nombre compris entre 1 et 10
10. Fait la somme des nombres écrits en rouge
11. Afficher le numéro de la semaine.
12. Convertir une valeur en euros
13. Donner le nombre de fichiers d'un répertoire dont vous saisissez le nom dans la cellule A1
14. Lister les fichiers dont la date d'enregistrement est > à une date donnée
15. Copier des fichiers d'un dossier à un autre.
16. Supprimer des fichiers
17. Renommer des fichiers
18. Empêcher l'enregistrement d'un fichier
19. Enregistrer un classeur sous le nom de la valeur d'une cellule
20. Tester l'existence d'un fichier
21. Mettre en majuscule tous les onglets d'un classeur
22. Fermer tous les classeurs ouverts
23. Mettre le nom Répertoire + nom du dossier dans le pied de page
24. Obliger l'écriture en majuscule
25. Obliger l'écriture d'un nombre numérique