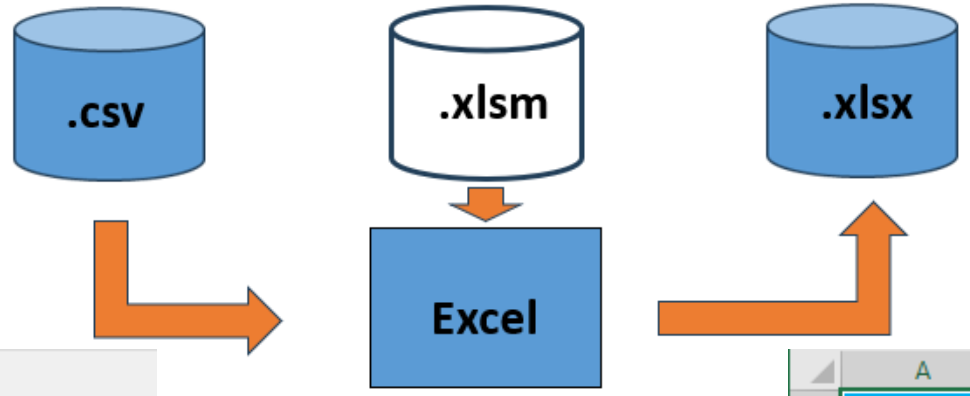
COURS E1 : EXCEL VBA (SECTIONS 1.0 À 3.4.2)

2.1 Enregistrement d'une macro VBA

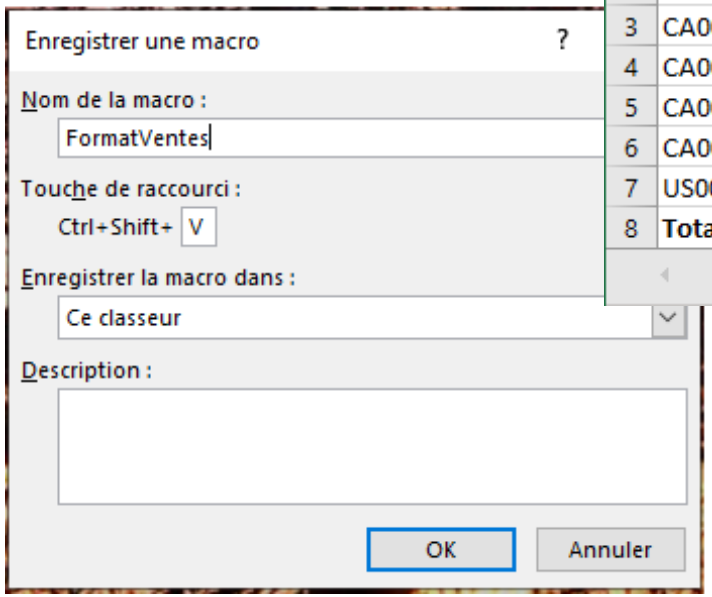


Ventes2023.csv

```
1  Succ,Dept,Montant
2  CA001,11,135021
3  CA001,12,283213
4  CA001,21,435789
5  CA002,11,214000
6  CA002,21,200456
7  US001,11,432990
8
```

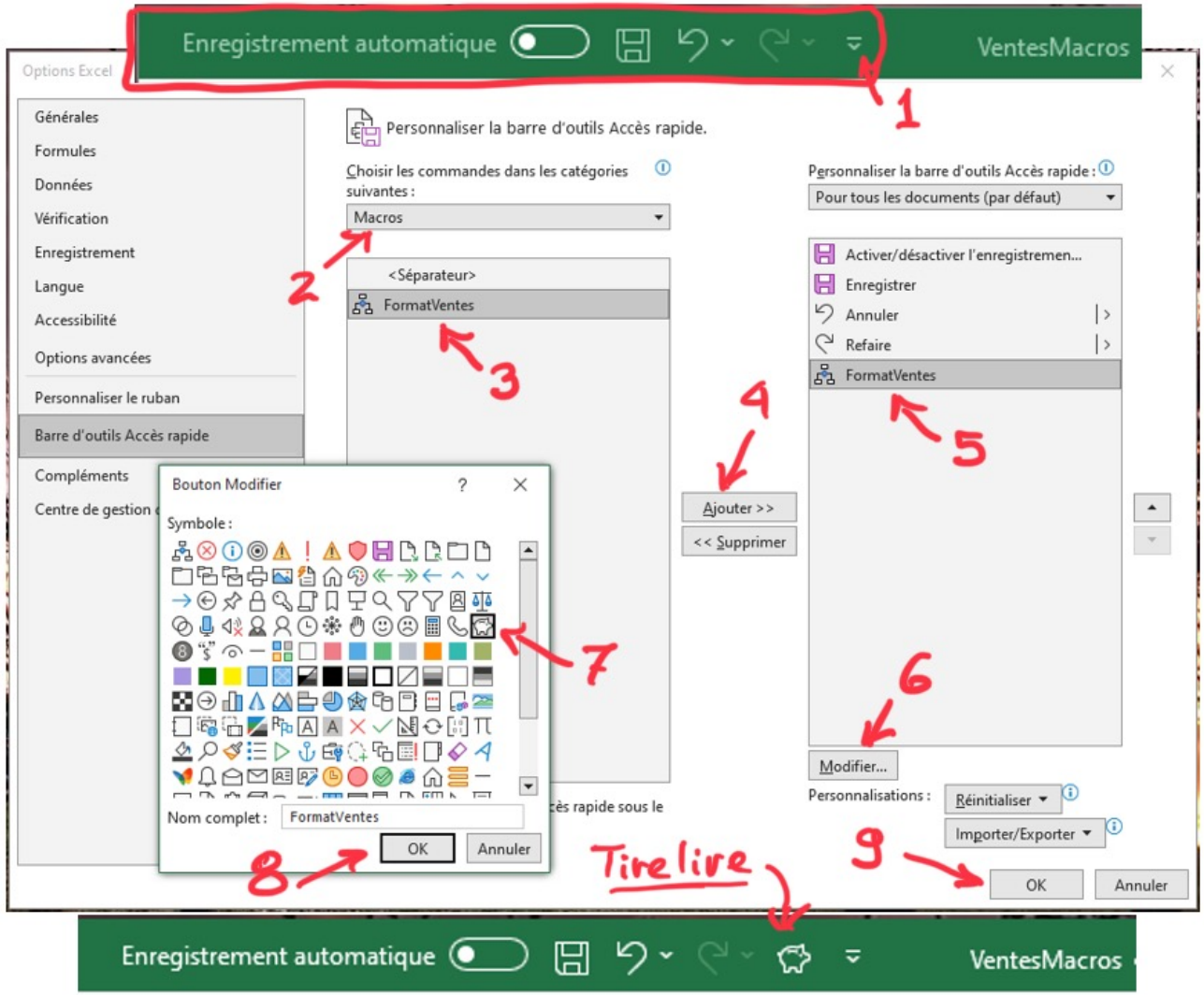
	A	B	C	D
1	Succ	Dept	Montant	
2	CA001	11	135021	
3	CA001	12	283213	
4	CA001	21	435789	
5	CA002	11	214000	
6	CA002	21	200456	
7	US001	11	432990	
8	Total des ventes		1701469	

Ventes2023



COURS E1 : EXCEL VBA

2.2 Barre d'outils Accès rapide (Windows et MacOS)



COURS E1 : EXCEL VBA

2.3 & 2.4 Optimisation du script VBA enregistré

```
Sub FormatVentes()
'FormatVentes Macro
'Touche de raccourci du clavier: Ctrl-Shift-V
  Workbooks.Open Filename:="X:\A_Faire\Ventes2023.csv"
  Range("A1:C1")
  Selection.Font.Bold = True
  With Selection.Interior
     .Pattern = xlSolid
'     .PatternColorIndex = xlAutomatic
     .Color = 15773696
'     .PatternTintAndShade = 0
  End With
  With Selection
     .HorizontalAlignment = xlCenter
'     .WrapText = False
'     .Orientation = 0
'     .AddIndent = False
'     .IndentLevel = 0
'     .ShrinkToFit = False
'     .ReadingOrder = xlContext
'     .MergeCells = False
  End With
  Range("A8").Select
  Selection.Font.Bold = True
  ActiveCell.FormulaR1C1 = "Total des ventes:"
```

```
Sub FormatVentes()
'FormatVentes Macro
'Touche de raccourci du clavier: Ctrl-Shift-V
  Workbooks.Open Filename:="X:\A_Faire\Ventes2023.csv"
  With Range("A1:C1")          'Ligne de titre
     .Font.Bold = True
     .Interior.Pattern = xlSolid
     .Interior.Color = 15773696
     .HorizontalAlignment = xlCenter
  End With
  With Range("A8")             'Texte ... Total des ventes
     .Font.Bold = True
     .FormulaR1C1 = "Total des ventes:"
End With
```

COURS E1 : EXCEL VBA

2.3 & 2.4 Optimisation du script VBA enregistré

```
  Range("C8").Select
  Selection.Font.Bold = True
'  Selection.Borders(xlDiagonalDown).LineStyle = xlNone
'  Selection.Borders(xlDiagonalUp).LineStyle = xlNone
'  Selection.Borders(xlEdgeLeft).LineStyle = xlNone
  With Selection.Borders(xlEdgeTop)
      .LineStyle = xlContinuous
'     .ColorIndex = 0
'     .TintAndShade = 0
      .Weight = xlThin
  End With
  With Borders(xlEdgeBottom)
      .LineStyle = xlDouble
'     .ColorIndex = 0
'     .TintAndShade = 0
      .Weight = xlThick
  End With
'  Selection.Borders(xlEdgeRight).LineStyle = xlNone
'  Selection.Borders(xlInsideVertical).LineStyle = xlNone
'  Selection.Borders(xlInsideHorizontal).LineStyle = xlNone
  ActiveCell.FormulaR1C1 = "=SUM(R[-6]C:R[-1]C)"
'  Range("C9").Select
```

```
With Range("C8")
   .Font.Bold = True
   With Borders(xlEdgeTop)
      .LineStyle = xlContinuous
      .Weight = xlThin
   End With
   With Borders(xlEdgeBottom)
      .LineStyle = xlDouble
      .Weight = xlThick
   End With
   .FormulaR1C1 = "=SUM(R[-6]C:R[-1]C)"
End With
```

```
Range("A1").Select
ActiveWorkbooks.SaveAs Filename:="X:\A_Faire\Ventes2023.xlsx", FileFormat:= _
   xlOpenXMLWorkbook, CreateBackup:= False   'Sauvegarde du fichier .xlsx
```

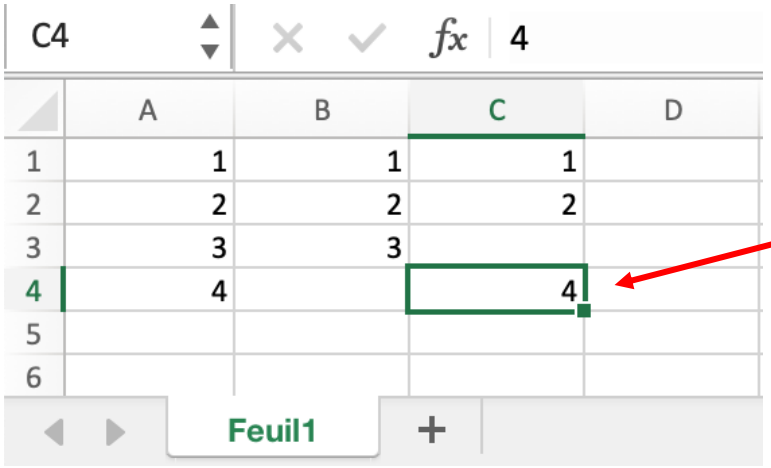
```
.xls  -> xlExcel8 (format Excel97)
.xlsx -> xlOpenXMLWorkbook (Classeur Excel)
.xlsm -> xlOpenXMLWorkbookMacroEnabled (Classeur avec macro)
.csc  -> xlCSVUTF8 (format texte csv)
```

2.5 Recommandations

En général, l'édition et l'optimisation d'un script VBA obtenu par enregistrement d'une macro requière plusieurs étapes de transformation. Voici quelques recommandations :

1. enlever tous les **`Select`** inutiles;
2. regrouper dans une même structure **`With...End With`** les modifications de propriétés d'un même objet;
3. utiliser des variables, tels que **`i`** et **`j`**, avec la méthode **`Cells(i,j)`** plutôt que **`Range(etiquette)`**;
4. utiliser les formules relatives **`R1C1`**;
5. transférer ou copier-coller des données en une seule instruction (sans sélection);
6. utiliser une méthode robuste pour déterminer la dernière ligne de données d'une colonne.

```
DerniereLigne = Cells(Rows.Count, 3).End(xlUp).Row 'Remonte à la dernière donnée de la col. 3
```



Dernière cellule de la colonne 3

COURS E1 : EXCEL VBA

3.1.1 Types des variables

Types numériques

```
Dim a As Byte      'Entier de 0 à 255
Dim b As Integer   'Entier de -32768 à 32767
Dim c As Long      'Entier entre +/- 2.15 milliard
Dim d As Single    'Nombre à point flottant simple précision
Dim e As Double    'Nombre à point flottant double précision
Dim f As Currency  'Nombre à point fixe à 4 décimales entre +/- 9.2 trillion
```

Types autres

```
Dim g As Boolean   'Logique vrai ou faux (True/False)
Dim h As String    'Chaîne de 1 à 65400 caractères
Dim i As Date      'Valeur de date entre 1 jan. 100 à 31 déc. 9999
Dim j As Variant   'Type à ajustement automatique (par défaut)
Dim k As Object    'Tous les objets Excel (Workbook, Worksheet, Range, Chart, etc)
```

Exemples d'objets Excel VBA

```
Dim MesBoutons As vbMsgBoxStyle      'Style de boîte message
Dim MaReponse  As vbMsgBoxResult     'Résultat d'une boîte message
Dim ChoixFichier As FileDialog       'Résultat d'une boîte de choix de fichier
Dim Cellule, Plage As Range          'Objet cellule ou plage de cellules
Dim MaFeuille As Worksheet           'Objet feuille de calcul
Dim MonClasseur As Workbook          'Objet classeur Excel
```

COURS E1 : EXCEL VBA

3.1.2 à 3.1.4 Opérateurs arith., booléens et logiques

Opérateurs arithmétiques

```
A + B   'Addition de A et B
A - B   'Soustraction de A par B
A * B   'Multiplication de A et B
A / B   'Division de A par B
A ^ B   'Exponentiation de A par B (A exposant B)
```

Opérateurs booléens

```
  10 >  8      'Vrai, puisque 10 est plus grand que 8
  10 <> 8      'Vrai, puisque 10 est différent de 8
  10 <= 8      'Faux, puisque 10 n'est pas plus petit ou égale à 8
  10 >= 10     'Vrai, puisque 10 est plus grand ou égale à 10, ici égale
 "A" <  "B"    'Vrai, puisque A est plus petit que B en ordre alphabétique
"MEC" = "GCH"  'Faux, puisque MEC n'est pas identique à GCH
```

Opérateurs logiques

```
A And B   'Vrai, si A et B sont tous les deux vrai
C Or D    'Vrai, si au moins C ou D est vrai (vrai si les 2 sont vrai)
C Xor D   'Vrai, si seulement C ou D est vrai (faux si les 2 sont vrai)
Not E     'Vrai, si E est faux
```

COURS E1 : EXCEL VBA
3.1.5 Fenêtre de débogage

```
Function MaLogique(a As Boolean) As Boolean
  Dim b, c As Boolean
  b = Not 4 <> 5
  Debug.Print "[1] a=" & a & " b=" & b
  c = a Or b
  MaLogique = "a" < "b" Xor c
  Debug.Print "[2] c=" & c & " MaLogique=" & MaLogique
End Function    'ExempleEB.xlsm
```

Fenêtre de débogage

```
Print 11 <= 12
True
r = MaLogique(False)
[1] a=False b=False
[2] c=False MaLogique=True
? r
True
```

Réponses VBA

Debug.Print

Ctrl-G (Windows) ou Ctrl-Cmd-G (MacOS)

Commandes usager

b = Not 4 <> 5 = Not True = False
c = a Or b = False Or False = False
MaLogique = “a” < “b” Xor c = True Xor False = True

COURS E1 : EXCEL VBA

3.1.6 Valeur et format des cellules

C2 <= 8

```
Workbooks("Classeur1.xlsx").Worksheets("Feuil1").Range("C2").Value = 8
Range("C2") = 8              'La propriété Value est optionnelle
Range(MaCellule) = 8         'Le nom MaCellule fait référence à C2
Range.Cells(2,3) = 8         'La méthode Cells permet d'utiliser des numéros de ligne et colonne
Cells(2,3) = 8               'La collection Cells peut être utilisée directement sans l'objet Range
Cells(1,1).Offset(1,2) = 8 'Le décalage demande l'addition des lignes et colonnes
```

C2:E3 <= 7

```
Range("C2:E3") = 7                      'C2:E3 = 7 (2 lig., 3 col.)
Range("A1:C2").Offset(1,2) = 7          'C2:E3 = (A1:C2) décalé de 1 lig., 2 col. = 7
Range(MaPlage) = 7                      'Le nom MaPlage fait référence à C2:E3
Range(Cells(2,3),Cells(3,5)) = 7        'C2:E3 = Cells(2,3) à Cells(3,5) = 7
Cells(2,3).Resize(1,2) = 7              'C2:E3 = C2 ajoute 1 lig., 2 colonnes = 7
Cells(1,1).Offset(1,2).Resize(1,2) = 7 '(C2 = A1 décalé de 1 lig., 2 col.) ajoute 1 lig., 2 col.
```

B2:B6 <= format

A1:E2 <= Brun

```
Range("B2:B6").Font.Bold = True       'Caractère gras de B2:B6
Range("B2:B6").Font.Italic = False    'Pas de caractère italic de B2:B6
Range("B2:B6").Font.Color = vbRed     'Caractère rouge de B2:B6
With Range("B2:B6").Font              'Idem aux lignes 1 à 3 sur B2:B6
  .Bold = True
  .Italic = False
  .Color = vbRed
End With
Cells(1,1).Resize(1,4).Interior.Color = RGB(213,153,49)   'Remplissage en brun de A1:E2
```

vbBlack : noir	**vbWhite** : blanc	**vbRed** : rouge	**vbGreen** : vert
vbBlue : bleu	**vbYellow** : jaune	**vbCyan** : cyan	**vbMagenta** : magenta

COURS E1 : EXCEL VBA

3.2.1 Décision If…End If

```
If expression_booléenne1 Then
  Instruction1
ElseIf expression_booléenne2 Then
  Instruction2
Else
  Instruction3
End If
```

```
Sub CelluleNonVide(Cellule AS String)   'Usage: CelluleNonVide("B1")
  If Not IsEmpty(Cellule) Then
    Debug.Print "Cellule " & Cellule & " est non-vide"
  End If
End Sub       'ExempleEB.xlsm
```

```
Sub AffichePositif(Ligne, Colonne AS Integer)   'Usage: Call AfficheP
  If Cells(Ligne,Colonne) > 0 Then
    Cells(Ligne,Colonne + 1) = "Positif"
  Else
    Cells(Ligne,Colonne + 1) = "Non positif"
  End If
End Sub       'ExempleEB.xlsm
```

COURS E1 : EXCEL VBA
3.3.1 Questionner l'usager (InputBox)

```
Dim Saisie As String
Saisie = InputBox("Question ? ", "Titre", "Réponse_par_défaut")
Debug.Print "Vous avez saisi : " & Saisie & "."
```

Utiliser **Saisie** de type Integer pour lire un entier!

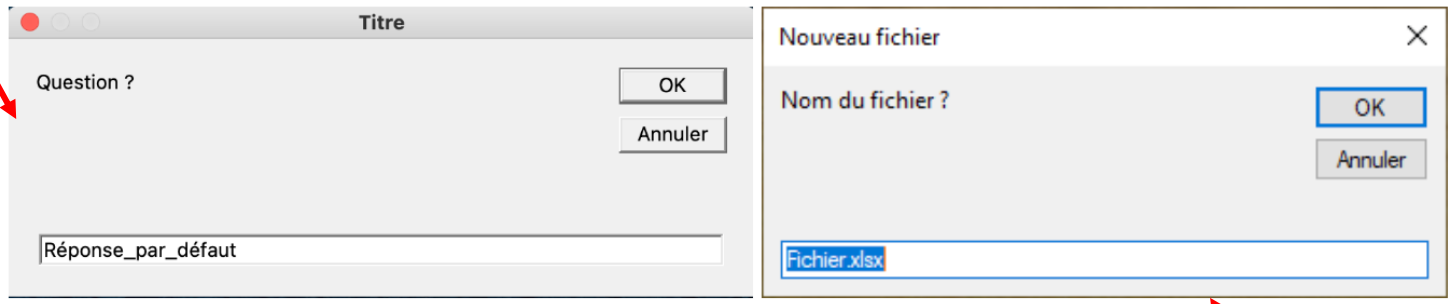


Fig. E.5 - Boîte de dialogue **InputBox** sous **MacOS** et **Windows**

```
Function LireNomFichier() AS String    'Usage: Nom = LireNomFichier()
  Dim NomFichier As String
  NomFichier = InputBox("Nom du fichier ? ", "Nouveau fichier", "Fichier.xlsx")
  If NomFichier = "" Then          'Vérifie si la saisie est vide
    Debug.Print "Opération annulée ou aucune saisie."
  Else
    Debug.Print "Le nom du fichier saisi est : " & NomFichier & "."
  End If
  LireNomFichier = NomFichier      'Retourne le nom saisi
End Sub
```

'ExempleEB.xlsm

```
Nom = LireNomFichier()
Le nom du fichier saisi est : Fichier.xlsx
```

COURS E1 : EXCEL VBA
3.3.2 Informer l'usager (MsgBox)

La variable **vbBoutons** de type **vbMsgBoxStyle** peut prendre les valeurs :

- **vbOKOnly** (0): Affiche le bouton **OK** seulement;
- **vbOKCancel** (1): Affiche les boutons **OK** et **Annuler**;
- **vbAbortRetryIgnore** (2): Affiche les boutons **Abandonner**, **Recommencer**, **Ignorer**;
- **vbYesNoCancel** (3): Affiche les boutons **Oui**, **Non** et **Annuler**;
- **vbYesNo** (4): Affiche les boutons **Oui** et **Non**;
- **vbRetryCancel** (5): Affiche les boutons **Recommencer** et **Annuler**;

et aussi ajouter les valeurs d'icônes suivantes :

- **vbCritical** (16): Affiche une icône d'erreur (X cercle rouge);
- **vbQuestion** (32): Affiche une icône de question (? cercle bleu);
- **vbExclamation** (48): Affiche une icône d'exclamation (! triangle Jaune);
- **vbInformation** (64): Affiche une icône d'information (i cercle bleu)

Style de boîte message

```
Dim vbBoutons As vbMsgBoxStyle
Dim vbReponse As vbMsgBoxResult
vbBoutons = ...
vbreponse = MsgBox("Message", vbBoutons, "Titre")
If vbReponse = ... Then
  actions ...
End If
```

Résultats de boîte message

La variable **vbReponse** de type **vbMsgBoxResult** peut prendre les valeurs :

- **vbOK** (1): Si le bouton **OK** est sélectionné;
- **vbCancel** (2): Si le bouton **Annuler** est sélectionné;
- **vbAbort** (3): Si le bouton **Abandonner** est sélectionné;
- **vbRetry** (4): Si le bouton **Recommencer** est sélectionné;
- **vbIgnore** (5): Si le bouton **Ignorer** est sélectionné;
- **vbYes** (6): Si le bouton **Oui** est sélectionné;
- **vbNo** (7): Si le bouton **Non** est sélectionné.

COURS E1 : EXCEL VBA
3.3.2 Informer l'usager (MsgBox)

```
MsgBox "Bonjour",, "Titre"                                                    1
MsgBox "Appuyer sur Enter pour continuer", vbOKOnly, "Pause temporaire"       2
Dim vbReponse As vbMsgBoxResult
vbReponse = MsgBox("Voulez-vous supprimer le fichier ?", vbYesNo + vbExclamation, "Confirmation")
                                                                                                3
Dim vbBoutons As vbMsgBoxStyle
vbBoutons = vbRetryCancel + vbQuestion
vbReponse = MsgBox("Fichier introuvable. Réessayer ?", vbBoutons, "Validation")  4
If vbReponse = vbRetry Then
  Debug.print "Recommancer la recherche"
Else
  Debug.Print "Abandonner la recherche"
End If          'ExempleEB.xlsm
```

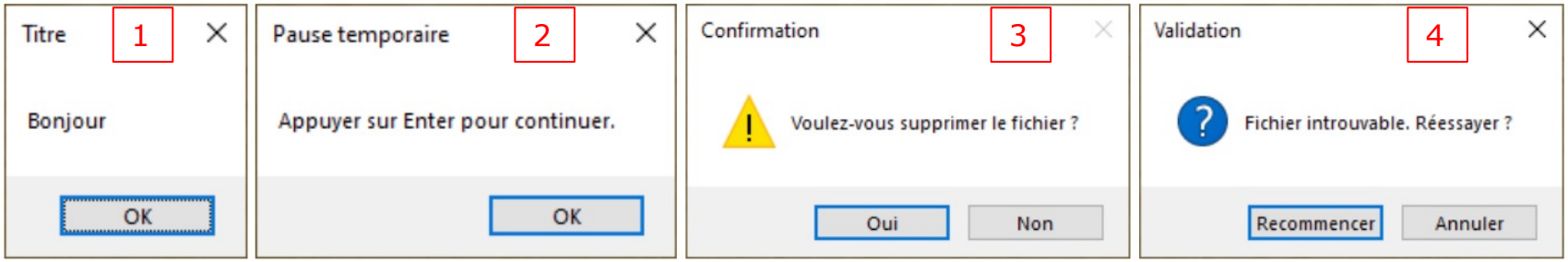


Fig. E.6 - Boîtes de dialogue **MsgBox** selon 4 styles différents sous **Windows**

COURS E1 : EXCEL VBA

3.4.1 Répétition (For Each…In…Next)

```
For Each element In Liste
  Instruction1
  Condition_de_sortie_immédiate Exit For
  Instruction2
Next element
```

Variable **Adresses** est une chaîne de caractères String
Variable **Plage** est un objet Range … donc assignation par Set
Variable **Cellule** est aussi un objet Range … assignation par Each et Next

```
Sub AfficheRouge(Adresses As String)   'Usage: AfficheRouge("B2:B6")
  Dim Cellule, Plage As Range
  Set Plage = Range(Adresses)
  For Each Cellule In Plage
    If Cellule > 50 Then
      Cellule.Font.Color = vbRed
    End If
  Next Cellule
End Sub    'ExempleEB.xlsm
```

	A	B	C	D	E
1	No Produit	Montrant			
2	S45-A	123 $			
3	P13-Z	65 $			
4	P14-Z	23 $			
5	V67-A	15 $			
6	B57-S	112 $			

Fig. E.8 - Résultats de l'exécution de la procédure **AfficheRouge("B2:B6")**

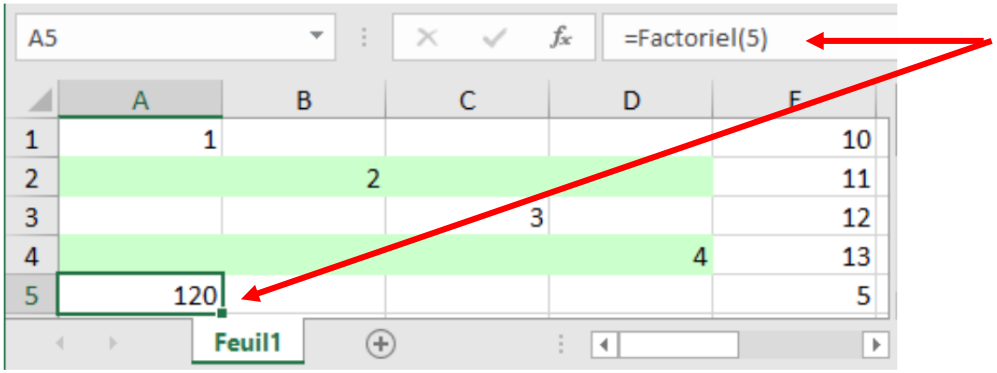
COURS E1 : EXCEL VBA
3.4.2 Répétition For…Next

```
Sub Diagonale(m As Integer)   'Usage: Diagonale(5)
  For i = 1 To m
    Cells(i,i) = i
  Next i
End Sub      'ExempleEB.xlsm
```

```
Function Factoriel(n As Long) As Long   'Usage: valeur = Factoriel(5)
  For i = n-1 To 1 Step -1
    n = n * i
  Next i
  Factoriel = n
End Functio  'ExempleEB.xlsm
```

Attention: une fonction déposée dans une cellule ne peut pas faire référence à elle-même.

```
Sub FiligrammePaire(c As Integer)   'Usage: FiligrammePaire(5)
  DerniereLigne = Cells(Rows.Count, c).End(xlUp).Row 'Trouve la dernière ligne de la colonne c
  For i = 2 To DerniereLigne Step 2
    Cells(i, 1).Resize(1, 4).Interior.ColorIndex = 35 'Étendre la plage à 1 ligne par 4 colonnes
  Next i
End Sub      'ExempleEB.xlsm
```



Nouvelle fonction Excel

Fig. E.9 - Résultats de l'exécution des procédures et la fonction à la cellule **A5**

COURS E1 : EXCEL VBA
Application 4.1

4.1 Intégrale d'une fonction f(x)

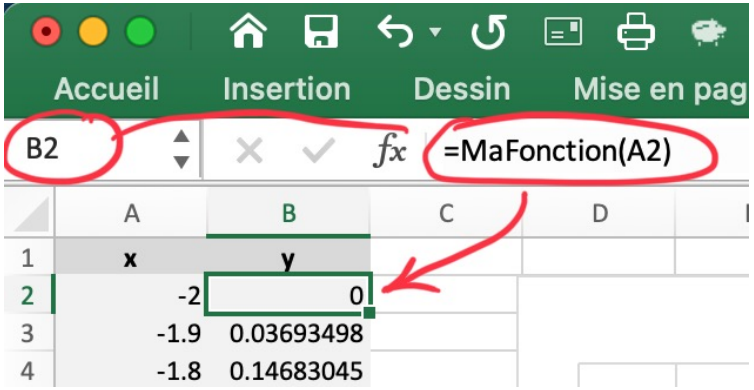
Un script VBA doit être préparé afin de calculer l'intégrale d'une fonction f(x) entre x1= -2 et x2= +2. Ainsi, nous avons :

f(x) = 3 cos(x a) + 3 pour x < -1 et x > 1;
f(x) = 3 autrement;

Fonction dans une cellule:
- pas de référence à d'autres cellules;
- pas d'interaction MsgBox() ou autres;
- pas de Debug.Print;

Sauf les entrées et sorties.

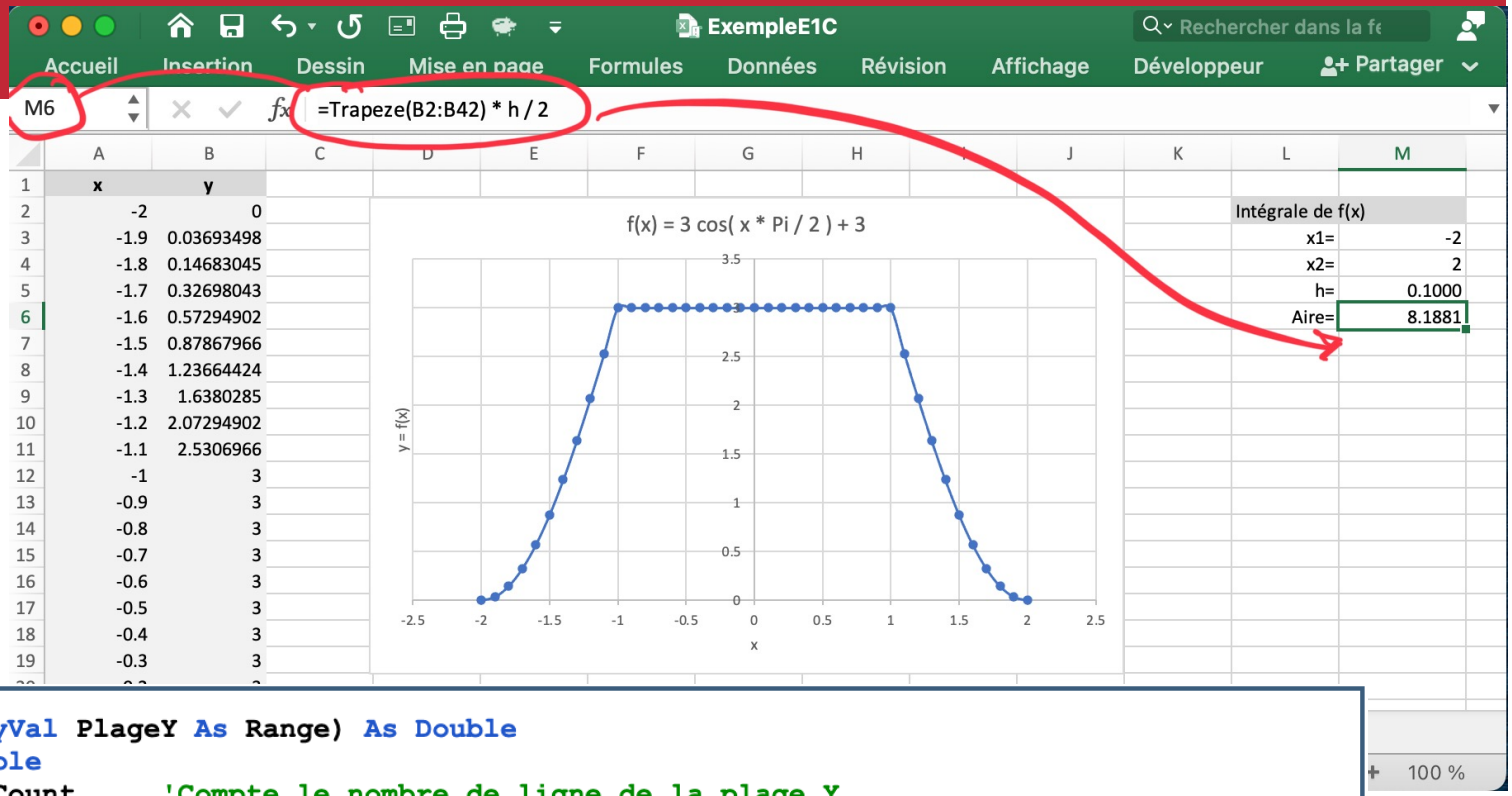
Objet Range comme paramètre d'entrée:
ByVal … référence par valeur avec source protégé;
ByRef … référence avec cellule source non protégé.




```
Function MaFonction(ByVal x As Range) As Double
  Dim a, dblPi As Double
  dblPi = WorksheetFunction.Pi()      'Valeur système pour Pi à 15 décimales
  a = dblPi / 2                       'Calcul de Pi / 2
  If x < -1 Or x > 1 Then             'Idem pour x.Value
    MaFonction = 3 * Cos( x * a) + 3  'Si x<-1 ou x>1
  Else
    MaFonction = 3                    'Si x>=-1 et x<=1
  End If
End Function    'ExempleEC.xlsm
```

COURS E1 : EXCEL VBA

Application 4.1



x	y
-2	0
-1.9	0.03693498
-1.8	0.14683045
-1.7	0.32698043
-1.6	0.57294902
-1.5	0.87867966
-1.4	1.23664424
-1.3	1.6380285
-1.2	2.07294902
-1.1	2.5306966
-1	3
-0.9	3
-0.8	3
-0.7	3
-0.6	3
-0.5	3
-0.4	3
-0.3	3

```
Function Trapeze(ByVal PlageY As Range) As Double
  Dim somme As Double
  n = PlageY.Rows.Count      'Compte le nombre de ligne de la plage Y
  somme = 0 : i = 0
  For Each y In PlageY       'Pour chaque cellule de la plage Y
    If i = 0 Or i = (n - 1) Then
      somme = somme + y      'Si premier ou dernier, on utilise 1 fois y (idem y.Value)
    Else
      somme = somme + 2 * y 'Sinon on utilise 2 fois y
    End If
    i = i + 1
  Next y
  Trapeze = somme
End Function    'ExempleEC.xlsm
```