Cours (excel et vba) : thème 2 - Question 3

Question 3 : La résolution de tous les problèmes de gestion est-elle automatisable ?

Notions abordées :
- synthèse des notions et techniques Excel utilisées jusqu'à ce jour ;
- dont : rappels de vocabulaire et rappels concernant les formules Excel à connaître ;
- dont : rappels concernant l'intégrité, la fiabilité et le contrôle des données (Thème 1, Question 1) ;
- introduction à la programmation sous Excel, i.e. découverte des notions de : variables et typages, structures alternatives (= conditionnelles), structures itératives, procédures et fonctions, événements.

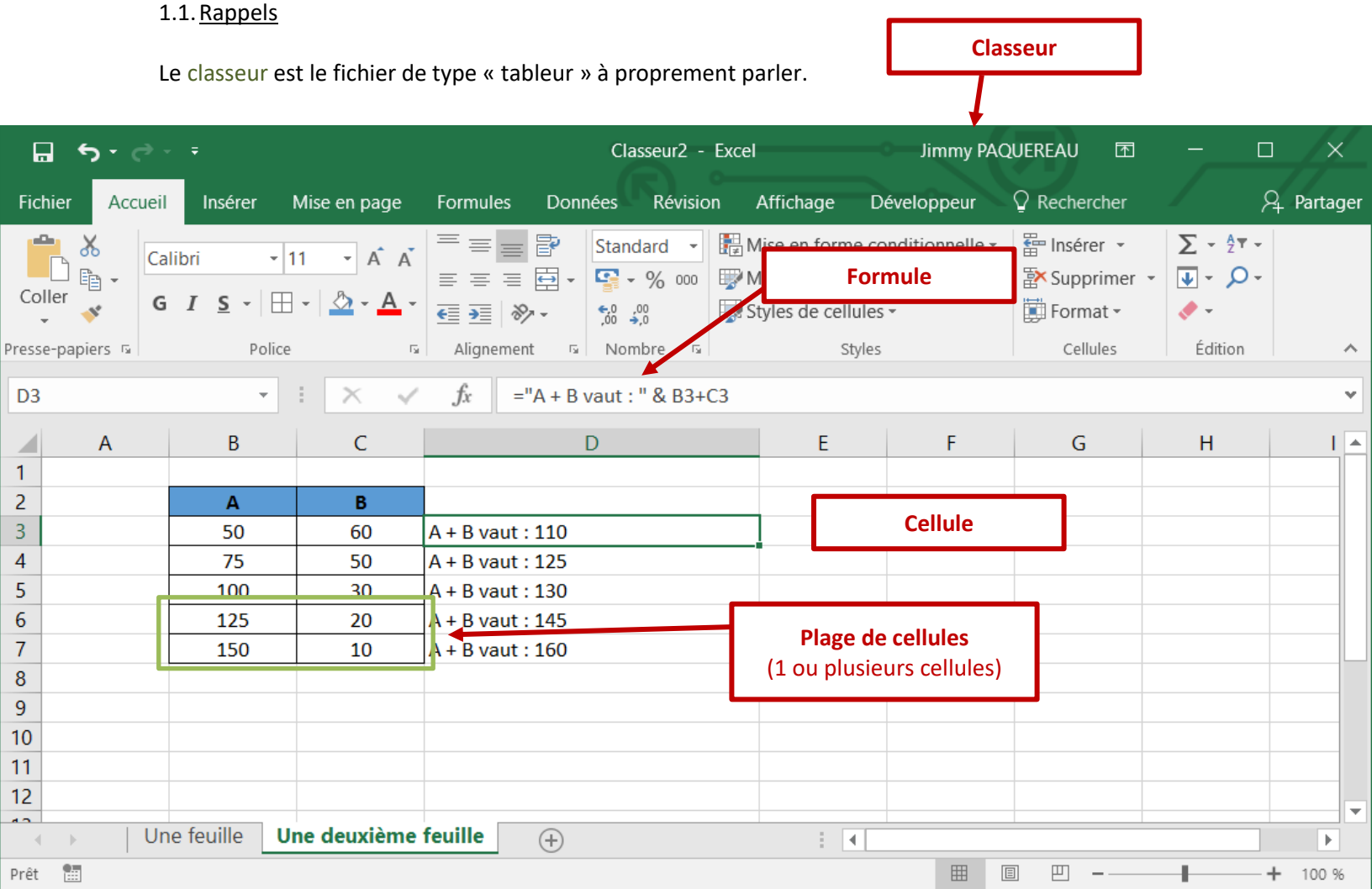
1. Préliminaire

Le présent cours constitue une première approche de la question 3 du programme. Il constitue ainsi une première approche de la programmation, des problématiques liées à l'automatisation des processus de gestion ainsi qu'au contrôle de l'intégrité des données. Les notions traitées au travers de ce cours seront réétudiées ultérieurement, au travers d'un nouveau cours et de nouvelles activités ainsi qu'au moyen d'autres outils, d'autres notions et d'autres technologies.

2. Tableur

1.1. Rappels

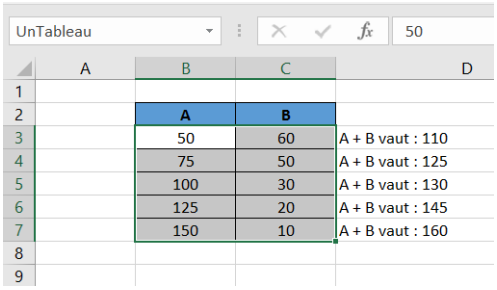
Le classeur est le fichier de type « tableur » à proprement parler.



Le classeur est divisé en une ou plusieurs feuille(s). Chaque feuille est constituée d'un tableau à deux dimensions, lui-même composé de cases appelées cellules. Un ensemble d'une ou plusieurs cellules contiguës (=voisines) formant un rectangle est appelé une plage de cellules.

Dans les cellules de type « standard » (voir 1.3. Formatage des données), il est possible d'insérer une formule. Une formule est une opération, un calcul, retournant un résultat qui peut se baser sur un ensemble de valeurs, y compris les valeurs d'une ou plusieurs autres cellules du classeur.

Il est possible d'identifier une cellule ou une plage de cellules à partir de :
- ses coordonnées ;
- à partir de son nom (si on nomme la cellule). On parle alors de cellule nommée et de plage nommée, ou plus généralement de référence.



Quelques exemples :
- Cellule « B3 » ou « 'Une deuxième feuille'!B3 » : contient la valeur 50 ;
- Plage « A2:C7 » ou plage « UnTableau » : liste des valeurs A et B (incluant la ligne d'en-têtes) ;
- Colonne « B:B » : toute la colonne ;
- Ligne « 2:2 » : toute la ligne.

Par ailleurs, on rappelle qu'un tableur permet d'étendre une sélection, c'est-à-dire de recopier la (ou les) formule(s) d'une sélection dans une ou plusieurs cellule(s) en faisant varier les indices/index. A cet égard, on retient qu'on peut utiliser le caractère $ (dollar) afin de figer l'indices des lignes et/ou des colonnes. On place alors le $ devant l'indice de la ligne et/ou de la colonne à figer.

Exemple :
- ligne 1 : on insère manuellement 0, 1, 2, …, 8 ;
- ligne 2 : on insère la formule « = B1 + A2 » dans la cellule B2. Puis on étend B2 jusqu'à I2 ;
- ligne 3 : on étend B2 à B3 puis B3 jusqu'à I3 ;
- ligne 4 : on insère la formule « = B1 + $A2 » dans la cellule B4. Puis on étend B4 jusqu'à I4 ;
- on obtient finalement le résultat ci-dessous.

	A	B	C	D	E	F	G	H	I
1	0	1	2	3	4	5	6	7	8
2	0	1	3	6	10	15	21	28	36
3	0	1	4	10	20	35	56	84	120
4	0	1	2	3	4	5	6	7	8

Explications :
- dans la formule « = B1 + A2 », l'indices des lignes et colonnes ne sont pas figés ;
- dans la formule « = B1 + $A2 », au contraire, on a figé une colonne ;
- en étendant les formules, le tableur « génère » ainsi les formules précisées ci-dessous.

Ligne/Colonne	**…**	**C**	**…**	**I**
1	…	2	…	8
2	…	= B2 + C1	…	= H2 + I1
3	…	= B3 + C2	…	= H3 + I2
4	…	= C1 + $A2	…	= C1 + $A2

1.2. Intégrité des données

Préserver l'intégrité des données, c'est faire de sorte que les données restent conformes à un référentiel. Autrement dit, contrôler l'intégrité des données, cela consiste à s'assurer que l'utilisateur ne puisse saisir des données incohérentes. Plus généralement, cela vise à s'assurer que les données dont on dispose restent conformes aux attentes, lesquelles sont typiquement décrites dans un cahier des charges et/ou une spécification ou documentation technique.

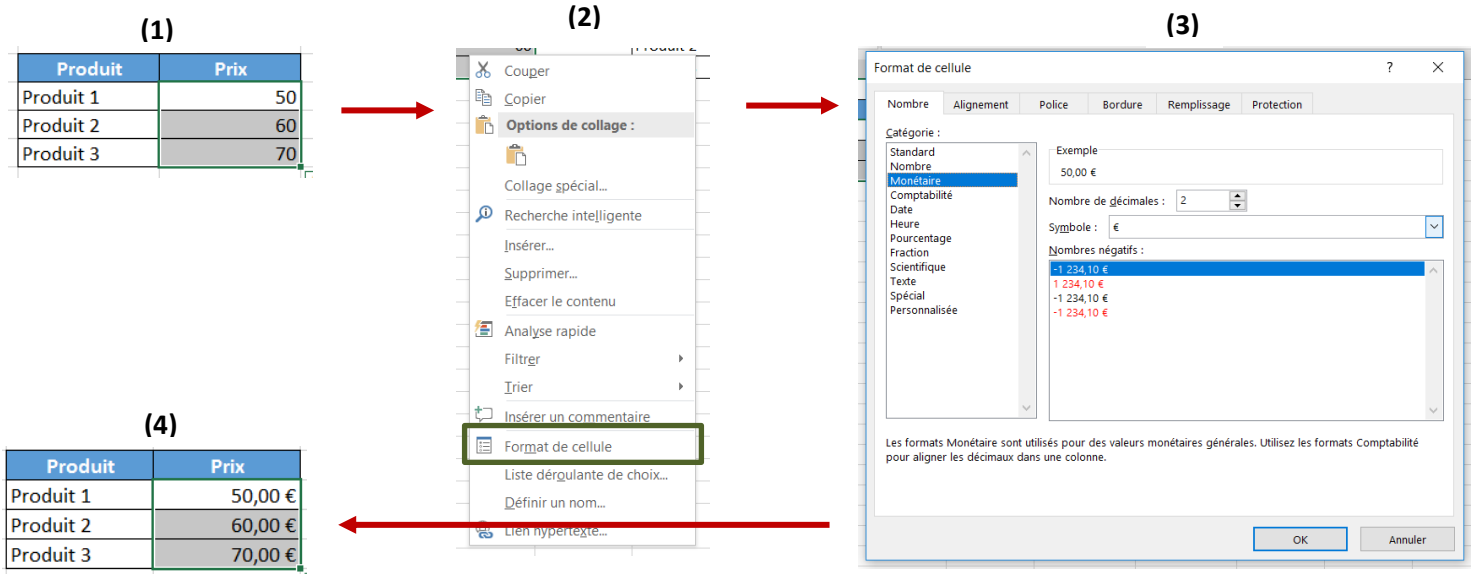
Exemple : lorsqu'un utilisateur saisit une facture, il ne doit pas pouvoir saisir des quantités ou des montants négatifs. Si on sauvegarde des factures, la date d'une facture ne doit ou ne devrait pas pouvoir être autre chose qu'une date.

Pour contrôle l'intégrité des données, Excel dispose d'un système de contrôle des saisies (validation des données) et de limitation des saisies (verrouillage des cellules et protection des feuilles et/ou du classeur). Il ne faut pas confondre validation des données et formatage des données.

1.3. Formatage des données

Le formatage des données répond à une problématique d'affichage. Comme cette expression le suggère, il s'agit de mettre en forme les données selon un certain format. Le formatage des données n'empêche aucunement l'utilisateur de saisir des données incohérentes !

Sous Excel, pour formater ses données :
(1) Sélection de la cellule ou de la plage de cellule à formater ;
(2) Clic droit puis clic sur « Format de cellule » ;
(3) Onglet « Nombre », choix du format de(s) cellule(s), clic sur « OK »
(4) Ça y est ! Les données sont formatées.

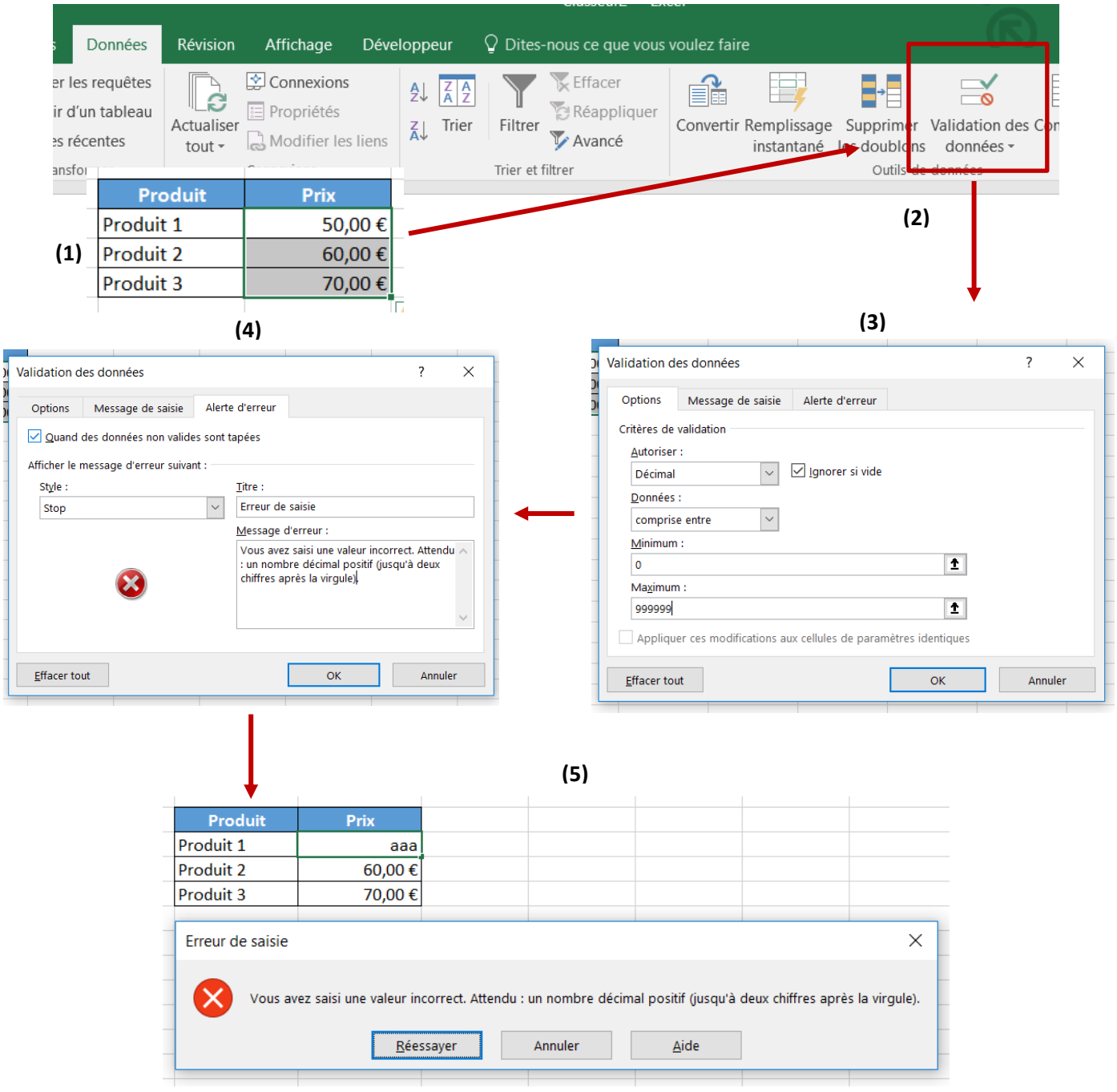


1.4. Validation des données

Contrairement au simple formatage des données, la validation des données (ou contrôle des saisies)

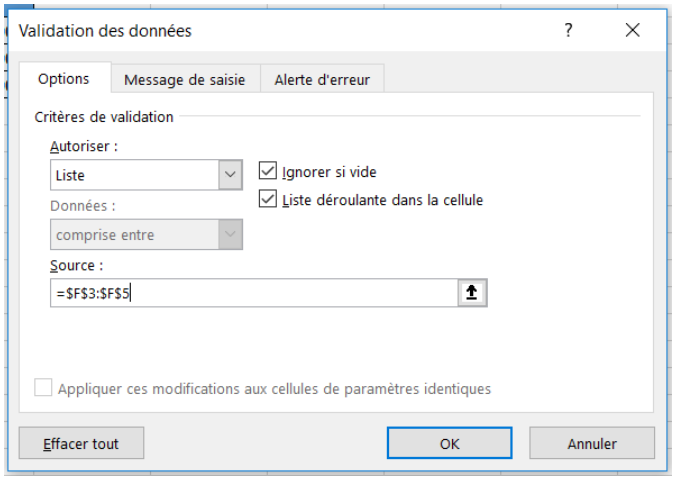
permet de s'assurer de l'intégrité des données, i.e. de fiabiliser les données. En effet, elle permet d'empêcher les saisies incohérentes.

Sous Excel, on peut ajouter quelques contrôles de saisies de la manière suivante :
(1) Sélection de la cellule ou plage de cellules à contrôler ;
(2) Clic sur l'onglet « Données », puis clic sur « Validation des données » ;
(3) Choix du type de validation ;
(4) Eventuellement (recommandé), configuration d'un message d'erreur, puis clic sur « OK » ;
(5) Ça y est, l'utilisateur ne peut saisir qu'une valeur respectant votre critère de validation.

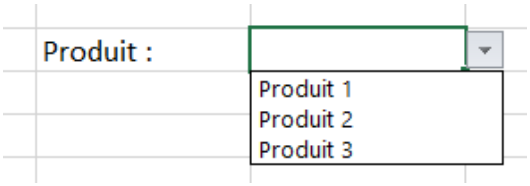


C'est également grâce aux validations de données que l'on peut, sous Excel, créer des listes déroulantes, très pratique pour s'assurer que l'utilisateur puisse saisir/sélectionner une valeur si et seulement elle fait partie d'une liste. On procède comme suit :
(1) Dans « Critère de validation », choisir « Liste » ;
(2) Compléter le champ « Source : » en précisant la plage de cellules dont les valeurs alimenteront la liste déroulante. Cliquer sur « OK ». De même que dans l'exemple précédent, on définira préférablement un message d'erreur ;
N.B. : en pratique, la fonction « DECALER » s'avère très utile pour définir la source de données d'une liste déroulante.

(1)



(2)

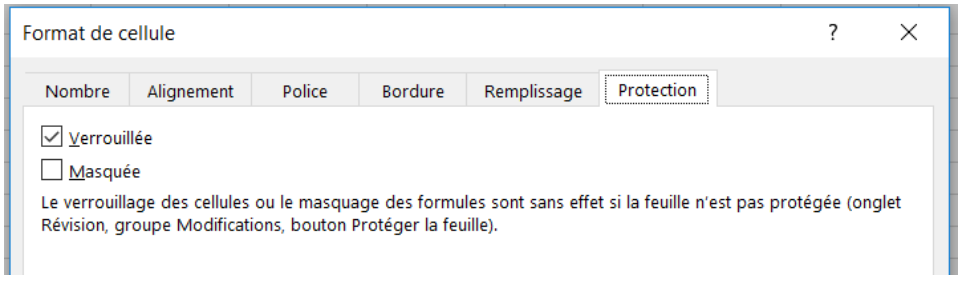


1.5. Limitation des saisies

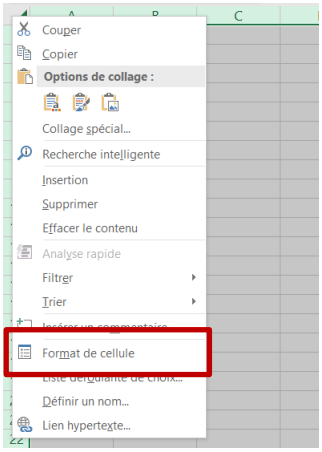
Dans le cadre de la conception d'un logiciel, afin de s'assurer de l'intégrité des données, il convient non seulement de contrôler les saisies mais encore de les limiter. En effet, un utilisateur ne doit pas ou ne devrait pas pouvoir passer outre son « périmètre d'utilisation ». Le principe à respecter est couramment le suivant : tout est interdit sauf ce qui est autorisé. Autrement dit, on interdit tout d'abord tout (accès aux fonctionnalités, saisies, etc.). On autorise ensuite l'utilisateur/acteur à effectuer uniquement ce dont il a besoin pour participer au processus. Ces limitations se traduisent en particulier par des restrictions de saisies.

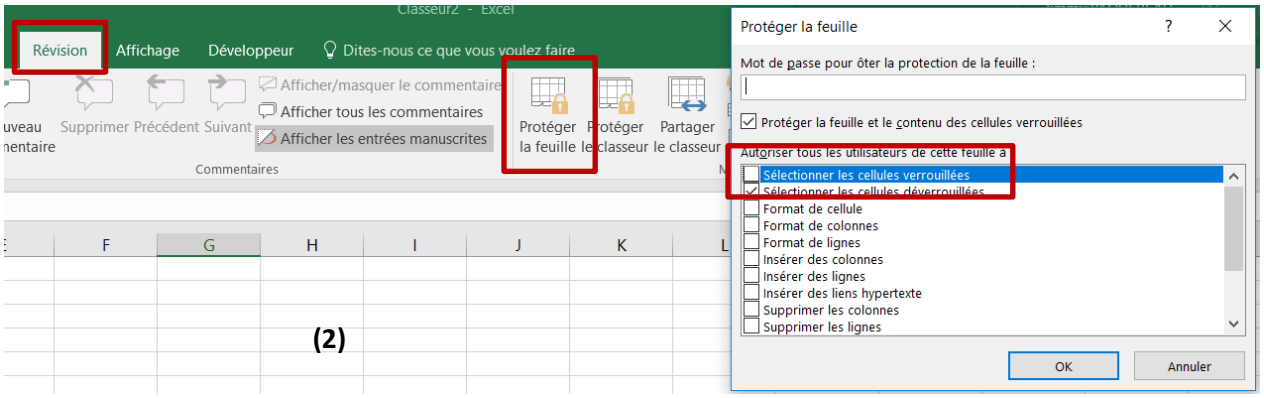
Sous Excel, on peut restreindre les saisies en procédant comme suit :
(1) Verrouiller une ou plusieurs (voire toutes) cellules du classeur : sélection des cellules, puis clic droit, puis clic sur « Format de cellule » (1.1). Onglet « Protection », puis cocher/décocher « Verrouillée » (1.2) ;
(2) Protéger les feuilles et/ou tout le classeur.

(1.2)



(1.1)

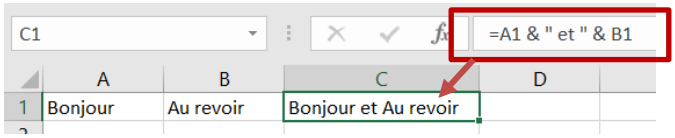




2. Tableur et formules

Les tableurs disposent de toute une panoplie de fonctions vous permettant de construire des calculs élaborés de manière à faciliter la création de calculs et de traitements automatisés. Nous traiterons les fonctions les plus usuelles.

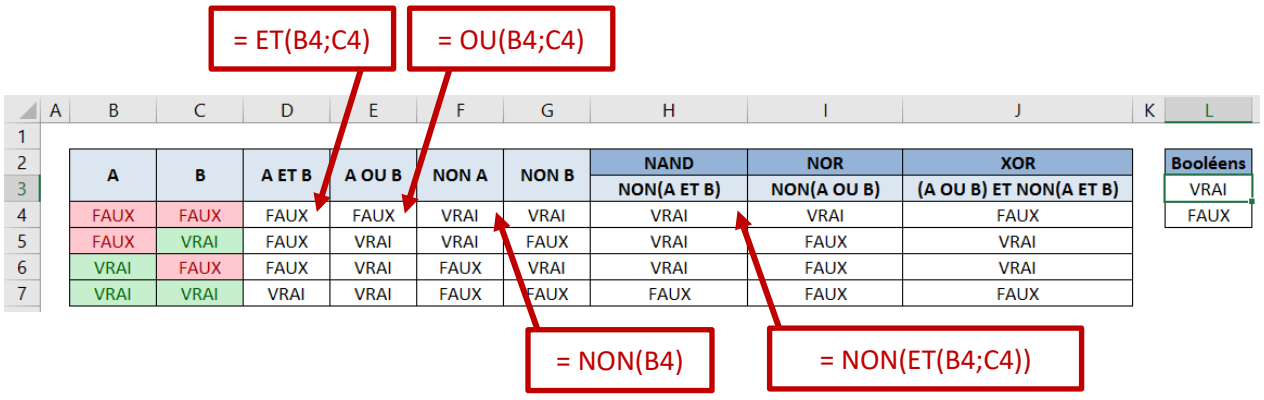
2.1. Concaténation, addition, multiplication, etc.



On appelle concaténation l'addition de chaînes de caractères. Elle consiste à mettre bout-à-bout deux chaînes de caractères. Pour ce faire, on utilise le caractère « & » (prononcé « et commercial »). On rappelle que toute chaîne de caractères doit être spécifiée entre guillemets !

Quant aux opérations algébriques usuelles, on utilise les caractères « * », « + », « - » et « / » pour effectuer respectivement des multiplications, additions, soustractions et divisions.

2.2. Opérateurs logiques (ET, OU, NON)



A	B	A ET B	A OU B	NON A	NON B	NAND NON(A ET B)	NOR NON(A OU B)	XOR (A OU B) ET NON(A ET B)
FAUX	FAUX	FAUX	FAUX	VRAI	VRAI	VRAI	VRAI	FAUX
FAUX	VRAI	FAUX	VRAI	VRAI	FAUX	VRAI	FAUX	VRAI
VRAI	FAUX	FAUX	VRAI	FAUX	VRAI	VRAI	FAUX	VRAI
VRAI	VRAI	VRAI	VRAI	FAUX	FAUX	FAUX	FAUX	FAUX

Booléens
VRAI
FAUX

2.3. Condition (SI)

La fonction SI(condition; valeurSiVrai; valeurSiFaux) permet de retourner une valeur si une condition est

vérifiée, une autre valeur dans le cas contraire (à savoir, lorsque la condition n'est pas vraie). La fonction SIERREUR(valeur; valeurSiErreur) permet de retourner une valeur si le résultat d'une formule n'est pas une erreur, une autre valeur s'il y a bien une erreur.

Pour construire une condition, on peut utiliser plusieurs opérateurs :
- les opérateurs logiques : ET, OU, NON ;
- les opérateurs de comparaison : >, >=, =, <, <= ;
- des fonctions retournant vrai ou faux. Exemple : ESTERREUR (permet de tester si une formule est en erreur).

= A2/B2

= SIERREUR(A2/B2; "Interdit")

	A	B	C	D	E
1	A	B	A / B	A / B (erreur gérée)	Positif, négatif ou nul
2	12	6	2	2	Positif
3	5	0	#DIV/0!	Interdit	
4	0	5	0	0	Positif
5	-12	6	-2	-2	Négatif

```
= SI (
   D2="Interdit"; "";
   SI(
      D2>0; "Positif";
      SI(D2=0; "Nul"; "Négatif")
   )
)
```

N.B. :
- on rappelle qu'on ne peut pas diviser des nombres réels par 0 ;
- on retiendra bien qu'il est tout à fait possible d'imbriquer des SI les uns dans les autres (SI imbriqués).

2.4. Fonctions statistiques (NB, NBVAL, NB.SI, SOMME, SOMME.SI, SOMME.SI.ENS., MOYENNE)

- NB(plage) : la fonction NB retourne le nombre de nombres que contient une plage de cellules.
Exemple : en reprenant l'exemple ci-dessus, NB($A:$A) retourne 4. Il y a 4 nombres dans la colonne : 12, 5, 0 et -12.

- NBVAL(plage) : la fonction NBVAL retourne le nombre de cellules non vides d'une plage de cellules.
Exemple : en reprenant l'exemple ci-dessus, NBVAL($A:$A) retourne 5. En effet, les cellules A1 à A5 sont non vides.

- NB.SI(plage; critère) : la fonction NB.SI permet de retourner le nombre de cellules d'une plage de cellules vérifiant une condition précisée dans une chaîne de caractères.
Exemple : en reprenant l'exemple ci-dessus, NB.SI($A:$A, ">0") retourne 2. Il y a en effet 2 nombres strictement supérieurs à 0 : 12 et 5.

- SOMME(plage) : la fonction SOMME retourne la somme des nombres contenus dans une plage de cellules.
Exemple : SOMME($A:$A) retourne 5. En effet : 12+5+0-12 = 5.

- SOMME.SI(plage; critère) : la fonction SOMME.SI permet de retourner la somme des nombres contenus dans une plage de cellules en se cantonnant aux nombres vérifiant une condition précisée dans une chaîne de caractères.
Exemple : SOMME.SI($A:$A; ">0") retourne 17. En effet, seuls 12 et 5 sont >0 et on a : 12+5 = 17.

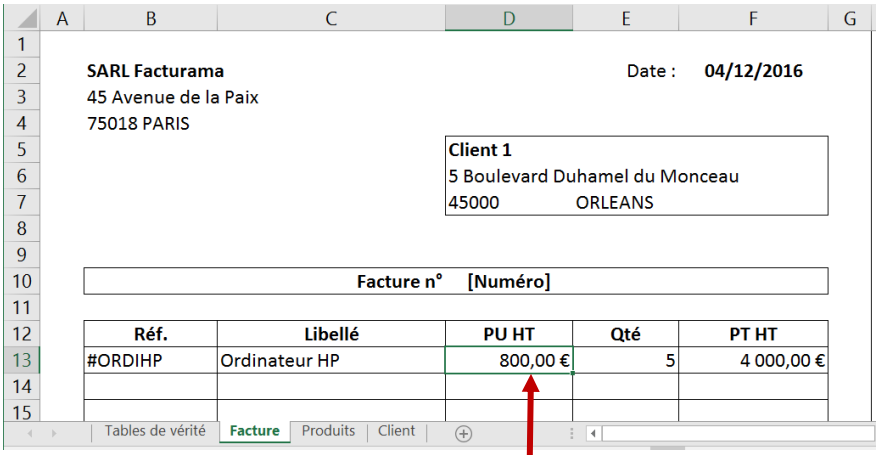
- SOMME.SI(plage; critère; plageSomme) : cette variante de la fonction SOMME.SI, plus générale, permet de retourner la somme des nombres contenus dans une plage de cellules si une condition est vérifiée.
Exemple : SOMME.SI($E:$E; "Positif"; $A:$A) retourne 12. En effet, seules E2 et E5 contiennent la valeur « Positif ». La somme porte donc sur A2 et A5, soit 12+0 = 12.

La fonction SOMME.SI.ENS est similaire à la fonction SOMME.SI à ceci près qu'elle permet d'effectuer une somme si plusieurs conditions sont vérifiées. Finalement, la fonction MOYENNE permet de calculer une simple moyenne arithmétique (non pondérée). Pour calculer une moyenne pondérée, on pourra regarder du côté de la fonction SOMMEPROD.

2.5. Autres fonctions usuelles

Parmi les autres fonctions à connaître :

- la fonction RECHERCHEV(critère; plage; colonne; FAUX) et *a fortiori* RECHERCHEH ;



SARL Facturama
45 Avenue de la Paix
75018 PARIS

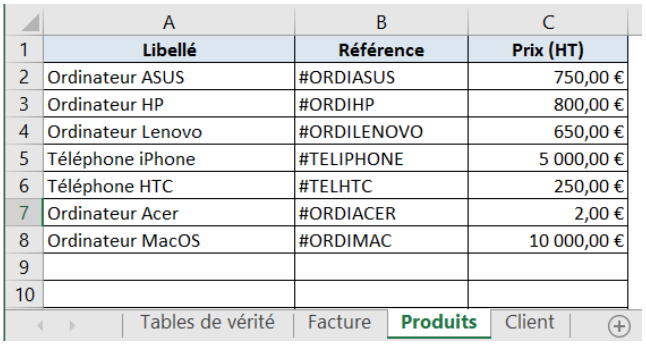
Date : 04/12/2016

Client 1
5 Boulevard Duhamel du Monceau
45000 ORLEANS

Facture n° [Numéro]

Réf.	Libellé	PU HT	Qté	PT HT
#ORDIHP	Ordinateur HP	800,00 €	5	4 000,00 €

Tables de vérité | Facture | Produits | Client



Libellé	Référence	Prix (HT)
Ordinateur ASUS	#ORDIASUS	750,00 €
Ordinateur HP	#ORDIHP	800,00 €
Ordinateur Lenovo	#ORDILENOVO	650,00 €
Téléphone iPhone	#TELIPHONE	5 000,00 €
Téléphone HTC	#TELHTC	250,00 €
Ordinateur Acer	#ORDIACER	2,00 €
Ordinateur MacOS	#ORDIMAC	10 000,00 €

Tables de vérité | Facture | Produits | Client

```
= SIERREUR(
   RECHERCHEV(C13; Produits!$A:$C; 3; FAUX);
   ""
)
```

N.B. : si C13 est vide (comme c'est le cas pour C14), RECHERCHEV est en erreur, d'où l'intérêt du SIERREUR. En Effet, s'il y a erreur, on retourne "" (chaîne vide). Le RECHERCHEV va en l'occurrence chercher C13 (ici « Ordinateur HP ») dans la première colonne de la plage Produits!$A:$C. Il trouve la ligne 3. Il retourne alors, comme demandé, la valeur contenue dans la 3ème colonne de cette même ligne (ici la valeur de Produits!C3, soit 800,00€).

- la fonction DECALER(plage; ligne; colonne; hauteur ; largeur) permet d'extraire un sous-tableau, à savoir une plage de cellules au sein d'une plage de cellules.

Exemples : en reprenant la feuille « Produits » ci-dessus, la formule suivante permet de récupérer exactement la liste de tous les libellés de produits.
Formule : DECALER(Produits!$A:$A; 1; 0; NBVAL(Produits!$A:$A)-1; 1)
Explication :
- on prend la colonne $A:$A. On décale d'une ligne vers le bas. On décale de 0 colonne vers la droite. On récupère NBVAL(Produits!$A:$A)-1 lignes, et 1 colonne. On obtient ainsi notre liste de libellés ;
- NBVAL(Produits!$A:$A) correspond au nombre de cellules non vides de la colonne A, donc 8 ;
- on retire 1 car le 8 obtenu inclus l'en-tête, soit finalement 7 lignes à récupérer.

3. Programmation

3.1. Qu'est-ce qu'un algorithme ?

Un algorithme est un enchaînement ordonné de traitements élémentaires, appelés instructions, servant à solutionner un problème donné. Ainsi, tout projet logiciel comporte *a minima* une phase consistant à développer des algorithmes. Cette phase (voir **Thème 4, Question 9**) est communément appelée phase de codage.

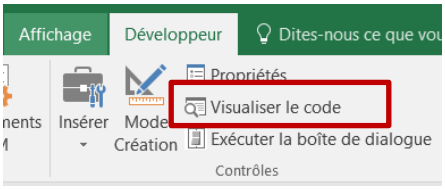
Un programme est un ensemble d'algorithmes écrit dans un certain langage, appelé langage de programmation. Nous nous intéresserons ici au langage VBA (*Visual Basic for Applications*), lequel peut être utilisé entre autres sous Excel. On notera qu'un algorithme Excel (fonction ou procédure, voir plus loin) est qualifié de macro.

En outre, on peut également percevoir un algorithme comme le moyen pour un programmeur de dire à un ordinateur ce qu'il doit faire.

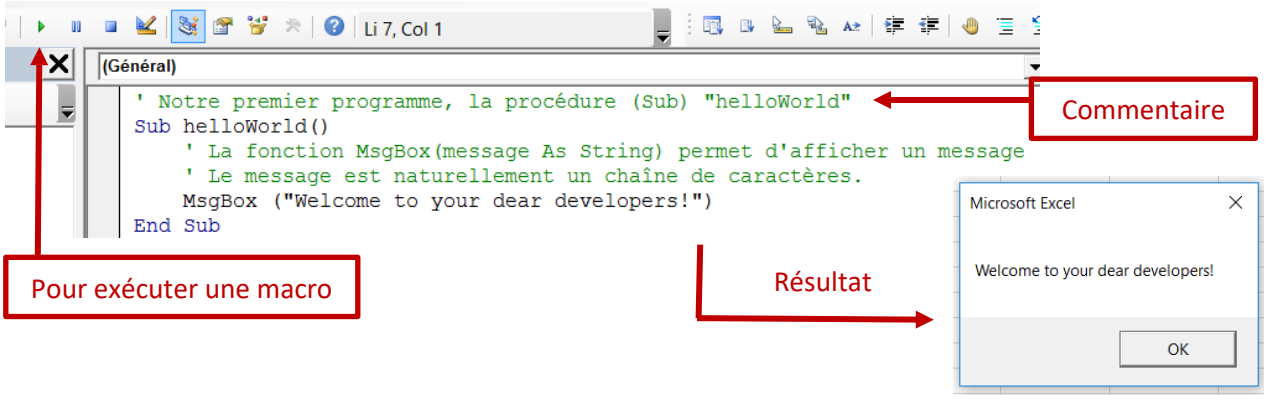
3.2. Environnement de développement intégré

Excel comporte un Environnement de Développement Intégré (EDI), plus souvent appelé IDE pour *Integrated Development Environment*. Et c'est via l'IDE que l'on peut coder.

Pour y accéder, il faut avant toute chose afficher l'onglet développeur. La documentation est disponible en ligne sur le support Microsoft Office (rechercher « afficher onglet développeur » sur Google). Ceci fait, dans l'onglet « Développeur », cliquer sur « Visualiser le code » pour rejoindre l'IDE.



Un premier programme...



3.3. Modifier une cellule ou une plage de cellules

Tout ce qu'Excel permet de faire manuellement, il permet de le faire en programmant. En particulier, vous pouvez modifier la valeur, la formule ou encore la couleur d'une cellule.
Pour ce faire, vous disposez des objets Cells et Range.

```
Sub modifionsDesCellules()
    ' Cells(ligne, colonne).Value permet de modifier le contenu d'une cellule
    ' Cells(2, 1) correspond à la cellule A2 (2ème ligne, 1ère colonne)
    Cells(2, 1).Value = "Je suis A2"
    ' Cells(ligne, colonne).Interior.Color permet de modifier la couleur
    ' d'arrière-plan d'une cellule.
    ' RGB(rouge, vert, bleu) permet de récupérer une couleur.
    ' RGB attend 3 nombres compris entre 0 et 255.
    Cells(2, 1).Interior.Color = RGB(255, 0, 0)
    ' On peut également utiliser l'objet Range et lui passer l'adresse de la
    ' ou des cellules.
    Range("B2").Value = "I'm B2"
    Range("A1:B1").Value = "J'appartient à A1:B1"
End Sub
```

Résultat

	A	B
1	J'appartient à A1:B1	J'appartient à A1:B1
2	Je suis A2	I'm B2
3		
4		

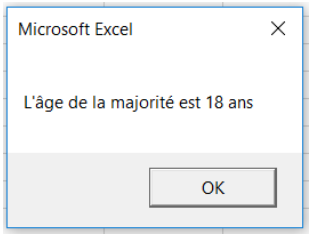
3.4. Variables et typages

De même qu'on dispose de cellules sous Excel, identifiées par leur adresse (C6, H12, O6, etc.) et dans lesquelles on peut placer et faire varier des valeurs, un algorithme peut créer ses « cellules ». En fait, il s'agit pour un algorithme de créer ce qu'on appelle des variables. On parle de déclaration de variable(s).

Une variable est un espace mémoire auquel on donne un nom et dans lequel on peut placer une valeur. Le fait de changer la valeur d'une variable s'appelle une affectation. Le qualificatif « variable » est très juste dans le sens où il est possible de changer (faire varier) la valeur d'une variable et dans la mesure où une variable a un type. En effet, un ordinateur ne stocke pas de la même manière une date, un entier, une chaîne de caractères ou encore un nombre à virgule. C'est pourquoi nombreux sont les langages (dits « fortement typés ») qui imposent au programmeur de préciser le type de chaque variable.

```
Sub desVariables()
    ' Déclaration d'un entier long (un grand entier)
    Dim unEntier As Long
    ' Déclération d'une chaîne de caractère
    Dim uneChaine As String
    ' Affectation de 18 à la variable unEntier
    unEntier = 18
    ' Affectation d'une chaîne à la variable uneChaine
    ' avec une concaténation.
    uneChaine = "L'âge de la majorité est " & unEntier & " ans"
    MsgBox (uneChaine)
End Sub
```

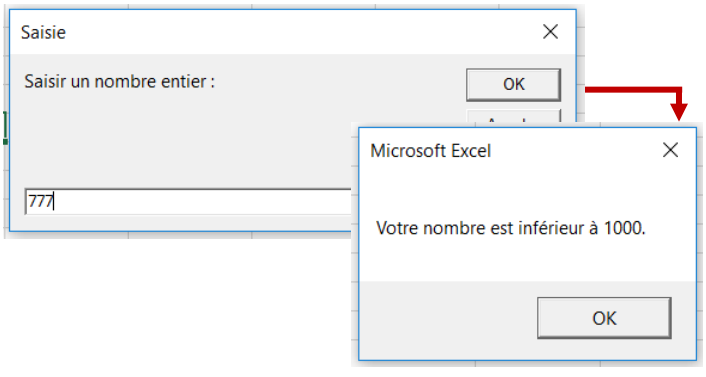
Résultat



3.5. Structures alternatives (if)

La structure alternative ou structure conditionnelle permet d'exécuter un ou plusieurs traitements sous réserve qu'une condition soit vérifiée. En VBA, la structure par excellence pour exécuter un traitement conditionnel est le bloc If … Then … ElseIf … Then … Else … End If.

```
Sub desConditions()
    Dim nombre As Long
    ' La fonction InputBox(message, titre) retourne une saisie
    ' utilisateur.
    nombre = InputBox("Saisir un nombre entier :", "Saisie")
    If nombre < 0 Then
        MsgBox ("Votre nombre est inférieur à 0.")
    ElseIf nombre < 100 Then
        MsgBox ("Votre nombre est inférieur à 100.")
    ElseIf nombre < 1000 Then
        MsgBox ("Votre nombre est inférieur à 1000.")
    Else
        MsgBox ("Votre nombre est supérieur ou égal à 1000.")
    End If
End Sub
```



3.6. Structures itératives (for, while)

Une structure itérative ou plus simplement une boucle permet de répéter un ou plusieurs traitements. La boucle tant que (*while*) permet de répéter un traitement tant qu'une condition est vérifiée. On utilise cette boucle entre autres quand on ne sait pas le nombre de fois qu'il faut répéter un ensemble de traitements.
La boucle pour (*for*) permet de répéter un traitement un nombre de fois déterminé. On utilise en outre cette boucle lorsque l'on sait le nombre de fois qu'on doit répéter un traitement. La boucle pour est basée sur un compteur qui augmente (=est incrémenté) ou diminue (=est décrémenté) à chaque tour de boucle. L'augmentation ou la diminution est appelée le pas (*step*) ou encore l'incrément.
Les compteurs sont particulièrement utilisés pour parcourir des ensembles, par exemple pour parcourir une liste de cellules (une boucle) ou encore un tableau de cellules (deux boucles imbriquées).

```
Sub unForEtUnWhile()
    Dim i As Long
    ' i va valoir successivement : 0, 2, 4, 6, 8, ..., 48, 50
    ' On va donc faire 25+1 = 26 tours de boucles
    For i = 0 To 50 Step 2
        ' On écrit successivement dans les cellules A1, A2, A3, ..., A51
        Cells(i + 1, 1).Value = "i vaut : " & i
    Next
    ' Même principe avec while
    i = 0 ' initialise le compteur
    Do While i <= 50 ' condition de sortie
        ' On écrit successivement dans les cellule B1, B2, B3, ..., B51
        Cells(i + 1, 1).Value = "i vaut : " & i
        i = i + 2 ' incrémentation
    Loop

End Sub
```

Résultat

	A	B
1	i vaut : 0	i vaut : 0
2	i vaut : 2	i vaut : 2
3	i vaut : 4	i vaut : 4
4	i vaut : 6	i vaut : 6
5	i vaut : 8	i vaut : 8
6	i vaut : 10	i vaut : 10
7	i vaut : 12	i vaut : 12
8	i vaut : 14	i vaut : 14
9	i vaut : 16	i vaut : 16
10	i vaut : 18	i vaut : 18
11	i vaut : 20	i vaut : 20
12	i vaut : 22	i vaut : 22
13	i vaut : 24	i vaut : 24
14	i vaut : 26	i vaut : 26
15	i vaut : 28	i vaut : 28
16	i vaut : 30	i vaut : 30
17	i vaut : 32	i vaut : 32
18	i vaut : 34	i vaut : 34
19	i vaut : 36	i vaut : 36
20	i vaut : 38	i vaut : 38
21	i vaut : 40	i vaut : 40
22	i vaut : 42	i vaut : 42
23	i vaut : 44	i vaut : 44
24	i vaut : 46	i vaut : 46
25	i vaut : 48	i vaut : 48
26	i vaut : 50	i vaut : 50

3.7. Procédures et fonctions

A bien y repenser, nous en avons déjà utilisées… Juste au-dessus, on a défini par exemple la procédure *unForEtUnWhile()*. Nous avons également utilisé la fonction *InputBox* pour récupérer une saisie. Dans vos formules Excel même, vous utilisez les fonctions RECHERCHEV ou encore SOMME.

Procédures et fonctions sont des algorithmes nommés et réutilisables. Procédures et fonctions peuvent avoir des paramètres. Par exemple, lorsque vous appelez la procédure *MsgBox*, vous lui passez en paramètre une chaîne de caractères, à savoir le message à afficher. La différence entre une fonction et une procédure réside dans le fait qu'une procédure ne retourne aucun résultat tandis qu'une fonction retourne un résultat d'un certain type. Par exemple, on peut récupérer la saisie de l'utilisateur lorsqu'on appelle la fonction *InputBox* parce que la fonction *InputBox* retourne cette valeur.

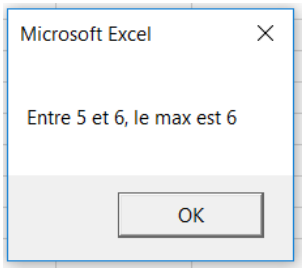
De même que l'on doit (du moins en VBA) normalement préciser le type d'une variable, l'on doit préciser

le type des paramètres d'une procédure ou d'une fonction. Par ailleurs, il faut préciser le type de la valeur de retour d'une fonction.

```
' Utilise la fonction "maximum" définie ci-après
Sub quelEstLeMaximum()
    Dim nb1 As Long
    Dim nb2 As Long
    Dim leMax As Long
    nb1 = 5
    nb2 = 6
    ' Appel de la fonction "maximum"
    leMax = maximum(nb1, nb2)
    MsgBox ("Entre " & nb1 & " et " & nb2 & ", le max est " & leMax)
End Sub

' La fonction maximum prend en paramètres deux nombres
' et retourne le maximum des deux.
Function maximum(nombre1 As Long, nombre2 As Long) As Long
    If nombre1 >= nombre2 Then
        ' Retourne nombre1
        maximum = nombre1
    Else
        ' Retour nombre2 dans le cas contraire
        maximum = nombre2
    End If
End Function
```

Résultat



Remarque ! Le fait de bien mettre des décalages (tabulations) vers la droite dans le code source s'appelle l'indentation. Cela améliore la lisibilité du code. Bien indenter et bien commenter vos codes est essentiel !

3.8. Evénements

Outre le fait qu'il soit possible de demander « manuellement » l'exécution d'une macro, il est encore possible d'exécuter une macro lorsqu'un événement se produit. Un événement peut être : un clic sur un bouton, l'ouverture du classeur, l'activation d'une feuille, la modification de la valeur d'une cellule, etc. Il est possible de réagir à ces événements.

On retiendra en particulier ce cas : le clic sur un bouton. Sous Excel, il est possible d'attacher une procédure au clic sur un bouton. On procède comme suit :