Excel
Programmation VBA
Visual Basic pour Applications

Hervé Hocquard

http://www.labri.fr/perso/hocquard

Programmation sous Excel via VBA (Visual Basic pour Applications)

Fonctions personnalisées
Complètement standardisée. Valable pour les autres classeurs et même, si pas d'accès aux objets spécifiques d'Excel, pour les autres outils Office.

Macros
Manipulation directe des objets Excel (classeurs, feuilles, cellules, graphiques, etc.)

Programmation de tâches
Programmation de séquences d'actions plus ou moins complexes, faisant intervenir ou non des mécanismes algorithmiques.

Interfaces graphiques
Boîtes de dialogues standards. Mais aussi les formulaires personnalisées pour faciliter les accès aux fonctionnalités. Nécessite une certaine formalisation et la connaissance des principes de la programmation évènementielle.

Enregistreur de macros
Transformation de séquences d'action en programme VBA. Ne nécessite pas la connaissance de la programmation, mais peu de souplesse (structure fixe, peu adaptable…)

Programmation des macros
Très puissant. Nécessite la connaissance des principes de la programmation et de la syntaxe d'accès aux objets Excel.

Points importants. Connaissance de l'algorithmie, langage de programmation Visual Basic. Les instructions sont écrites dans des fonctions (*function*) et procédures (*sub*), qui sont regroupées dans des modules. Nous travaillons dans VBE (Visual Basic Editor).

Généralités sur la programmation

ALGORITHMIE - PROGRAMMATION

Algorithmie

• Solution « informatique » relative à un problème

• Suite d'actions (instructions) appliquées sur des données

• 3 étapes principales :

1. saisie (réception) des données
2. Traitements
3. restitution (application) des résultats

Programme

• Transcription d'un algorithme avec une syntaxe prédéfinie

• **Visual Basic pour Applications**

• Même principes fondamentaux que les autres langages objets (Java, C#, etc.)

• VBA agit en interaction avec les fonctions prédéfinies disponibles dans la suite Office

Langage interprété : + portabilité application ; - lenteur (R, **VBA**, Python…)

Langage compilé : + rapidité ; - pas portable

(solution possible : write once, compile anywhere ; ex. Lazarus)

Langage pseudo-compilé : + portabilité plate-forme ; - lenteur (?)

(principe : write once, run anywhere ; ex. Java et le principe JIT)

VBA (Visual Basic pour Applications) est un langage de programmation dédié principalement aux applications Microsoft Office. Il est basé sur le langage Visual Basic, mais ne peut s'exécuter que dans une application hôte Microsoft Office, et non de manière autonome.

1. Déterminer les besoins et fixer les objectifs : que doit faire le logiciel, dans quel cadre va-t-il servir, quels seront les utilisateurs types ? On rédige un cahier des charges avec le commanditaire du logiciel (Remarque : commanditaire = **maître d'ouvrage** ; réalisateur = **maître d'œuvre**)
2. Conception et spécifications : quels sont les fonctionnalités du logiciel, avec quelle interface ?
3. Programmation : modélisation et codage
4. Tests : obtient-on les résultats attendus, les calculs sont corrects, y a-t-il plantage et dans quelles circonstances ? (tests unitaires, tests d'intégration, etc.)
5. Déploiement : installer-le chez le client (vérification des configurations, installation de l'exécutable et des fichiers annexes, etc.)
6. Maintenance : corrective, traquer les bugs et les corriger (patches) ; évolutive (ajouter des fonctionnalités nouvelles au logiciel : soit sur l'ergonomie, soit en ajoutant de nouvelles procédures)

Programme : suite d'instructions manipulant des données

LANGAGE VISUAL BASIC

Données typées. Visual Basic propose les types usuels de la programmation : entier, réels, booléens, chaîne de caractères.

Structures avancées de données. Gestion des collections de valeurs (énumérations, tableaux) et des objets structurés (enregistrements, classes).

Séquences d'instructions, c'est la base même de la programmation, pouvoir écrire et exécuter une série de commandes sans avoir à intervenir entre les instructions.

Structures algorithmiques : les branchements conditionnels et les boucles.

Les outils de la programmation structurée : pouvoir regrouper du code dans des procédures et des fonctions. Organisation du code en modules et possibilité de distribuer ces dernières.

Visual Basic n'est pas « case sensitive », il ne différencie pas les termes écrits en minuscule et majuscule.

Le type de données définit le type d'opérateurs qu'on peut leur appliquer.

- **Numérique** qui peut être réel (double) ou entier (long). Les opérateurs applicables sont : +, -, *, / (division réelle), \ (division entière), mod (modulo)

Exemple : 5 / 2 → 2.5 ; 5 \ 2 → 2 ; 5 mod 2 → 1

- **Booléen** (boolean) qui ne prend que deux valeurs possibles : True et False. Les opérateurs sont : not, and, or.

Exemple : True and False → False

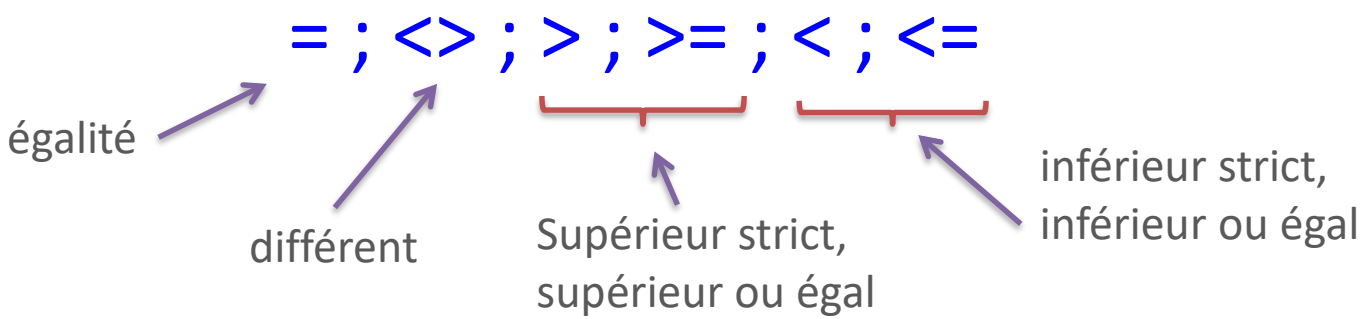
- **Chaîne de caractères** (string) qui correspond à une suite de caractères délimitée par des guillemets '' ''. Les opérateurs possibles sont la concaténation, la suppression d'une sous-partie, la copie d'une sous-partie, etc.

Exemple : ''toto'' est une chaîne de caractères, toto on ne sait pas ce que c'est (pour l'instant)

Habituellement, les opérations font intervenir des données de type identique et renvoie un résultat du même type.

Type	Valeurs
Boolean	Vrai, faux
Integer	Entiers
Long	Entiers
Single	Réels
Double	Réels
Currency	4 chiffres après la ,
Date	1/1/100 à 31/12/9999
String	Chaines de caractères
Object	Tout objet
Variant	N'importe quel type

Les opérateurs de comparaison confrontent des données de même type, mais le résultat est un booléen



Exemples

5 > 2 → True

5 > ''toto'' → illicite

5 <> 5 → False

''toto'' > ''tata'' → True

Licite. Comparaison de gauche à droite basée sur le code ASCII. Arrêt des comparaisons dès que l'indécision est levée.

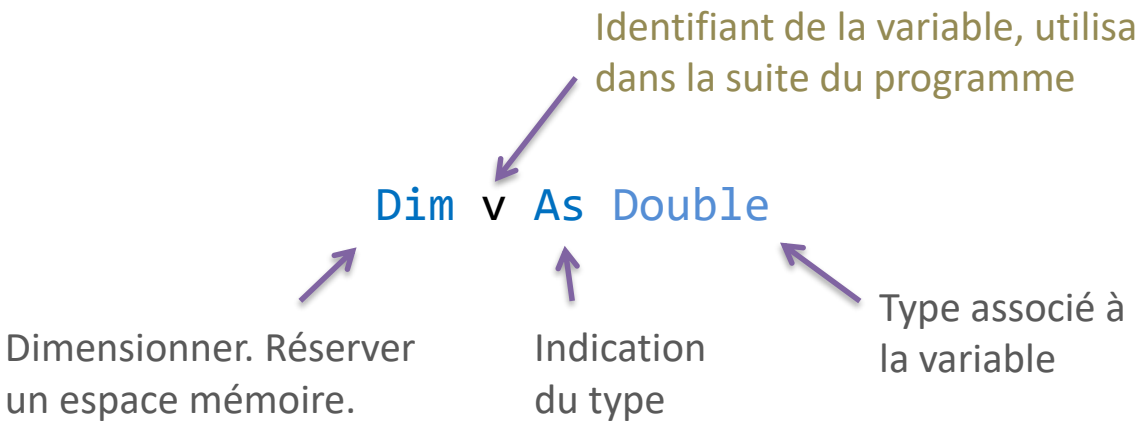
- Valeur absolue: Abs(-9) retourne 9
- Signe: Sgn(-18) retourne –1 (ou 0 ou 1)
- Troncation: Int(13.12) retourne 13
  Int(-14.8) retourne –15
  - Tronque à l'entier inférieur le plus proche
- Partie entière: Fix(-18.3) = -18
  Fix(18.3) = 18
  - Enlève la partie décimale

- Sqr, Exp, Log
  - Sqr(4) retourne 2, Exp(5) retourne 148.413…,
    Log(9) retourne 2.197224… (en base e)
- Nombres aléatoires
  - Rnd retourne un nombre aléatoire entre 0 (compris) et 1 (non compris)
  - a = Rnd a peut valoir 0.12131441
  - Int((b – a + 1) * Rnd + a) retourne un nombre aléatoire entier entre a et b
- Sin, Cos, Tan, Atn (arc-tangente)

- Date retourne la date actuelle
- Time retourne l'heure courante
  - Date et Time peuvent retourner des chaînes de carctères String
- DateSerial retourne une valeur unique pour une date donnée, sous forme Variant
  - dv1 = DateSerial(2003, 4, 22)
    dv2 = DateSerial(1928, 5, 3)
    dv1 – dv2 représente le nombre de jours entre ces deux dates
- Day, Month et Year retourne respectivement le jour, le mois et l'année d'une date.
  - Year(Date) retourne 2019 cette année (en entier)

Les **variables** correspondent à des identifiants auxquels sont associés des valeurs d'un type donné. Elles matérialisent un espace mémoire avec un contenu que l'on peut lire ou écrire.

Déclaration d'une variable



Affectation. Attribuer une valeur à la variable

```
v = 2.5
```

= est le symbole d'affectation. A gauche de = on modifie le contenu dans une variable, à droite on lit le contenu d'une variable. C'est pour cette raison que l'instruction v = v + 1 est licite.

Opération et affectation

```
x = v * 2
```

La valeur 5 est écrite dans la variable x qui doit être déclarée au préalable.

Ecriture et utilisation des fonctions personnalisées dans Excel

FONCTIONS PERSONNALISÉES

Une fonction personnalisée est une fonction VBA qui peut être appelée dans un classeur Excel. Elle prend en entrée des informations en provenance des feuilles du classeur (principalement) et renvoie une valeur insérée dans une cellule (le plus souvent également).

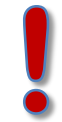
Formalisme `Function NomFonction(paramètres) as type de donnée`

Est un identifiant qui doit respecter la syntaxe VBA

Type de la valeur retournée par la fonction.

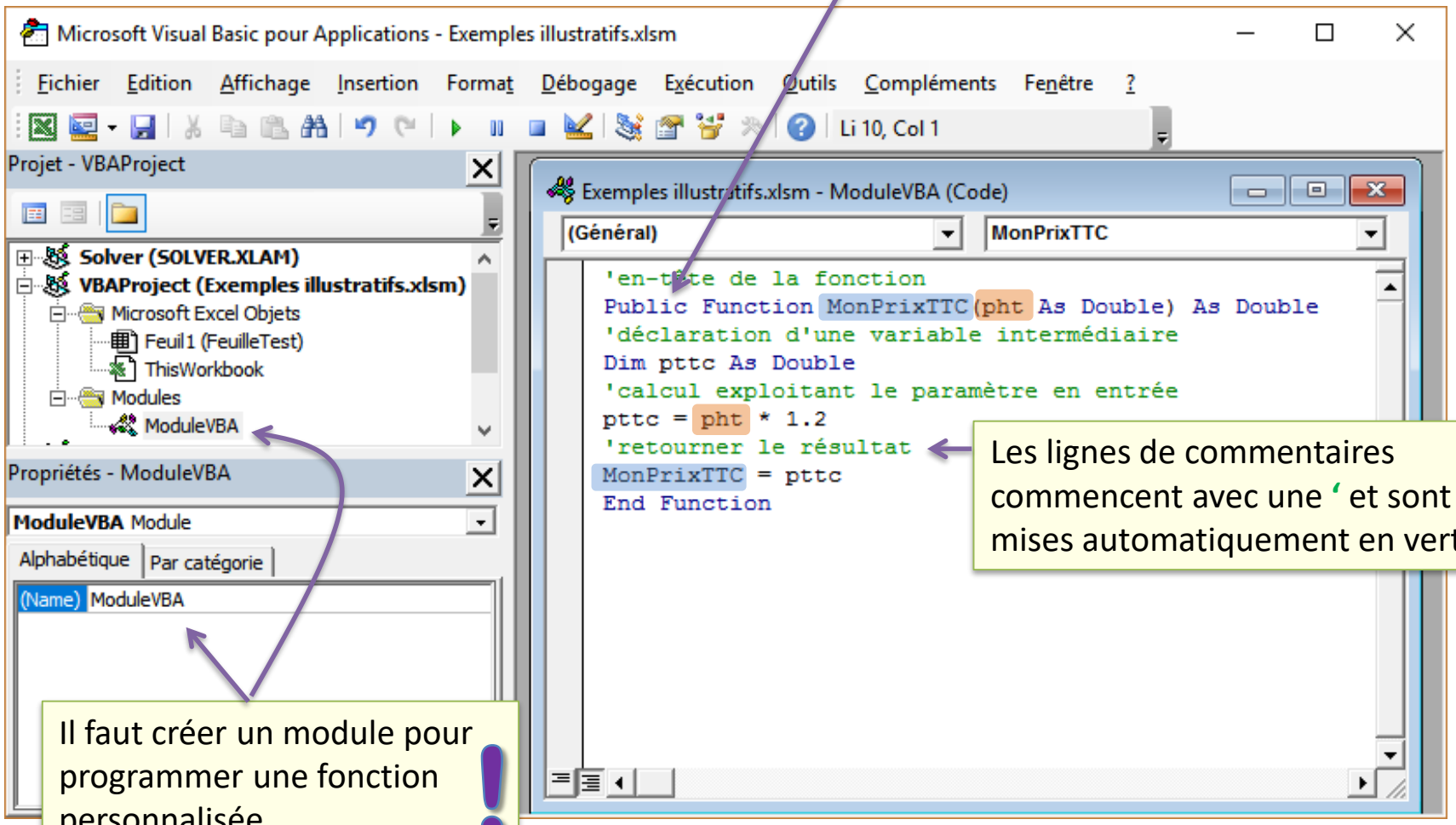
Les informations que prend en entrée la fonction, elles prennent la forme *nom_parametre as type de paramètre*. Il peut y en avoir plusieurs, ils sont séparés par des « , » dans ce cas.

Un classeur Excel contenant du code VBA doit être enregistré au format XLSM, prenant en charge les macros. Sinon on perd son code.



Entrée : prix HT (réel)
Sortie : prix TTC (réel)

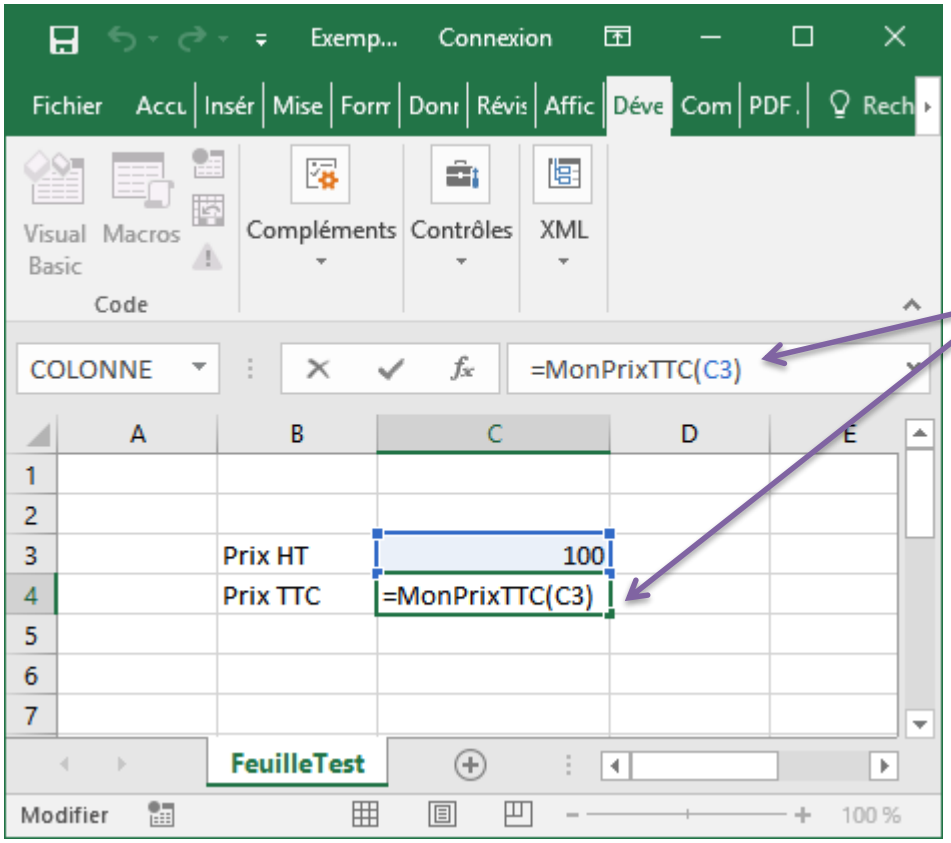
Public pour que la fonction soit visible en dehors du module, notamment dans la feuille de calcul



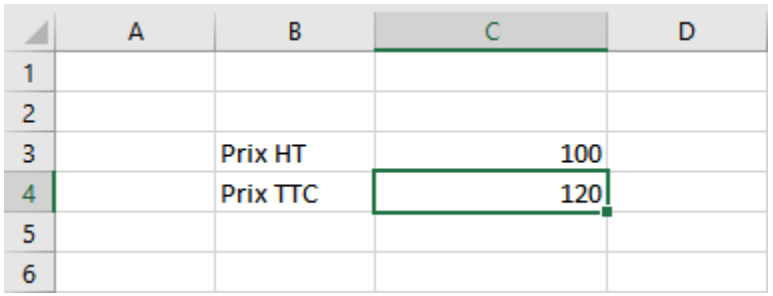
Les lignes de commentaires commencent avec une ' et sont mises automatiquement en vert.

Il faut créer un module pour programmer une fonction personnalisée !

Utilisation de la fonction dans une feuille Excel



La fonction est insérable dans la feuille de calcul comme n'importe quelle autre fonction Excel. Elle est accessible dans la catégorie « Fonctions personnalisées ».

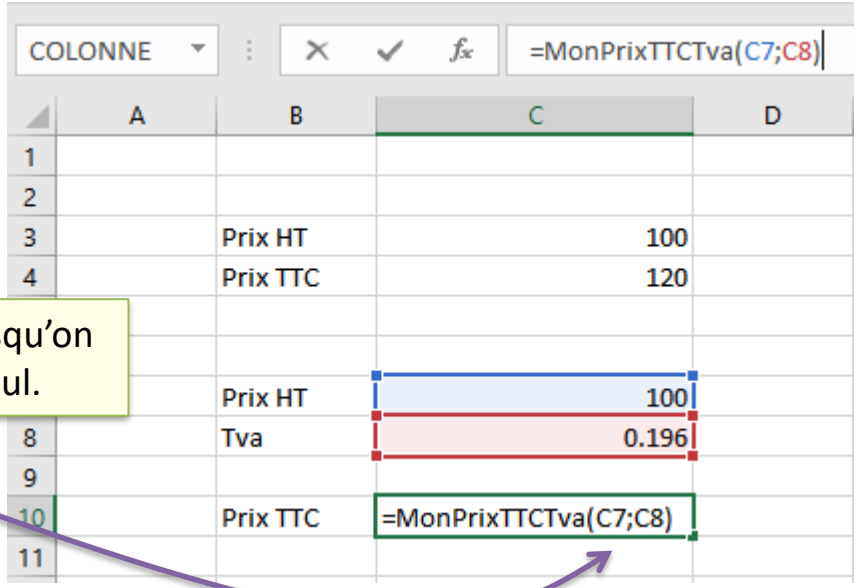


Le résultat s'affiche une fois la fonction insérée et validée. La fonction est automatiquement appelée à chaque fois que la feuille a besoin d'être recalculée (comme pour les autres fonctions standards d'Excel).

Entrées : prix HT (réel), tva (réel)
Sortie : prix TTC (réel)

(1) Le séparateur de paramètres est la « , » lors de la définition de la fonction.

```
'fonction avec 2 paramètres
Public Function MonPrixTTCTva(pht As Double, tva As Double) As Double
'déclaration d'une variable intermédiaire
Dim pttc As Double
'calcul exploitant les paramètres en entrée
pttc = pht * (1 + tva)
'retourner le résultat
MonPrixTTCTva = pttc
End Function
```



(2) Très étrangement, il devient « ; » lorsqu'on appelle la fonction dans la feuille de calcul.

Plus loin avec la programmation…

STRUCTURES ALGORITHMIQUES

Permet d'activer une partie du code en fonction de la réalisation d'une condition ou pas.

Syntaxe

```
If condition Then
  bloc d'instructions
  si la condition est vraie
Else
  bloc d'instructions
  si la condition est fausse
End If
```

(1) Condition est souvent une opération de comparaison

(2) La valeur de retour de Condition est de type booléen (True ou False)

(3) Then doit être sur la même ligne que If

(4) La partie Else est facultative (ne rien faire si la condition est fausse)

(5) Il est possible d'imbriquer une autre structure conditionnelle If dans les blocs d'instructions

Entrées : prix HT (réel), catégorie de produit (chaîne)
Sortie : prix TTC (réel)

```
Public Function MonTTCBis(pht As Double, cat As String) As Double
'déclarer la variable de calcul
Dim pttc As Double
'en fonction de la catégorie de produit
If (cat = "luxe") Then
    pttc = pht * 1.33
Else
    ‘la valeur de cat est différente de ‘’luxe’’
    pttc = pht * 1.2
End If
'renvoyer le résultat
MonTTCBis = pttc
End Function
```

Permet d'activer une partie du code en fonction des valeurs prises par une variable de contrôle. Peut se substituer au IF, mais pas toujours, tout dépend de la forme de la condition (*condition composée, on doit passer par un IF*).

Syntaxe

```
Select Case variable
       Case valeur 1
              bloc d'instructions
       Case valeur 2
              bloc d'instructions
       ...
       Case Else
              bloc d'instructions
End Select
```

(1) Variable est la variable de contrôle, elle peut être de n'importe quel type en VBA, y compris un réel ou une chaîne de caractères

(2) Valeur doit être de type compatible avec variable

(3) La partie Case Else est facultative

(4) L'imbrication avec un autre IF ou un autre Select Case (autre variable de contrôle) est possible.

Entrées : prix HT (réel), catégorie de produit (chaîne)
Sortie : prix TTC (réel)

```
'fonction select case
Public Function MonTTCSelon(pht As Double, cat As String) As Double
'déclarer la variable de calcul
Dim pttc As Double
'en fonction de la catégorie de produit
Select Case cat
    Case "luxe"
        pttc = pht * 1.33
    Case Else
        pttc = pht * 1.2 'toute autre valeur que ‘’luxe’’
End Select
'renvoyer le résultat
MonTTCSelon = pttc
End Function
```

Il est possible d'introduire des plages de valeurs dans la partie Case de la structure Select Case. La comparaison devient plus sophistiquée. Variable est un numérique dans ce cas, entier ou même réel.

Syntaxe

```
Select Case variable
       Case Is op.de.comparaison valeur
              bloc d'instructions

       Case valeur de départ To valeur de fin
              bloc d'instructions

       Case Else
              bloc d'instructions
End Select
```

Entrée : quantité (entier)
Sortie : prix unitaire (réel)
Calcul : quantité < 100 → p.u. = 0.5
100 ≤ quantité ≤ 200 → p.u. = 0.3
quantité > 200 → p.u. = 0.2

```
'calcul du prix unitaire en fonction de la quantité
Public Function MonPU(quantite As Long) As Double
'variable intermédiaire
Dim pu As Double
'selon les valeurs de quantité
Select Case quantite
    Case Is < 100
        pu = 0.5
    Case 100 To 200
        pu = 0.3
    Case Is > 200 'Case Else aurait fait l'affaire aussi
        pu = 0.2
End Select
MonPU = pu
End Function
```

Faire répéter l'exécution d'un bloc d'instructions. Le nombre d'itérations est contrôlé par un indice.

Syntaxe

```
For indice = val.départ to val.fin step pas
        bloc d'instructions
        ...
Next indice
```

(1) Indice est un type ordonné, très souvent un numérique

(2) pas contrôle le passage d'une valeur à l'autre d'indice, si omis, pas = 1 par défaut

(3) Next entérine le passage à la valeur suivante de indice, si cette prochaine valeur est > à val.fin, on sort de la boucle

(4) Val.fin doit être superieure à val.départ pour que l'on rentre dans la boucle

(5) Si pas est négatif, val.fin doit être inférieure à val.départ cette fois-ci

(6) L'instruction Exit For permet de sortir prématurément de la boucle

(7) On peut imbriquer des boucles (une boucle à l'intérieur d'une autre boucle)

Entrée : n (entier)
Sortie : S (réel)
Calcul : $S = 1^2 + 2^2 + \ldots + n^2$

```
'calcul de la somme des carrés des valeurs
Public Function MaSommeCarre(n As Long) As Double
'variables de calcul (s pour la somme, i : indice)
Dim s As Double, i As Long
'initialisation
s = 0
'boucle avec l'indice i
For i = 1 To n Step 1
    s = s + i ^ 2
‘Next joue le rôle de l’incrémentation (i suivant)
Next i
'renvoyer le résultat
MaSommeCarre = s
End Function
```

Faire répéter l'exécution d'un bloc d'instructions. Le nombre d'itérations est contrôlé par une condition. Attention à la boucle infinie c.-à-d. la condition permettant de sortir de la boucle n'est jamais déclenchée.

Syntaxe

```
Do While condition
        Bloc d'instructions...
        ...
Loop
```

(1) Condition est un booléen, c'est souvent une opération de comparaison

(2) On continue l'exécution TANT QUE la condition est vraie ; si la condition est fausse, on sort de la boucle

(3) Exit Do permet de provoquer la sortie prématurée de la boucle

Si la condition est fausse d'emblée. On peut ne pas rentrer dans la boucle.

Boucle DO WHILE…LOOP (un exemple)

Entrée : n (entier)
Sortie : S (réel)
Calcul : $S = 1^2 + 2^2 + \ldots + n^2$

```
'calcul de la somme des carrés des valeurs
Public Function MaSommeCarreWhile(n As Long) As Double
'variables de calcul
Dim s As Double, i As Long
'initialisation
s = 0
'il nous revient aussi d'initialiser l'indice
i = 1
'boucle TANT QUE
Do While (i <= n)
    'sommer
    s = s + i ^ 2
    'pas de next, nous devons incrémenter l'indice
    i = i + 1
Loop
'renvoyer le résultat
MaSommeCarreWhile = s
End Function
```

Faire répéter l'exécution d'un bloc d'instructions. Le nombre d'itérations est contrôlé par une condition.

Syntaxe

```
Do
        Bloc d'instructions
        ...
        ...
Loop While condition
```

- On est sûr de rentrer au moins une fois dans la boucle. !
- Le choix de la bonne structure (Faire.. Tant Que ou Tant Que.. Faire) dépend du problème à traiter !

Les boucles DO contrôlées par une condition sont très riches en VBA.

```
Do { While | Until } condition
        [ statements ]
        [ Exit Do ]
        [ statements ]
Loop
-or-
Do
        [ statements ]
        [ Exit Do ]
        [ statements ]
Loop { While | Until } condition
```

Le Répeter… Jusqu'à (Until) existe aussi.

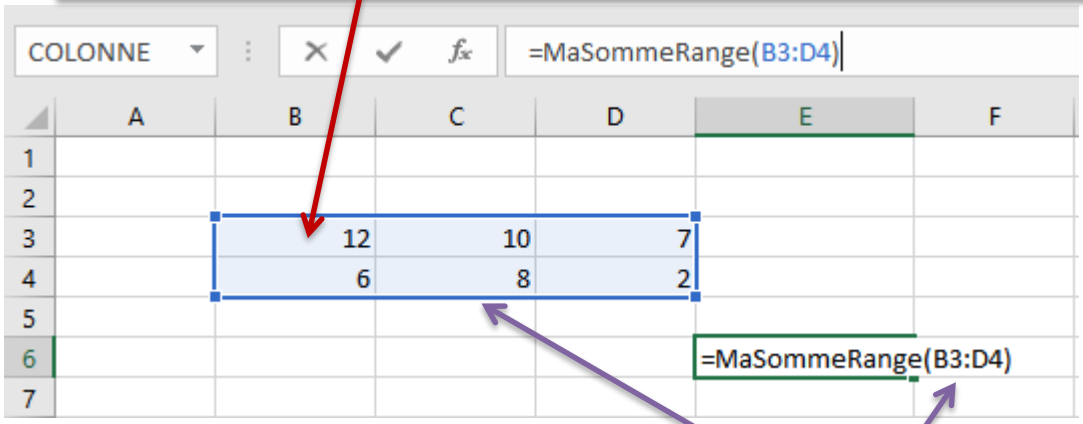
Le type « plage de cellules » spécifique à Excel

LE TYPE RANGE

Le type RANGE désigne une plage de cellules, c'est un type spécifique à Excel.

Exemple

Coin en haut et à gauche de la plage de cellules passée en paramètre de la fonction = coordonnée (1, 1) c.-à-d. ligne n°1 et colonne n°1, quelle que soit la position absolue de la plage dans la feuille de calcul (ici le coin nord-ouest est en B3)



Un bloc de cellules (B3:D4) est passé en paramètre de la fonction. Ce bloc est forcément rectangulaire, avec, ici : 2 lignes et 3 colonnes.

La fonction MaSommeRange() est censée faire la même chose que la fonction standard SOMME() d'Excel.

Entrée : plage (range)
Sortie : S (réel)
Calcul : Somme des valeurs

Lignes et colonnes commencent à l'indice 1, quelle que soit la position de la plage dans la feuille.

```
'Travail sur le type Range
Public Function MaSommeRange(plage As Range) As Double
'variables intermédiaires
Dim s As Double, i As Long, j As Long
'initialisation de la somme
s = 0
'parcours de la plage de cellules
For i = 1 To plage.Rows.Count Step 1 'lignes
    For j = 1 To plage.Columns.Count Step 1 'colonnes
        'lecture des valeurs et somme
        s = s + plage.Cells(i, j).Value
    Next j
Next i
'renvoyer le résultat
MaSommeRange = s
End Function
```

Nombre de lignes de la plage de cellules.

Nombre de colonnes.

Accès à la valeur (Value) de la cellule : ligne n°i, colonne n°j

La boucle For Each est adaptée au parcours des collections.

Or une plage de cellules est une collection de cellules.

```
'Travail sur le type Range avec un For Each
Public Function MaSommeRangeEach(plage As Range) As Double
'variables intermédiaires
Dim s As Double, cellule As Range
'initialisation de la somme
s = 0
'parcours de la plage de cellules
For Each cellule In plage
    s = s + cellule.Value
Next cellule
'renvoyer le résultat
MaSommeRangeEach = s
End Function
```

Une cellule est une plage de cellules avec une seule cellule.

plage fait figure de collection à traiter. Qu'importe le sens du parcours ici (ligne par ligne, ou colonne par colonne).

C'est bien la valeur contenue dans la cellule qui est exploitée pour la somme.

Type spécial qui peut contenir toutes sortes de valeur

LE TYPE VARIANT

Le type de variant peut gérer tout type de valeurs. Il est très souple, particulièrement commode quand on ne connaît pas à l'avance le type à utiliser. Mais attention, il ne faut pas en abuser, il est très lent parce que multiplie les vérifications à chaque accès à la variable correspondante.

Entrée : a, b (réel)
Sortie : a/b si b ≠ 0, « division par zéro » sinon

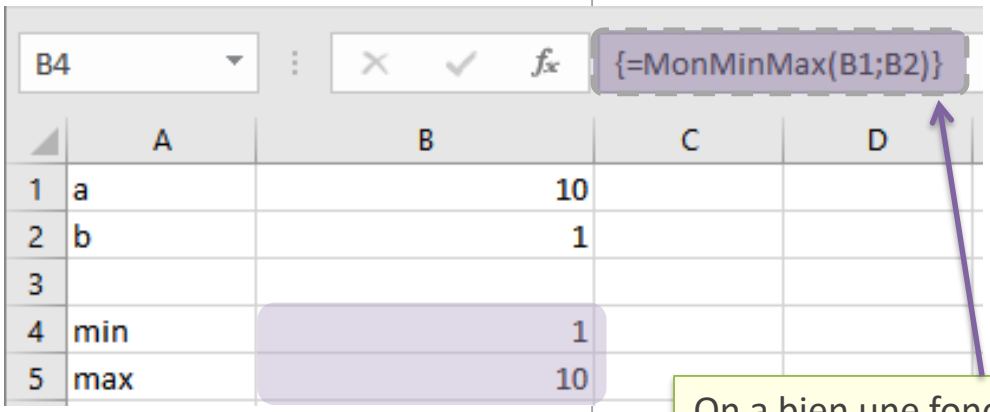
Un coup, la fonction renvoie un réel, un autre coup elle doit renvoyer une chaîne de caractères.

```
'utilisation du type variant
Public Function MaDivision(a As Double, b As Double) As Variant
'var. intermédiaire
Dim resultat As Variant
'calcul
If (b <> 0) Then
    resultat = a / b
Else
    resultat = "division par zéro"
End If
'renvoyer le résultat
MaDivision = resultat
End Function
```

Dans la même variable resultat, de type variant, on peut affecter un réel et une chaîne de caractères.

On peut s'en servir pour renvoyer un tableau. Une fonction peut donc renvoyer plusieurs valeurs d'un coup, à l'instar des fonctions matricielles d'Excel (il faut valider la saisie de la fonction avec la séquence de touches CTRL + MAJ + ENTREE).

```
'renvoyer plusieurs valeurs
Public Function MonMinMax(a As Double, b As Double) As Variant
'un tableau interne - matrice 2 lignes et 1 colonne
Dim tableau(1 To 2, 1 To 1) As Double
'identifier le min et le max
If (a < b) Then
    tableau(1, 1) = a
    tableau(2, 1) = b
Else
    tableau(1, 1) = b
    tableau(2, 1) = a
End If
'renvoyer le tableau
MonMinMax = tableau
End Function
```



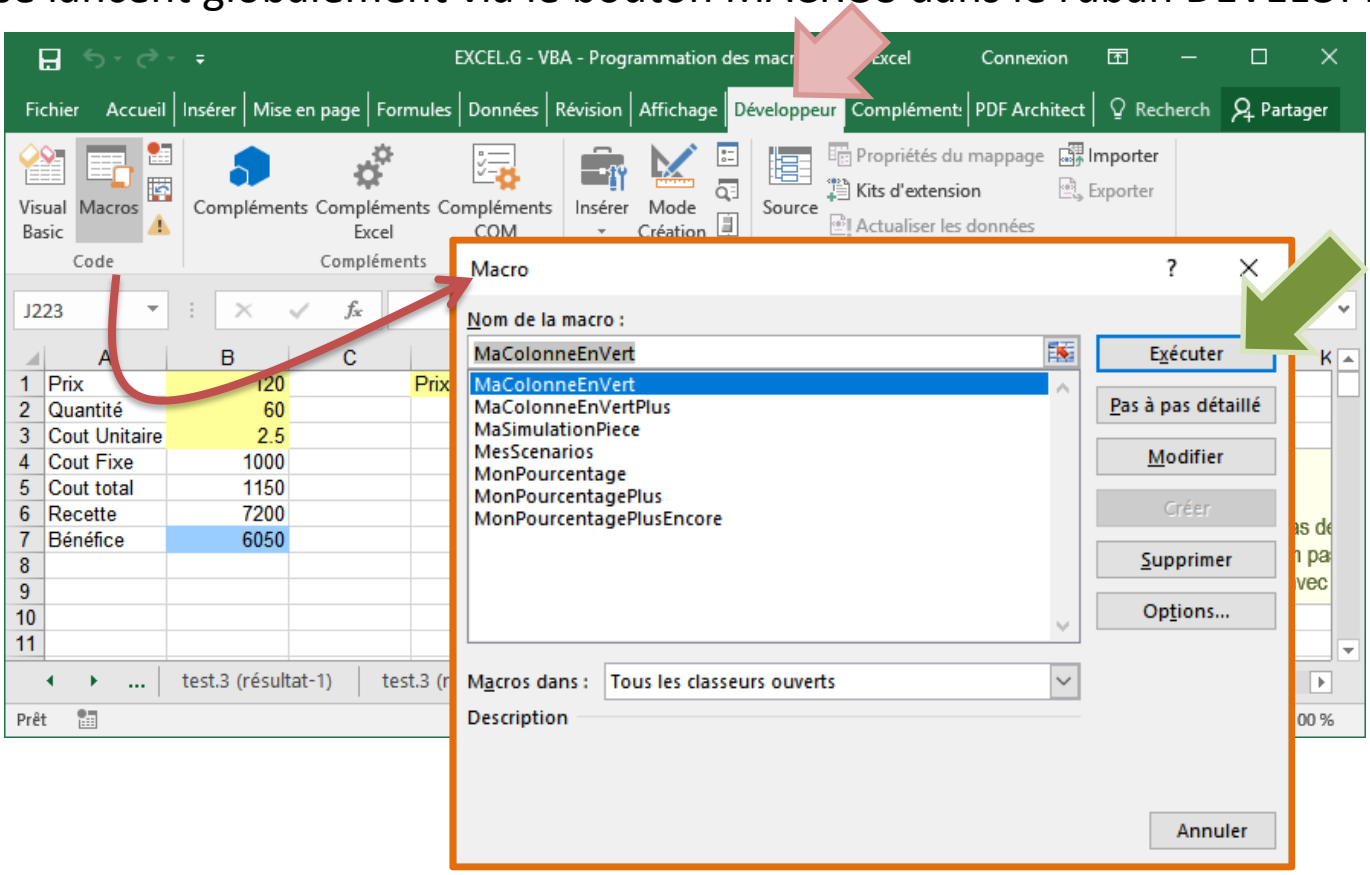
On a bien une fonction matricielle comme peuvent en témoigner les accolades { } qui encadrent l'appel de la fonction.

Programmation des macros – Travailler directement sur les feuilles

LES MACROS (1)

Les macros sont également des procédures que l'on crée à l'intérieur d'un module. Mais, à la différence des Function, ce sont des Sub() sans paramètres qui peuvent manipuler (accéder et modifier) directement les objets Excel (classeurs, feuilles, cellules, graphiques, scénarios, tableaux croisés dynamiques…).

Ils ne s'exécutent pas de la même manière. Au lieu de les insérer dans une cellule, ils se lancent globalement via le bouton MACROS dans le ruban DEVELOPPEUR.

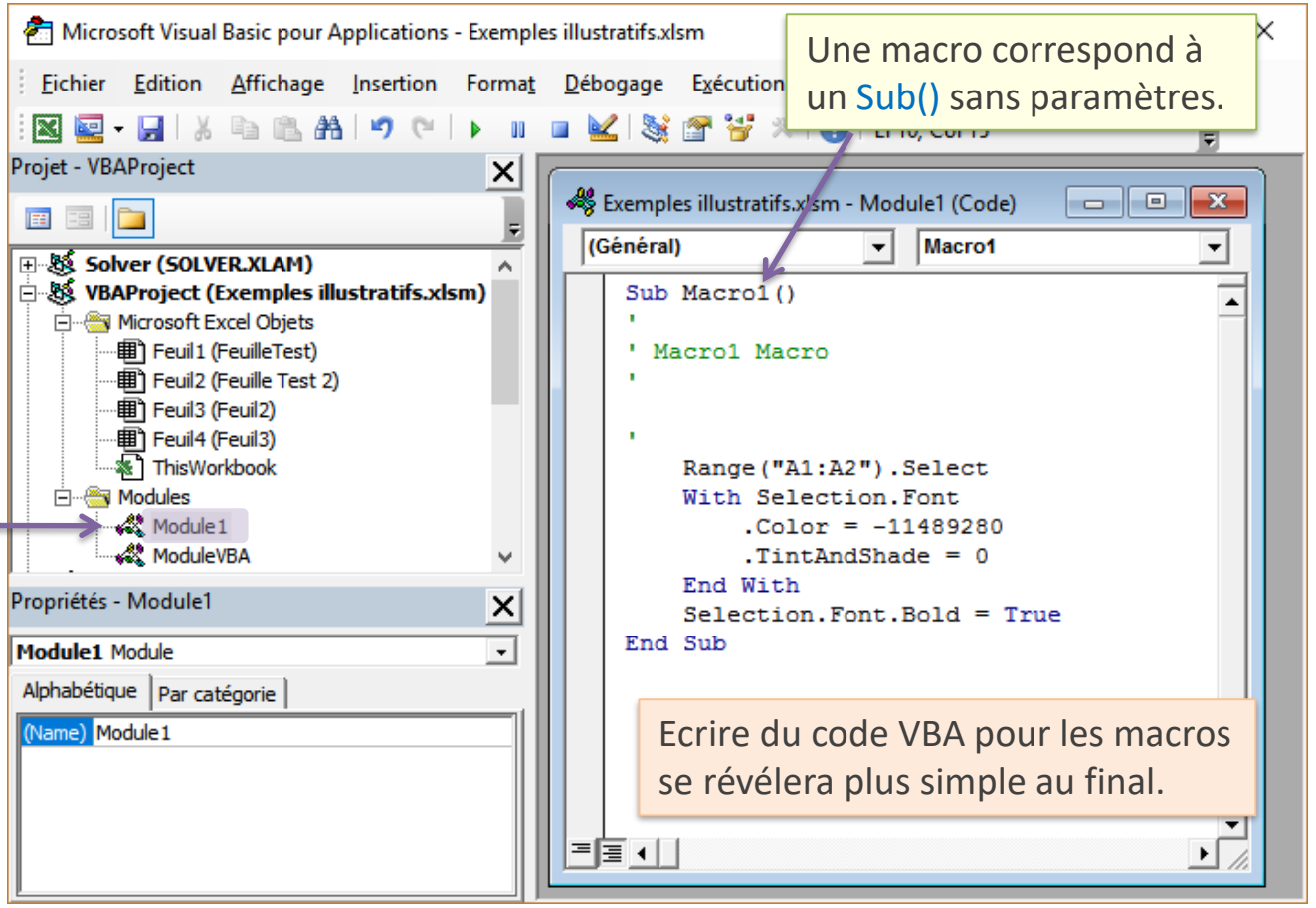


Une manière simple de générer une macro est de lancer l'enregistreur de macros.

Du code VBA est automatiquement généré.

Exemple : mettre en gras et vert le contenu des cellules A1 et A2

	A
1	10
2	12



Avantages :

- Il n'y a pas plus simple pour produire du code, on peut créer et exécuter une macro sans aucune notion de programmation
- Il nous donne des indications précieuses sur les commandes associées aux objets Excel

Inconvénients :

- On travaille à structure fixée, si la configuration de la feuille change, il n'est pas possible de lancer la macro
- On ne bénéficie pas de la puissance des structures algorithmiques

En définitive :

- Il peut nous aider à rédiger notre code en nous donnant des pistes sur la syntaxe des commandes et les objets adéquats à manipuler (ex. imprimer automatiquement des feuilles, on lance l'enregistreur une fois, on intègre son code dans le notre à l'intérieur d'une boucle).

Ecrire directement des macros est simple une fois assimilé la philosophie de l'approche, et identifié les principaux objets et l'accès à leurs propriétés et méthodes (l'enregistreur peut nous y aider).

Classeurs

```
Workbooks(''classeur1.xlsm'').Activate
```

Activer (sélectionner) le classeur dont le nom de fichier est ''classeur1.xlsm''

Feuilles

```
Sheets(''Feuil1'').Activate
```

Dans le classeur courant, activer la feuille de calcul dont le nom est ''Feuil1'' (visible dans la languette au bas de la feuille)

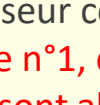
```
Workbooks(''classeur1.xlsm'').Sheets(''Feuil1'').Activate
```

On peut combiner les écritures.

Cellules

```
Cells(1,1).Value = 15
```

Dans la feuille courante du classeur courant, insérer la valeur 15 dans la cellule ligne n°1, colonne n°1 c.-à-d. en A1, les coordonnées sont absolues ici.



```
Sheets(''Feuil1'').Cells(1,1).Value = 15
```

De nouveau, on peut combiner.

Exemple de macros – Simulation valeurs de TVA

Ecrire une macro qui insère différentes valeurs de TVA en **B2** et récupère les valeurs de prix TTC en **B3**.

Les différentes valeurs de TVA testées doivent être retranscrites au fur et à mesure dans la colonne D.

Les valeurs de Prix TTC correspondantes doivent être recensées en colonne E

	A	B	C	D	E
1	PHT	100		TVA (%)	Prix TTC
2	TVA (%)	30			
3	PTTC	=B1*(1+B2/100)			
4					
5					
6					

```
Sub SimulationTVA()
'variables
Dim pht As Double, pttc As Double
Dim tva As Double
Dim i As Long
'début d'écriture des valeurs en ligne 2
i = 2
'récupérer la valeur du PHT
pht = Cells(1, 2).Value 'en B1
'faire varier la tva de 10% à 30% avec un pas de 5%
For tva = 10 To 30 Step 5
    'insérer la valeur de la TVA en B2
    Cells(2, 2).Value = tva
    'Récupérer le prix ttc en B3
    pttc = Cells(3, 2).Value
    'inscription des valeurs
    'TVA en colonne D
    Cells(i, 4).Value = tva
    'PTTC en colonne E
    Cells(i, 5).Value = pttc
    'passage à la ligne suivante
    i = i + 1
Next tva
End Sub
```

Remarque : il faut être sur la feuille adéquate avant de lancer la macro, sinon le programme ne saura pas où chercher **Cells(…).**

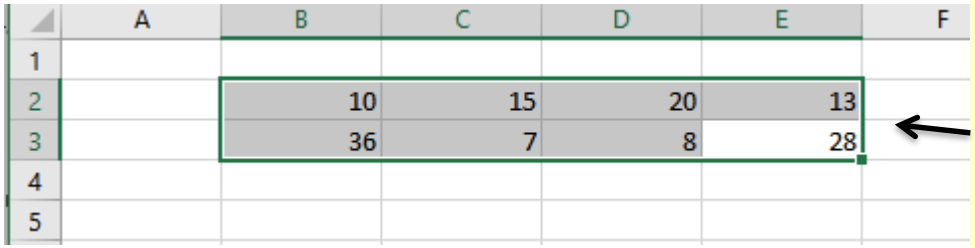
A l'issue de la simulation…

	A	B	C	D	E
1	PHT	100		TVA (%)	Prix TTC
2	TVA (%)	30		10	110
3	PTTC	130		15	115
4				20	120
5				25	125
6				30	130

Travailler sur les sélections de l'utilisateur

LES MACROS (2)

Comment programmer une macro qui manipule directement une plage de cellules sélectionnée par l'utilisateur ? Attention, nous ne sommes pas dans la même configuration que les fonctions personnalisées ici, nous n'insérons pas un résultat dans une cellule, nous manipulons et modifions directement la plage sélectionnée.



	A	B	C	D	E	F
1						
2		10	15	20	13	
3		36	7	8	28	
4						
5						

Exemple : dans cette sélection (les cellules doivent être sélectionnées avant de lancer la macro !), mettre en police verte les cellules contenant une valeur paire.

```
Sub MesValeursPaires()
'variable intermédiaire
Dim cellule As Range
'boucler sur la sélection
For Each cellule In Selection
    'tester le contenu
    If (cellule.Value Mod 2 = 0) Then
        'modifier la couleur de la police
        cellule.Font.ColorIndex = 4
    End If
Next cellule
End Sub
```

Selection est un objet Excel (Selection est donc un mot clé). Il est de type Range (que nous connaissons bien).

Résultat…

10	15	20	13
36	7	8	28

Dans une sélection, de nouveau les coordonnées sont relatives c.-à-d. le coin en haut à gauche d'une sélection correspond à la cellule (ligne n°1, colonne n°1) quelle que soit la position de la sélection dans la feuille.

```
Sub MesValeursPairesBis()
'variables intermédiaires
Dim i As Long, j As Long
'boucler sur les lignes
For i = 1 To Selection.Rows.Count
    'boucler sur les colonnes
    For j = 1 To Selection.Columns.Count
        'tester le contenu
        If (Selection.Cells(i, j).Value Mod 2 = 0) Then
            'modifier la couleur de la police
            Selection.Cells(i, j).Font.ColorIndex = 4
        End If
    Next j
Next i
End Sub
```

Aucun doute, Selection est bien de type Range

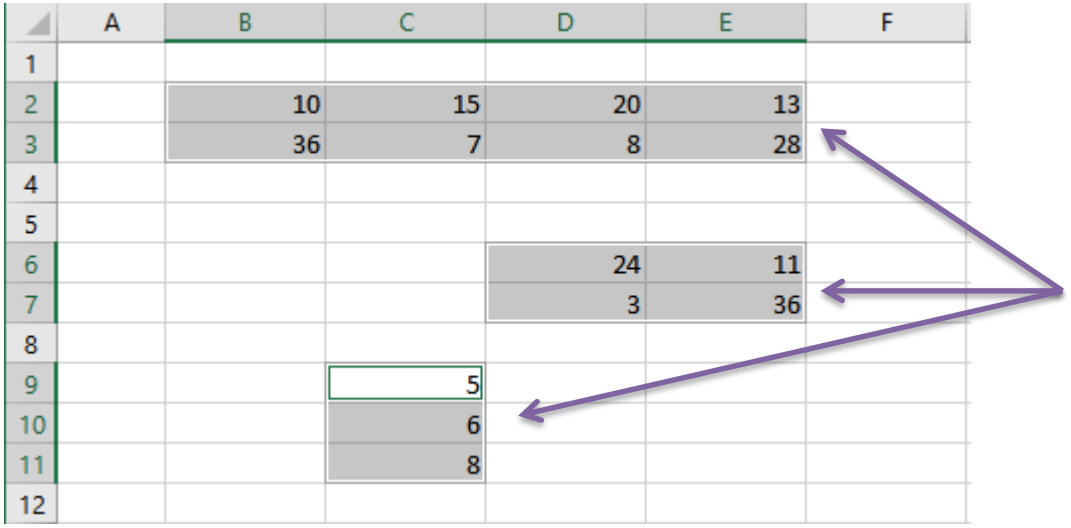
Identifier la première cellule contenant la valeur minimale dans une plage, mettre sa police en bleu.

10	15	20	13
36	7	8	28

```
Sub MonMinBleu()
‘variables intermédiaires
‘min va servir de cellule témoin
Dim cellule As Range, min As Range
'initialisation du témoin sur la 1ère cellule
Set min = Selection.Cells(1, 1)
'parcourir
For Each cellule In Selection
    'comparer avec le contenu de la cellule témoin
    If (cellule.Value < min.Value) Then
        'màj de la cellule témoin
        Set min = cellule
    End If
Next cellule
'mettre la couleur pour la cellule minimale
min.Font.ColorIndex = 5
End Sub
```

Range est un **objet**. Une affectation pour une variable objet doit être réalisée à l’aide de l’instruction Set

Une sélection peut être multiple aussi c.-à-d. contenant plusieurs ‘’zones’’



	A	B	C	D	E	F
1						
2		10	15	20	13	
3		36	7	8	28	
4						
5						
6				24	11	
7				3	36	
8						
9			5			
10			6			
11			8			
12						

Un exemple de sélection multiple avec 3 zones.

Très curieusement, le même mot clé Selection peut être exploité.

```
Selection.Areas.Count
```

Nombre de ‘’zones’’ dans la sélection.

```
Selection.Areas(k)
```

Accès à la zone n°k (qui est de type Range).
Areas est une collection de zones.

```
Sub MonMinZoneBleu()
'var. intermédiaires
Dim zone As Range, min As Range
'pour chaque zone
For Each zone In Selection.Areas
    'à l'intérieur de chaque zone
    'initialisation
    Set min = zone.Cells(1, 1)
    'parcours des cellules
    For Each cellule In zone
        'comparer
        If (cellule.Value < min.Value) Then
            'màj de la variable témoin
            Set min = cellule
        End If
    Next cellule
    'mettre la couleur pour la cellule minimale
    min.Font.ColorIndex = 5
'passage à la zone suivante
Next zone
End Sub
```

Pour chaque zone, mettre en police bleue la cellule contenant la valeur minimale.

Selection.Areas est une collection. On peut utiliser un For Each. On aurait pu aussi passer par un accès indicé. Par ex.

```
For k = 1 to Selection.Areas.Count
   Set zone = Selection.Areas(k)
   Etc…
```

Résultat…

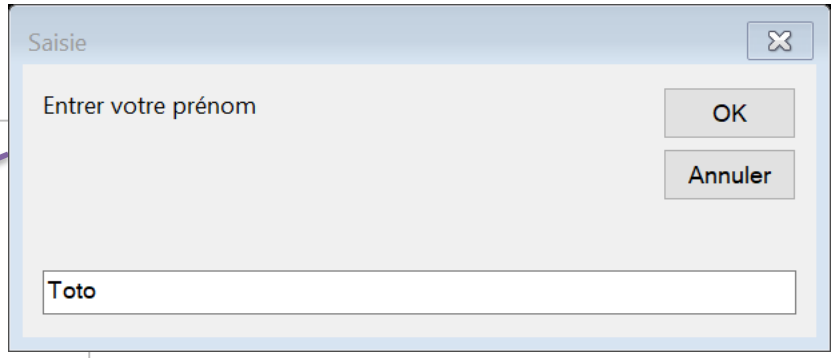
	A	B	C	D	E	F
1						
2		10	15	20	13	
3		36	7	8	28	
4						
5						
6				24	11	
7				3	36	
8						
9			5			
10			6			
11			8			
12						

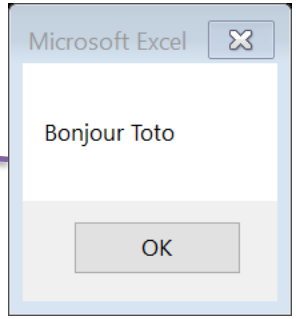
BOÎTES DE DIALOGUE

Les boîtes de dialogue permettent d'interagir avec l'utilisateur. Nous nous en tenons aux plus simples ici. InputBox() pour la saisie, MsgBox() pour l'affichage.

InputBox() est une fonction

```
Sub MesBoitesDeDialogue()
'var. intermédiaire
Dim prenom As String
'saisie
prenom = InputBox("Entrer votre prénom", "Saisie", "")
'affichage
MsgBox ("Bonjour " & prenom)
End Sub
```



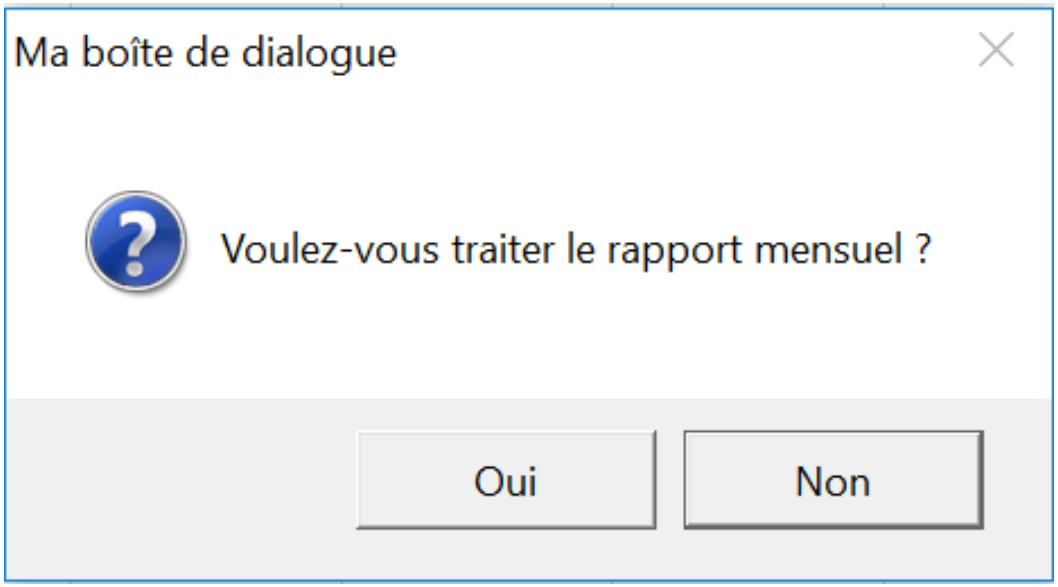


Noter la concaténation de chaînes de caractères pour faire apparaître le prénom dans la boîte de dialogue.

- Boite de dialogue de base, "personnalisable"

vbOKOnly	0	N'affiche que le bouton ok
vbOKCancel	1	Ok et Annuler
vbAbortRetryIgnore	2	Abandonner, Recommencer, Ignorer
vbYesNoCancel	3	Oui, Non, Annuler
vbYEsNo	4	Oui, Non
vbRetryCancel	5	Recommencer, Annuler
vbCritical	16	Icône message critique
vbQuestion	32	Icône Question
vbExclamation	48	Icône exclamation
vbInformation	64	Icône Information
vbDefaultButton1	0	Le premier bouton est par défaut
vbDefaultButton2	256	Le 2ième bouton est par défaut
vbDefaultButton3	512	Le 3ième bouton est par défaut
vbDefaultButton4	768	Le 4ième bouton est par défaut
vbSystemModal	4096	Suspend tout jusqu'à une réponse de l'utilisateur

```
Sub MaMsgBox()
Dim Config As Integer
Dim Reponse As Integer
Config = vbYesNo + vbQuestion + vbDefaultButton2
Reponse = MsgBox("Voulez-vous traiter le rapport mensuel ?", Config, "Ma boîte de dialogue")
If Reponse = vbYes Then
    MsgBox ("Youpi")
Else
    MsgBox ("Ohhhh")
End If
End Sub
```



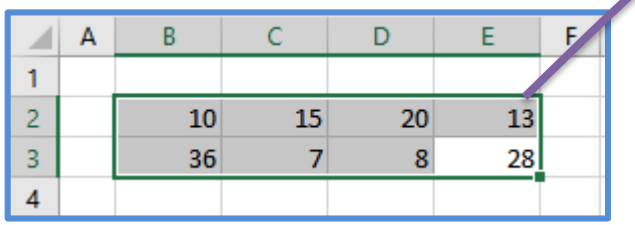
EXPLOITER LES FONCTIONS NATIVES D'EXCEL

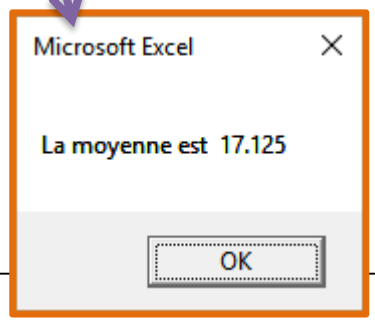
Excel dispose de fonctions natives puissantes. Nous pouvons y accéder dans nos programmes VBA.

Exemple : Vérifier qu'une sélection est simple (une seule zone), puis calculer et afficher la moyenne des valeurs dans une boîte de dialogue.

```
Sub MaMoyenneSelection()
'var. intermédiaire
Dim moyenne As Double
'vérifier la sélection
If (Selection.Areas.Count > 1) Then
    MsgBox ("Attention, ce n'est pas une sélection simple")
Else
    'faire calculer la moyenne de la sélection par Excel
    moyenne = Application.WorksheetFunction.Average(Selection)
    MsgBox ("La moyenne est " & Str(moyenne))
End If
End Sub
```

Noter la syntaxe.





De la documentation à profusion (**inutile d'acheter des livres sur VBA**)

Site de cours de Microsoft

VBA sous Excel : https://msdn.microsoft.com/fr-fr/library/office/ee814737(v=office.14).aspx

Structures de décision : https://msdn.microsoft.com/fr-fr/library/hh892482(v=vs.90).aspx

Structures de boucle : https://msdn.microsoft.com/fr-fr/library/ezk76t25(v=vs.90).aspx

Autres cours et supports

Le Compagnon Info : http://www.lecompagnon.info/excel/

Excel Easy : http://www.excel-easy.com/

Cours VBA Gratuit : http://www.excel-pratique.com/fr/vba.php

Excel VBA for Complete Beginners : http://www.homeandlearn.org/index.html

Et d'autres très nombreux encore… faites chauffer les moteurs de recherche.

Merci

Hervé Hocquard (hocquard@labri.fr)

http://www.labri.fr/perso/hocquard/Teaching.html

Ricco Rakotomalala

http://eric.univ-lyon2.fr/~ricco/