Programmation VBA

Attention !

- **Tout ce qui est dans ce cours est *simplifié* !**
  - Et par conséquent *inexact* !
- **On ne cherche pas l'efficacité !**
  - Certains exemples ici sont inefficaces
  - Ils servent surtout à illustrer certains concepts

Plan du cours

- **Visual Basic for Applications**
- **Dans la macro il y a…**
- **Diviser pour régner**
- **Le problème derrière le clavier**

Visual Basic for Applications

- **Visual Basic for Applications**
  - « VBA »
  - Caractéristiques
    - Interprété, impératif
    - Orienté objet
    - Événementiel
    - Typage hybride
  - Autres
    - Interfacé directement avec les applications bureautiques de la suite Office

Visual Basic for Applications

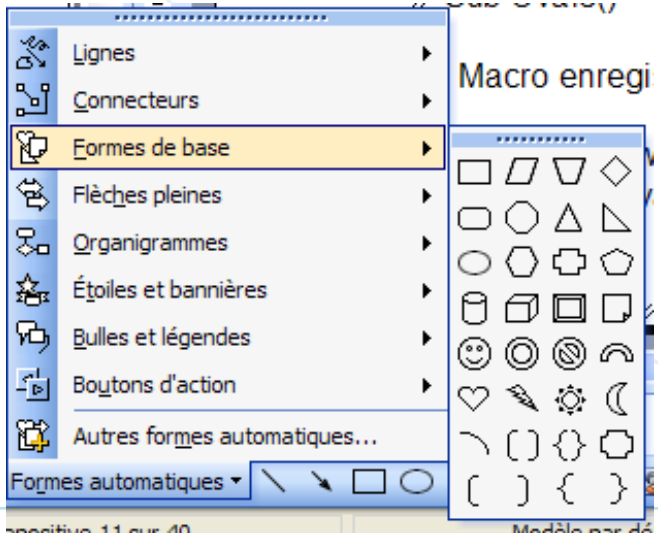
- **Utilité**
  - Automatiser les tâches,
  - Créer des applications complètes,
  - Sécuriser saisies et documents,
  - Créer de nouveaux menus et de nouvelles fonctions pour compléter le logiciel.
- **Comment ?**
  - Macro-commandes pré-programmées,
  - Écriture de macro-commandes personnalisées,
  - Écriture de fonctions personnalisées

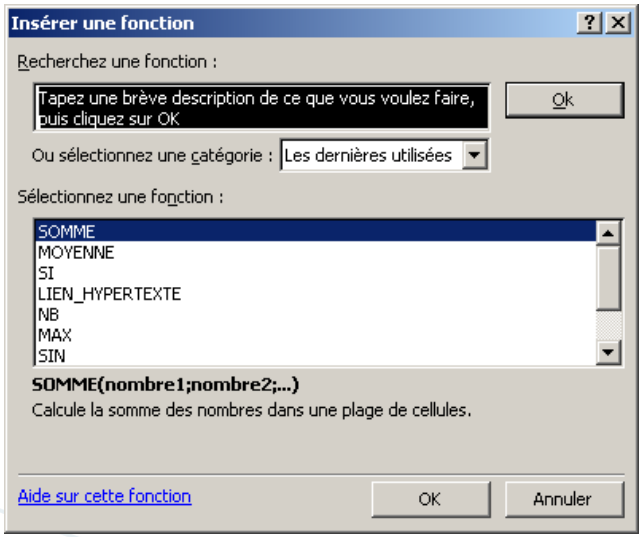
Visual Basic for Applications

- **Dans quels outils peut-on utiliser VBA**
  - Excel
  - Word
  - PowerPoint
  - Access
  - Et certainement d'autres …
- **Chaque outil dispose de ses propres API**
  - API : Applicative Programmation Interfaces
  - Ensemble de fonctions prédéfinies
- **Ce cours se limitera à présenter un sous ensemble réduit des fonctionnalités au sein d'Excel**

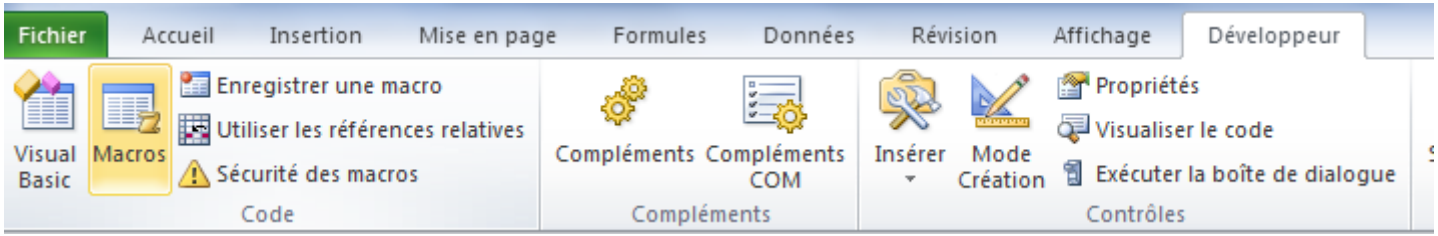
Macro-commande ?

- **Automatisation d'actions répétitives**
- **Macros préprogrammées**
  - *Exemples :*
    - Powerpoint : ovale, cercle, rectangle, carré, segment...
    - Excel : fonctions.



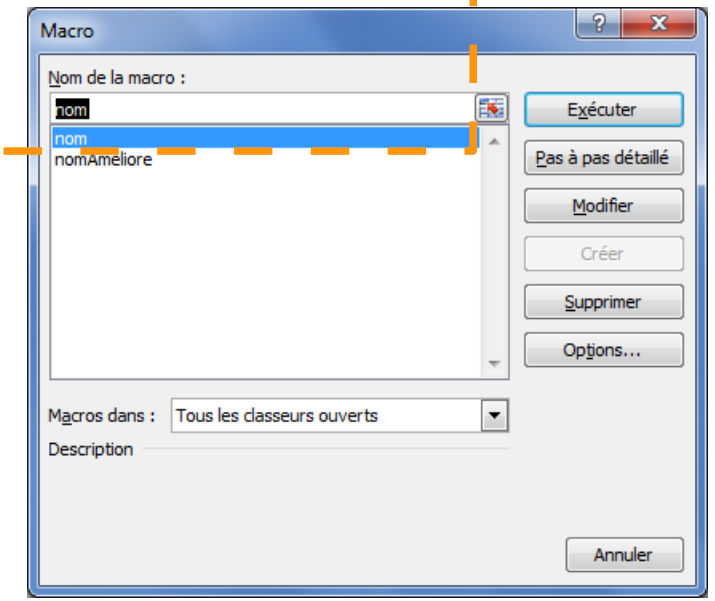


Édition du code d'une macro



- **Accès**
  - Outils de développement
    - Options → personnaliser le Ruban → Ajout de l'onglet « Développeur »
    - Bouton « macros »
  - Raccourci Alt+F8.
- **Action**
  - Sélectionner la macro voulue,
  - Cliquer le bouton Modifier.

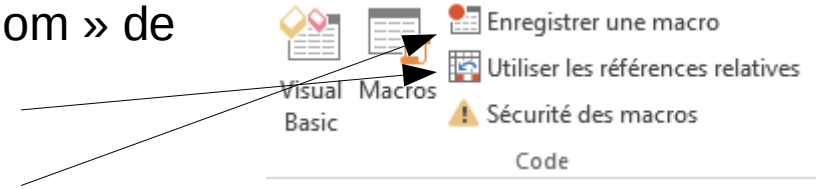
A faire sur votre PC



Création d'une macro

Prénom	Nom	Note 1	Note 2	Note 3	Moyenne
	Maximum	20	20	20	
Thomas	Guyet	12,00	8,00	11,00	10,33
Olivier	Demzo	15,00	14,00	17,00	15,33
Franck	Meleville	8,00	5,00	9,00	7,33
Armelle	Stiprel	12,00	13,00	15,00	13,33
Virginie	Lucuvier	17,00	10,00	12,00	13,00
	Moyennes :	12,80	10,00	12,80	11,87

- **Enregistrement d'une macro sous Excel**
  - Ouvrir le fichier `ClasseurTravail_Base.xls` dans la feuille `base`
  - Placer votre curseur sur la case « Prénom » de la feuille Basse
    - Utilisation de « référence relatives » ??
  - Lancer l'enregistrement de votre macro
    - La nommer « comme vous voulez » puis validez ...
  - Effectuer les opérations suivantes
    - Calcul des moyennes par étudiants
    - Calcul de la moyenne de classe
    - La mise en forme : bordures + couleurs pour les moyennes



Exécution d'une macro

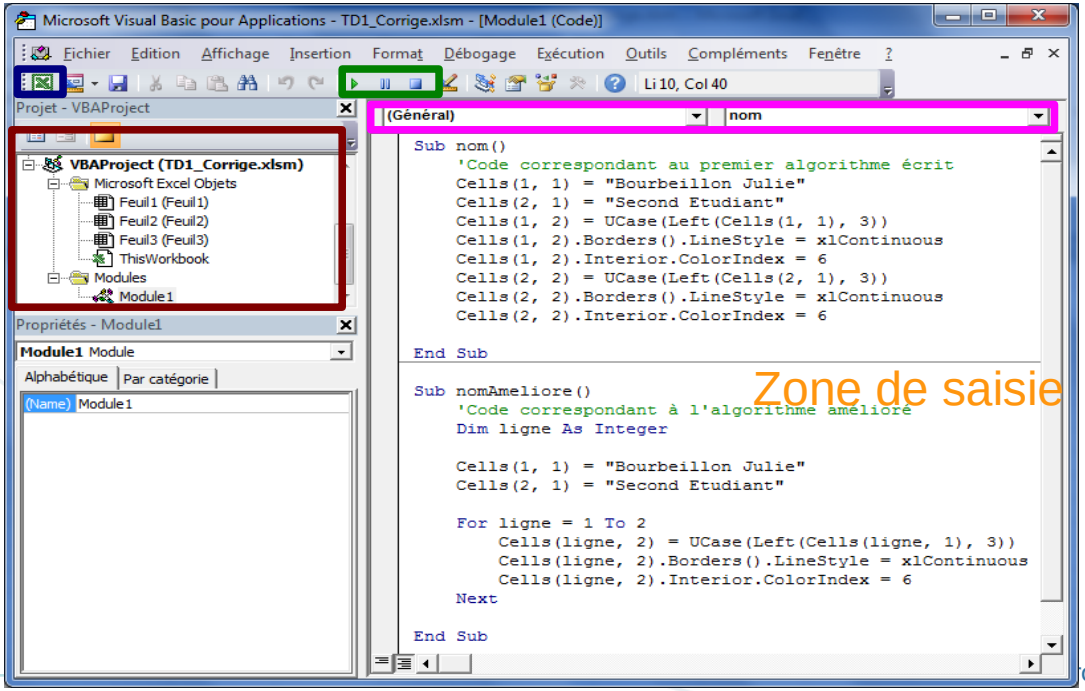
- **Exécution d'une macro**
  - Principe : Pour exécuter votre macro, ouvrez la fenêtre de macro **Alt+F8,** sélectionner votre macro puis cliquer sur **Exécuter**
  - Sur la feuille « `Test_Execution1` »
    - Placer le curseur sur la cellule « Prénom »
    - Lancer la macro
  - Sur la feuille « `Test_Execution2` »
    - Placer le curseur sur la cellule « Prénom »
    - Lancer la macro
    - Placer le curseur sur la cellule **B12**
    - Lancer la macro
  - Constater et analyser

À l'intérieur d'une macro …

- **Avec l'icône macro lancer l'interface de modification d'une Macro**
  - **Alt+F8** puis sélectionner une macro et cliquer sur Editer
  - Touches de raccourcis : **alt+F11**

Retour mode Classeur

Navigateur de projet (par Fichier Excel)



Exemple de macro enregistrée

```
Range("G3").Select
ActiveCell.FormulaR1C1 = "Moyenne"
Range("G5").Select
ActiveCell.FormulaR1C1 = _
    "=SUMPRODUCT(RC[-3]:RC[-1],R4C[-3]:R4C[-1])/SUM(R4C[-3]:R4C[-1])*20"
Range("G5").Select
ActiveCell.FormulaR1C1 = _
    "=SUMPRODUCT(RC[-3]:RC[-1],R4C[-3]:R4C[-1])/SUM(R4C[-3]:R4C[-1])"
Range("G5").Select
Selection.AutoFill Destination:=Range("G5:G9"), Type:=xlFillDefault
Range("G5:G9").Select
Range("D10").Select
ActiveCell.FormulaR1C1 = "=AVERAGE(R[-5]C:R[-1]C)"
Range("D10").Select
Selection.AutoFill Destination:=Range("D10:G10"), Type:=xlFillDefault
```

```
Range("D10:G10").Select
Range("G3:G9").Select
Selection.Borders(xlDiagonalDown).LineStyle = xlNone
Selection.Borders(xlDiagonalUp).LineStyle = xlNone
With Selection.Borders(xlEdgeLeft)
    .LineStyle = xlContinuous
    .ColorIndex = 0
    .TintAndShade = 0
    .Weight = xlThin
End With
With Selection.Borders(xlEdgeTop)
    .LineStyle = xlContinuous
    .ColorIndex = 0
    .TintAndShade = 0
    .Weight = xlThin
End With
With Selection.Borders(xlEdgeBottom)
    .LineStyle = xlContinuous
    .ColorIndex = 0
    .TintAndShade = 0
    .Weight = xlThin
End With
With Selection.Borders(xlEdgeRight)
    .LineStyle = xlContinuous
    .ColorIndex = 0
    .TintAndShade = 0
    .Weight = xlThin
End With
With Selection.Borders(xlInsideVertical)
    .LineStyle = xlContinuous
    .ColorIndex = 0
    .TintAndShade = 0
    .Weight = xlThin
End With
With Selection.Borders(xlInsideHorizontal)
    .LineStyle = xlContinuous
    .ColorIndex = 0
    .TintAndShade = 0
    .Weight = xlThin
End With
Range("D10:G10").Select
Selection.Font.Bold = True
Range("I9").Select
```

VBA un langage orienté objet

- **Les « classes »**
  - caractérisée par ses **propriétés**
  - permet de faire certaines actions : les **méthodes**
  - Exemple : Vehicule
    - Propriétés : Nombre de roue, Nombre de siège, Moteur (O/N)
    - Méthodes : rouler, …
- **Les « instances de classe » ou « objets »**
  - Une instance est la réalisation concrète d'une classe
  - Exemples :
    - Citroen C1 immatriculée 345 FI 456 : 4 roues, 5 sièges, O
    - Vélo immatriculé 34523456 : 2 roues, 2 sièges, N

VBA un langage orienté objet

- **En VBA tout est « objet »**
  - Je ne reviendrai pas trop sur les classes
  - Il faut surtout retenir que **les objets ont des propriétés/méthodes** et que en fonction de la nature des objets (leur classe), ces propriétés changent !
- **Une notation importante : la notation pointée**
  - Donne la propriété d'un objet
  - La valeur de la propriété peut être modifiée

```
Range("G3").Select
ActiveCell.FormulaR1C1 = "Moyenne"
Selection.Borders(xlDiagonalDown).LineStyle = xlNone
Selection.Font.Bold = True
```

VBA un langage orienté objet

- **Notations pointées … simplifiées**
  - VBA utilise le contexte pour vous faciliter les écritures
    - **Range("G3").Select** est en fait
    - **ActiveWorkbook.ActiveSheet.Range("G3").Select**
  - La notation **With** permet de regrouper des préfixes

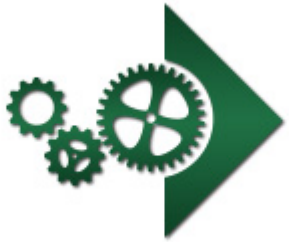
```
With Selection.Borders(xlEdgeBottom)
    .LineStyle = xlContinuous
    .ColorIndex = 0
    .TintAndShade = 0
    .Weight = xlThin
End With
```

```
Selection.Borders(xlEdgeBottom).LineStyle = xlContinuous
Selection.Borders(xlEdgeBottom).ColorIndex = 0
Selection.Borders(xlEdgeBottom).TintAndShade = 0
Selection.Borders(xlEdgeBottom).Weight = xlThin
```

Sources d'informations

- **Sur le langage et l'environnement**
  - Guide VBA (fonctions de base et propriétés les plus utiles)
  - Aide de l'éditeur VBA
  - Informations diverses sur l'internet
- **En cas de problèmes (80 % du tps à corriger les bugs!)**
  - Déjà, ***réfléchir***
    - Comparer code et résultat
    - Lire attentivement les messages d'erreur
  - ***Tester*** et ***re-tester*** avec de petites modifications
  - Rechercher le texte d'une erreur
    - Dans l'aide ou sur l'internet

Macros en VBA



- Forme générale
  - Des « procedures » structurés dans des « *Modules* »
  - Des commentaires
- Une macro est une procédure
  - Commence par « **Sub** *NomDeLaMacro()* »
  - Finit par « **End Sub** »
  - Contient une des instructions et des commentaires
- Une macro pourra être associée à des événements
  - Par ex : clic sur un bouton

Macros en VBA

```
' Commentaire
Sub NomMacro()
    ' Commentaire
    Instruction 1
    Instruction 2
    ...
    Instruction n
End Sub

' Blah blah
Sub NomMacro2()
    ' etc...
End Sub
```

- **Commentaires**
  - Texte libre
  - Destiné aux programmeurs
  - Ni trop, ni trop peu
    - « Ce que fait la macro »
    - Détails un peu compliqués
- **Procédures**
  - Contenu indenté : pour des question de lisibilité

Premier exemple

- **Modification de votre macro précédente**
  - Ouvrir votre macro en modification (**Alt+F11**)
    - Chercher votre macro dans le `Module 1`
  - Tester quelques modifications de votre macro
    - Changer les textes créés
    - Modifier les couleurs choisies
  - Ré-exécuter votre macro après modification (touche F5)

Plan du cours

- **Visual Basic for Applications**
- **Dans la macro il y a…**
- **Diviser pour régner**
- **Le problème derrière le clavier**

Dans la macro il y a …

- **Des instructions**
  - Une **instruction** est un expression qui indique une action a réaliser par l'ordinateur
  - Exemples
    - Modification du contenu d'une variable et de sa mise en forme
    - Mais bien d'autres …
  - Remarque : une macro enregistrée est une succession d'instructions

Second exemple

- **Création d'une première macro**
  - Vous travaillez maintenant sur la feuille « `Tarifs` »
  - Création d'un module
  - Création d'une procédure
  - Écrire la procédure pour qu'elle
    - Donne une valeur à une cellule (par ex. la cellule **G5**)
    - Calculer le contenu d'une cellule en fonction des autres
      - Prix total pour un produit
  - Exécution d'un procédure
    - Par l'interface
    - Utilisation de la touche F5

Fonctions utiles :
- `Cells(row, column)`
- opérations arithmétiques

Second exemple

- **Programmer des formules Excel en VBA**
  - Il y a **deux niveaux de calculs**
    - Les formules du tableau Excel calcule des valeurs qui sont affichées dans une feuille Excel
    - VBA calcule des contenus de cellules du tableau Excel : valeur … ou formule
  - Une **formule Excel** est une *valeur* pour un programme VBA

    Cells(4,7) = **‘’coucou‘‘**
    Cells(5,7)= **‘’=E5*D5+F5’’**

    - Entre guillemets : comme le texte saisie …
  - Il faut savoir jouer avec ses 2 niveaux

Dans la macro il y a...

- **Variable ??**
  - « Boîte » contenant des données
    - Correspond à une position en mémoire
    - Possède un type
  - Accès au contenu de la boîte par une « étiquette »
    - En l'occurrence le nom de la variable
  - Permet de stocker provisoirement des valeurs
    - C'est une « mémoire » interne au programme
    - Possède un **type** qui indique *comment* on manipule le contenu
  - On doit d'abord *déclarer* la variable pour que le programme sache comment la manipuler

Dans la macro il y a...

- **« Déclarer » une variable**
  - Créer la « boîte » et lui mettre une « étiquette »,
  - Déclarer ce qu'on voudra mettre dans la « boîte »
    - Définition de la taille (espace mémoire à réserver),
    - Définition du type de codage utilisé.
      - Le type est en fait optionnel pour VBA … mais recommandé

```
Dim X As Integer
Dim tot As Double
Dim txt As String
X=5          'Affectation de la variable
Cells(X,7) = ''coucou''    ' Utilisation de la variable
```

Types de variables de base

- **Types numériques**
  - **Integer** : entier de -2 147 483 648 à 2 147 483 647,
    - Attention : ça « tourne »

      2147483647 + 1 = -2147483648
  - **Double** : réel de $-1{,}79 \times 10^{308}$ à $1{,}79 \times 10^{308}$
    - Attention : plus le nombre est gros, moins il est précis
- **Types alphanumériques**
  - **Char** : un caractère (lettre, chiffre, ponctuation...)
  - **String** : une suite de caractères.
- **Type booléen**
  - Deux valeurs possibles : vrai ou faux.
  - **Boolean**
    - **True** pour vrai
    - **False** pour faux

Dans la macro il y a...

- **Expressions**
  - Une série d'opérations qui donnent un résultat
  - Opérateurs
    - Par exemple arithmétique basique : +, -, *, /
    - Priorité des opérateurs
  - Valeurs, références à des variables
  - « Sous-expressions » entre parenthèses
    - Pour contourner les priorités (par exemple « (1 + 2) * 3 »)
    - Pour rendre le code plus lisibles, parfois

Dans la macro il y a...

- **Expressions**
  - « 1 » : la valeur 1
  - « 1 + 2 » : résultat de l'addition des valeurs 1 et 2
  - « x + 2 » : résultat de l'addition du contenu de la variable x à la valeur 2
  - « x * (2 + y) » : résultat de la multiplication du contenu de la variable x par la somme de la valeur 2 et du contenu de la variable y

Petit exemple

- **Illustration de l'utilisation des variables dans VBA**
  - Calcul des totaux
    - Calculer le total vendu sans visualiser le prix des ventes de chaque ligne !

Ici, on ne demande pas de mettre une formule Excel qui calcule le prix dans la cellule, mais de calculer en VBA une valeur qui sera donnée comme valeur de cellule.

Dans la macro il y a...

- **Instructions affectant l'application**
  - Exemples
    - Modification du contenue d'une variable
    - Modification du texte d'une cellule
    - Modification de la mise en forme d'une cellule
    - Mais bien d'autres ...
  - La forme peut paraître un peu obscure
    - Il y a beaucoup fonctions différentes (et on se limite à Excel)
    - Consulter les aides (documents fournis et doc en ligne)
    - Faites vous vos habitudes : pratiquez et vous retiendrez !!

Dans la macro il y a...

- **Instructions affectant l'application**
  - Modification du texte d'une cellule

```
Sub MacroPerso()
    Cells( 1 , 1 ) = "lol"
End Sub
```

  - Ajout de bordures à une cellule

```
Sub MacroPerso2()
    Cells( 1 , 1 ).Borders.LineStyle = xlContinuous
End Sub
```

Dans la macro il y a...

- **Structure arborescente**
  - **Cells(y,x)** « contient »
    - **Borders** qui « contient »
      - **LineStyle** qui indique le style de ligne
      - **ColorIndex** qui indique un numéro couleur

```
Sub MacroPerso2Bis()
    Cells( 1 , 1 ).Borders.LineStyle = xlContinuous
    Cells( 1 , 1 ).Borders.ColorIndex = 4
End Sub
```

Dans la macro il y a...

- **Structure arborescente**
  - Bloc **With** … **End With**
  - Deux lignes de plus, mais le contenu est beaucoup plus lisible

```
Sub MacroPerso2Ter()
    With Cells( 1 , 1 ).Borders
        .LineStyle = xlContinuous
        .ColorIndex = 4
    End With
End Sub
```

Cells et Range

Cells vs Range

- Deux fonctions pour sélectionner des cellules
  - Cells : utilise des indices numériques au sein d'une feuille
  - Range :
    - Utilise les nommages de cellules (comme dans les formules)
    - Permet de sélectionner des plages de cellules

```
Sub MacroPerso2Ter()
    With Range( 'C3:H8' ).Borders
        .LineStyle = xlContinuous
        .ColorIndex = 4
    End With
End Sub
```

Cells et Range

- ***Cells* vs *Range***
  - Deux fonctions pour sélectionner des cellules
    - Cells : utilise des indices numériques au sein d'une feuille
    - Range :
      - Utilise les nommages de cellules (comme dans les formules)
      - Permet de sélectionner des plages de cellules
- **Plus de détails sur la sélection de cellules dans Excel**
    - Accès aux autres feuilles voire à d'autres documents possible !
      - https://support.microsoft.com/fr-fr/help/291308/how-to-select-cells-ranges-by-using-visual-basic-procedures-in-excel

Variables et objets

- **Une variable peut « contenir » un objet constitutif du tableau Excel**
  - Range (ensemble de cellules), Sheet (feuille), Workbook (classeur)
- **L'affectation d'un tel objet se fait via l'instruction Set**

```
Dim Rng As Range
Set Rng = Range("H1:I45")

Dim S As Sheets
Set S = ActiveWorkbook.Sheets(1)
```

Macro comme **fonction** … vers des fonctions personnalisées

- **Fonction**
  - Une fonction est une procédure qui **renvoie une valeur**
  - Du coup, une fonction a un type de retour
    - i.e. le type de la valeur que renvoie la fonction
- **Exemple**
  - « Je veux savoir quelle est le prix avec TVA si une cellule contient un certain nombre »
  - Paramètres :
    - La valeur d'une cellule ...
    - Type de TVA (5.5 vs 20 %)
  - Type de retour : valeur numérique

Macro comme **fonction** … vers des fonctions personnalisées

- **Fonction**
  - Bloc **Function**…**End Function**
  - Indique le type de retour

```
Function TVA( prixht As Double, type55 As Boolean) As Double
    ' …



End Function
```

Macro comme **fonction** … vers des fonctions personnalisées

- **Fonction**
  - Doit contenir un « assignement » au nom de la fonction elle-même
  - C'est ce qui détermine la *valeur renvoyée*

```
Function TVA( prixht As Double, type55 As Boolean) As Double
    ' …


    TVA = prixht * 1.205
End Function
```

Macro comme fonction … vers des **fonctions personnalisées**

- **La fonction peut être utilisée comme une nouvelle fonction excel personnalisée**
- **À tester**
  - Écrire la fonction ci-dessous dans un Module de votre classeur et enregistrez
  - Revenir dans la feuille de calculs
  - Créer une formule de type « =TVA(C5) »

```
Function TVA( prixht As Double) As Double
    TVA = prixht * 1.205
End Function
```

Ce qu'on a vu ...

- Une Macro VBA manuelle est presque inutile
- Une macro est un programme informatique qui décrit des opérations sur le tableau Excel
  - Modification du contenu d'une cellule
  - Modification de la mise en forme d'une cellule
- Les éléments du tableau Excel sont accessibles en VBA comme des propriétés d'objets
  - Les cellules sont accessibles par l'expression `Cells(r,c)`
- Une macro VBA peut définir une formule du tableur
- Il est possible de créer des fonctions personnalisées

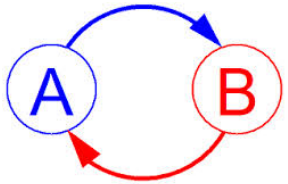
Dans la macro il y a...

- **Si on se limitait à ça...**
  - Seulement des tâches simples
  - Obligation de lister toutes les étapes manuellement
  - Aucune interaction avec le contenu du classeur
  - Aucune interaction avec l'utilisateur
  - Macros potentiellement énormes
- **On n'y gagnerait pas grand-chose**

Dans la macro il y a...

- **« Itération »**
  - Répéter une opération, à quelques détails près
  - Par exemple, « mettre une bordure autour des 100 premières cellules de la première colonne »

Dans la macro il y a...

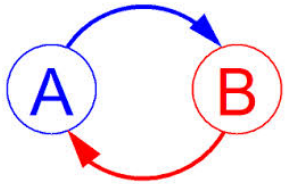


- **« Itération »**

```
Sub MacroPerso3()
    Cells(4, 8) = Cells(4, 5) * Cells(4, 6)
    Cells(5, 8) = Cells(5, 5) * Cells(5, 6)
    Cells(6, 8) = Cells(6, 5) * Cells(6, 6)
    Cells(7, 8) = Cells(7, 5) * Cells(7, 6)
    Cells(8, 8) = Cells(8, 5) * Cells(8, 6)

    ' Ici, le programmeur s'est pendu
End Sub
```

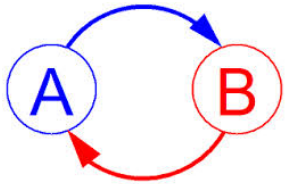
Dans la macro il y a...



- **« Itération »**
  - On voudrait **répéter l'instruction** pour chaque ligne
  - Deux choses nécessaires
    - Le numéro de la ligne à modifier
    - Une structure indiquant la répétition

```
Sub MacroPerso3()
    ???     ' Faire répéter la ligne suivante
            ' en faisant changer le numéro de ligne de 1 à 100
    Cells(???, 8) = Cells(???, 5) * Cells(???, 6)
    ' Ici on aura besoin du numéro de ligne
End Sub
```

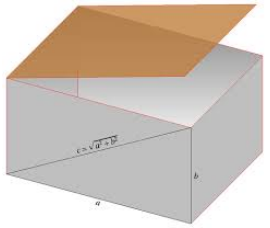
Dans la macro il y a...



- **« Itération »**
  - On voudrait répéter l'instruction pour chaque ligne
  - Deux choses nécessaires
    - **Le numéro de la ligne à modifier**
    - Une structure indiquant la répétition

```
Sub MacroPerso3()
    ???     ' Faire répéter la ligne suivante
            ' en faisant changer ligne de 1 à 100
    Cells(ligne, 8) = Cells(ligne, 5) * Cells(ligne, 6)
    ' Mais d'où vient « ligne » ?
End Sub
```

Dans la macro il y a...



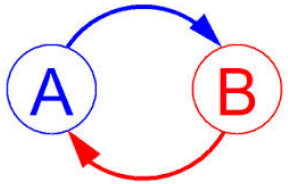
- **« Itération »**
  - On voudrait répéter l'instruction pour chaque ligne
  - Deux choses nécessaires
    - **Le numéro de la ligne à modifier**
      - Change à chaque itération
        → C'est une ***VARIABLE***
    - Une structure indiquant la répétition

Dans la macro il y a...

- **« Itération »**
  - On voudrait répéter l'instruction pour chaque ligne
  - Deux choses nécessaires
    - **Le numéro de la ligne à modifier**
    - Une structure indiquant la répétition

```
Sub MacroPerso3()
    Dim ligne As Integer
    ???     ' Faire répéter la ligne suivante
            ' en faisant changer ligne de 1 à 100
    Cells(ligne, 8) = Cells(ligne, 5) * Cells(ligne, 6)
End Sub
```

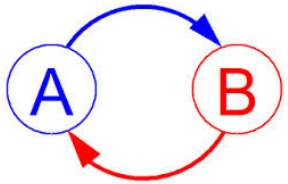
Dans la macro il y a...



- **« Itération »**
  - On voudrait répéter l'instruction pour chaque ligne
  - Deux choses nécessaires
    - Le numéro de la ligne à modifier
    - **Une structure indiquant la répétition**

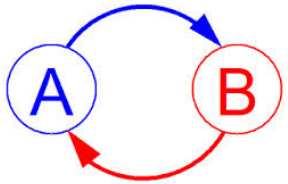
```
Sub MacroPerso3()
    Dim ligne As Integer
            ' Faire répéter la ligne suivante
            ' en faisant changer ligne de 1 à 100
    Cells(ligne, 8) = Cells(ligne, 5) * Cells(ligne, 6)
End Sub
```

Dans la macro il y a...



- **« Itération »**
  - On voudrait répéter l'instruction pour chaque ligne
  - Deux choses nécessaires
    - Le numéro de la ligne à modifier
    - **Une structure indiquant la répétition**
      - « Une boucle For »
      - Il s'agit d'un bloc
        - » Commence par **For**
        - » Finit par **Next**
      - Peut contenir n'importe quelle séquence d'instructions

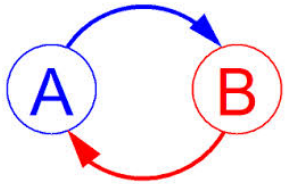
Dans la macro il y a...



- « Itération »
  - On voudrait répéter l'instruction pour chaque ligne
  - Deux choses nécessaires
    - Le numéro de la ligne à modifier
    - **Une structure indiquant la répétition**

```
Sub MacroPerso3()
    Dim ligne As Integer
    For ??? ' « ligne de 1 à 100 »
        Cells(ligne, 8) = Cells(ligne, 5) * Cells(ligne, 6)
    Next
End Sub
```

Dans la macro il y a...



- « Itération »
  - On voudrait répéter l'instruction pour chaque ligne
  - Deux choses nécessaires
    - Le numéro de la ligne à modifier
    - **Une structure indiquant la répétition**

```
Sub MacroPerso3()
    Dim ligne As Integer
    For ligne = 1 To 100
        Cells(ligne, 8) = Cells(ligne, 5) * Cells(ligne, 6)
    Next ligne
End Sub
```

Exemples pratiques

- **Traitement de toutes les lignes avec le calcul**
  - Quelles sont les bonnes limites ??
    - Faire attention à bien mettre ses limites
    - Mieux : calculer la limite à partir des données

```
Dim dl As Integer
dl = Range("H65536").End(xlUp).Row
```

  - Ajouter une ligne et re-tester
  - Pour information … dans les fonctions
    - Utilisation de la propriété count (cf TP)
- **Ajouter une colonne avec le calcul automatique d'un indice**

Dans la macro il y a...

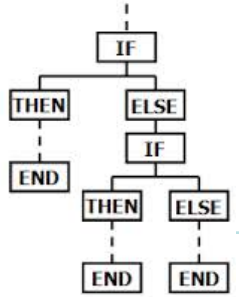
- « Itération »
  - **For**...**Next** est un bloc
    - Peut contenir ce qu'on veut
    - Donc d'autres boucles par exemple

```
Sub MacroPerso4()
    Dim li, co As Integer
    For li = 1 To 100
        For co = 1 To 100
            Cells( li , co ).Borders.LineStyle = xlContinuous
        Next co
    Next li
End Sub
```

Dans la macro il y a...

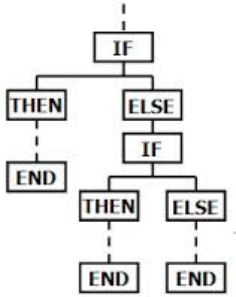
- **Si on se limitait à ça...**
  - Seulement des tâches simples
  - ~~Obligation de lister toutes les étapes manuellement~~
  - Aucune interaction avec le contenu du classeur
  - Aucune interaction avec l'utilisateur
  - Macros potentiellement énormes
- **On y gagnerait juste un peu de temps**

Dans la macro il y a...



- **Exécution conditionnelle**
  - « Si *quelque chose* alors faire *un truc* sinon faire *un autre truc* »
  - Par exemple, « si le numéro de la ligne est pair, alors mettre une bordure noire, sinon mettre une bordure verte »

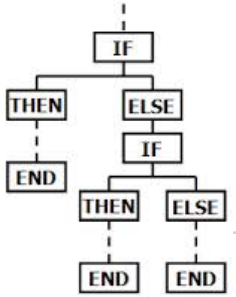
Dans la macro il y a...



- **Exécution conditionnelle**

```
Sub MacroPerso5()
    Dim ligne As Integer
    For ligne = 1 To 100
        Dim couleur As Integer
        ' Ici on détermine la couleur
        Cells( ligne , 1 ).Borders.LineStyle = xlContinuous
        Cells( ligne , 1 ).Borders.ColorIndex = couleur
    Next
End Sub
```

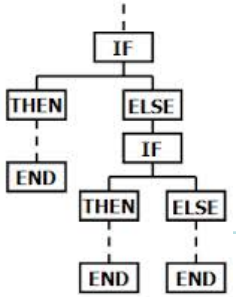
Dans la macro il y a...



- **Exécution conditionnelle**
  - Blocs **If** … **Then** … **Else** … **End If**

```
Sub MacroPerso5()
    Dim ligne As Integer
    For ligne = 1 To 100
        Dim couleur As Integer
        If ??? Then
            ??? ' Mettre 1 (noir) dans couleur
        Else
            ??? ' Mettre 4 (vert) dans couleur
        End If
        Cells( ligne , 1 ).Borders.LineStyle = xlContinuous
        Cells( ligne , 1 ).Borders.ColorIndex = couleur
    Next
End Sub
```

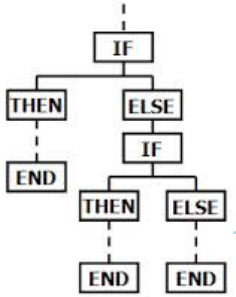
Dans la macro il y a...



- **Exécution conditionnelle**
  - *Assignement* : mettre une valeur dans une variable

```
Sub MacroPerso5()
    Dim ligne As Integer
    For ligne = 1 To 100
        Dim couleur As Integer
        If ??? Then
            couleur = 1 ' Noir
        Else
            couleur = 4 ' Vert
        End If
        Cells( ligne , 1 ).Borders.LineStyle = xlContinuous
        Cells( ligne , 1 ).Borders.ColorIndex = couleur
    Next
End Sub
```

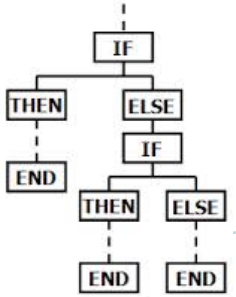
Dans la macro il y a...



- **Exécution conditionnelle**
  - Condition ?
    - Une expression dite *booléenne*
    - C'est-à-dire que son résultat est soit « vrai » soit « faux »
  - En l'occurrence on veut tester la parité du contenu de la variable *ligne*

```
If ??? Then
    couleur = 1 ' Noir
Else
    couleur = 4 ' Vert
End If
```

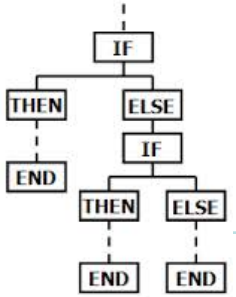
Dans la macro il y a...



- **Exécution conditionnelle**
  - Tester la parité ?
    - Une valeur est paire si le *reste* de sa division entière par 2 est 0
    - Une valeur est impaire si le *reste* de sa division entière par 2 est 1
  - Reste de la division entière ?
    - Opérateur « mod »

```
If (ligne Mod 2) = 0 Then
    couleur = 1 ' Noir
Else
    couleur = 4 ' Vert
End If
```

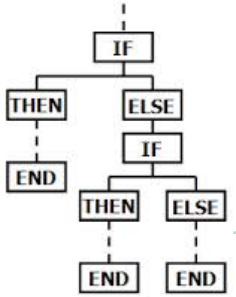
Dans la macro il y a...



- **Exécution conditionnelle**
  - Tester l'égalité ?
    - Opérateurs de comparaison
      - =, <, > (égal, inférieur, supérieur)
      - <= et >= (inférieur ou égal, supérieur ou égal)
      - <> (différent)

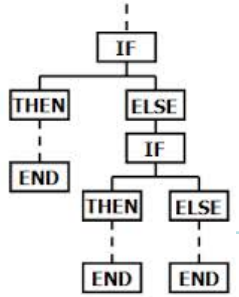
```
If ligne Mod 2 = 0 Then
    couleur = 1 ' Noir
Else
    couleur = 4 ' Vert
End If
```

Dans la macro il y a...



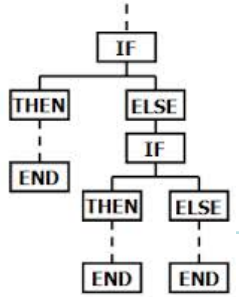
- **Expressions booléennes plus complexes**
  - Exemples :
    - « Si *quelque chose* **et** *autre chose* alors... »
    - « Si *quelque chose* **ou** *autre chose* alors... »
    - « Si **pas** *quelque chose* alors... »
  - Opérateurs logiques
    - **And, Or** et **Not**, respectivement

Dans la macro il y a...



- **Expressions conditionnelles**
  - Parfois on ne veut faire quelque chose que dans un cas
    - « Si X alors … sinon RIEN. »

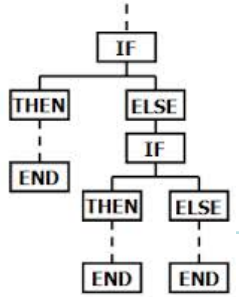
Dans la macro il y a...



- **Expressions conditionnelles**
  - Parfois on ne veut faire quelque chose que dans un cas
    - « Si X alors … sinon RIEN. »
    - On pourrait faire quelque chose comme ça...

```
If ligne Mod 2 = 0 Then
   Cells( ligne , 1 ) = "Gné !"
Else
   ' Rien.
End If
```

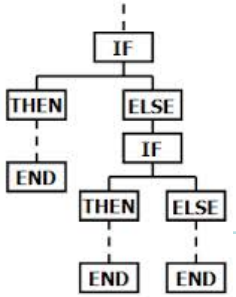
Dans la macro il y a...



- **Expressions conditionnelles**
  - Parfois on ne veut faire quelque chose que dans un cas
    - « Si X alors … sinon RIEN. »
    - Donc on fait comme ça :

```
If ligne Mod 2 = 0 Then
   Cells( ligne , 1 ) = "Gné !"
End If
```

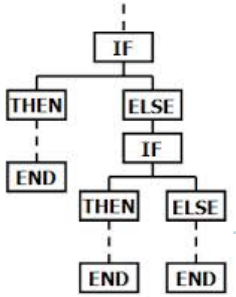
Dans la macro il y a...



- **Expressions conditionnelles**
  - Conditions multiples
    - « Si X alors … sinon si Y alors … sinon si Z alors … »
    - On pourrait imbriquer...

```
If ligne Mod 3 = 0 Then
   Cells( ligne , 1 ) = "Gné !"
Else
   If ligne Mod 3 = 1 Then
      Cells( ligne , 1 ) = "Blah !"
   Else
      Cells( ligne , 1 ) = "Haha !"
   End If
End If
```

Dans la macro il y a...



- **Expressions conditionnelles**
  - Conditions multiples
    - « Si X alors … sinon si Y alors … sinon si Z alors … »
    - Mieux … mais attention à l'ordre des tests

```
If ligne Mod 3 = 0 Then
   Cells( ligne , 1 ) = "Gné !"
ElseIf ligne Mod 3 = 1 Then
   Cells( ligne , 1 ) = "Blah !"
Else
   Cells( ligne , 1 ) = "Haha !"
End If
```

Exemple

- **Exemple pratique**
  - Mettre en rouge la cellule de commission lorsque que la commission totale sera inférieure à 40€, sinon mettre la cellule en vert
  - Infos utiles
    - La marge est données par `prix*quantité*commission/100`
    - Modification de la couleur de fond : `.Interior.ColorIndex`

Dans la macro il y a....

- **Itérations, bis**
  - Boucle **For** vue plus tôt très limitée
    - Permet simplement de « compter » entre deux valeurs
  - Parfois on veut faire quelque chose...
    - … *tant qu'*une condition n'est pas atteinte
    - … *jusqu'à ce qu'*une condition soit atteinte
  - L'un et l'autre sont équivalents
    - « Jusqu'à ce que X » → « Tant que pas X »

Dans la macro il y a....

- **Itérations, bis**
  - Exemple : « Mettre une bordure autour des 100 premières cellules de la première colonne, sauf si une cellule contient le mot FIN auquel cas on s'arrête ».

```
Sub MacroPerso6()
    Dim ligne As Integer
    For ligne = 1 To 100
        Cells( ligne , 1 ).Borders.LineStyle = xlContinuous
        ' Comment faire pour ne pas continuer ?
    Next
End Sub
```

Dans la macro il y a....

- **Itération, bis**
  - Boucle « **While** »

```
While ??? ' Condition
    ' Actions
Wend
```

  - Boucle « **Do While** »

```
Do
    ' Actions
Loop While ??? ' Condition
```

  - Différence ?
    - **Do**...**Loop While** s'exécute toujours une fois

Dans la macro il y a....

- **Itérations, bis**
  - On va donc utiliser une boucle **While**
    - Mais il va falloir compter « à la main »
  - Commençons par là.
    - Ci-dessous, boucle équivalente à la précédente.

```
Sub MacroPerso6()
    Dim ligne As Integer
    ligne = 1
    While ligne <= 100
        Cells( ligne , 1 ).Borders.LineStyle = xlContinuous
        ligne = ligne + 1
    Wend
End Sub
```

Dans la macro il y a....

- **Itérations, bis**
  - On veut aussi regarder la cellule actuelle
    - Tant qu'on n'est pas à la ligne 100 **et** que la cellule ne contient pas « FIN »

```
Sub MacroPerso6()
    Dim ligne As Integer
    ligne = 1
    While ligne <= 100 And Cells( ligne , 1 ).value <> "FIN"
        Cells( ligne , 1 ).Borders.LineStyle = xlContinuous
        ligne = ligne + 1
    Wend
End Sub
```

Dans la macro il y a....

- **Itérations, bis**
  - Hmmm… Là, il y a un petit bug...

```
Sub MacroPerso6()
    Dim ligne As Integer
    ligne = 1
    While ligne <= 100 And Cells( ligne , 1 ).value <> "FIN"
        Cells( ligne , 1 ).Borders.LineStyle = xlContinuous
        ligne = ligne + 1
    Wend
End Sub
```

Dans la macro il y a....

- **Itérations, bis**
  - Hmmm… Là, il y a un petit bug…
    - Quand on arrive à la ligne 101, on la teste quand même !
    - Pas un problème dans ce cas précis, mais cela pourrait.

```
Sub MacroPerso6()
    Dim ligne As Integer
    ligne = 1
    While ligne <= 100 And Cells( ligne , 1 ).value <> "FIN"
        Cells( ligne , 1 ).Borders.LineStyle = xlContinuous
        ligne = ligne + 1
    Wend
End Sub
```

Dans la macro il y a....

- **Itérations, bis**
  - De manière générale
    - Se méfier des conditions complexes
    - Attention aux boucles infinies !

```
Sub MacroQuiPlante()
    Dim ligne As Integer
    ligne = 1
    While ligne <= 100 And Cells( ligne , 1 ).value <> "FIN"
        Cells( ligne , 1 ).Borders.LineStyle = xlContinuous
    Wend
End Sub
```

Dans la macro il y a...

- **Si on se limitait à ça...**
  - ~~Seulement des tâches simples~~
  - ~~Obligation de lister toutes les étapes manuellement~~
  - ~~Aucune interaction avec le contenu du classeur~~
  - Aucune interaction directe avec l'utilisateur
  - Macros potentiellement énormes

Plan du cours

- **Qu'est-ce que la programmation ?**
- **Les langages de programmation**
- **Visual Basic for Applications**
- **Dans la macro il y a…**
- **Diviser pour régner**
- **Le problème derrière le clavier**

Diviser pour régner

- **Imaginons que l'on veuille…**
  - Colorier en vert toutes les cellules des 100 premières lignes de la colonne A contenant le mot « VERT »
  - Colorier en rouge toutes les cellules des 200 premières lignes de la colonne B contenant le mot « ROUGE »
  - Colorier en bleu toutes les cellules des 150 premières lignes de la colonne C contenant le mot « BLEU »

Diviser pour régner

- **Et de un…**

```
Sub MacroIdiote7()
    Dim ligne As Integer
    For ligne = 1 To 100
        If Cells( ligne , 1 ).value = "VERT" Then
            Cells( ligne , 1 ).Interior.ColorIndex = 4
        End If
    Next
End Sub
```

Diviser pour régner

- **Et de deux…**

```
Sub MacroIdiote7()
    Dim ligne As Integer
    For ligne = 1 To 100
        If Cells( ligne , 1 ).value = "VERT" Then
            Cells( ligne , 1 ).Interior.ColorIndex = 4
        End If
    Next
    For ligne = 1 To 200
        If Cells( ligne , 2 ).value = "ROUGE" Then
            Cells( ligne , 2 ).Interior.ColorIndex = 3
        End If
    Next
End Sub
```

Diviser pour régner

```
Sub MacroIdiote7()
    Dim ligne As Integer
    For ligne = 1 To 100
        If Cells( ligne , 1 ).value = "VERT" Then
            Cells( ligne , 1 ).Interior.ColorIndex = 4
        End If
    Next
    For ligne = 1 To 200
        If Cells( ligne , 2 ).value = "ROUGE" Then
            Cells( ligne , 2 ).Interior.ColorIndex = 3
        End If
    Next
    For ligne = 1 To 150
        If Cells( ligne , 3 ).value = "BLEU" Then
            Cells( ligne , 3 ).Interior.ColorIndex = 5
        End If
    Next
End Sub
```

Diviser pour régner

- **Problèmes de l'approche « Copy/paste »**
  - Ça devient long et illisible
  - S'il y a un bug dans le code, il faudra corriger partout
  - Facile d'oublier de modifier un détail
- **Solution**
  - Identifier les points communs
  - Identifier ce qui change
  - Extraire et séparer le code correspondant

Diviser pour régner

- **Pour en revenir à l'exemple...**

```
Dim ligne As Integer
For ligne = 1 To 100
    If Cells( ligne , 1 ).value = "VERT" Then
        Cells( ligne , 1 ).Interior.ColorIndex = 4
    End If
Next
```

Diviser pour régner

- **Pour en revenir à l'exemple...**

```
Dim ligne As Integer
For ligne = 1 To 200
    If Cells( ligne , 2 ).value = "ROUGE" Then
        Cells( ligne , 2 ).Interior.ColorIndex = 3
    End If
Next
```

Diviser pour régner

- **Pour en revenir à l'exemple...**

```
Dim ligne As Integer
For ligne = 1 To 150
    If Cells( ligne , 3 ).value = "BLEU" Then
        Cells( ligne , 3 ).Interior.ColorIndex = 5
    End If
Next
```

Diviser pour régner

- **Utiliser des variables serait insuffisant**
  - Il faudrait tout de même copier/coller le code de la boucle

```
Dim ligne , nLignes , couleur , colonne As Integer
Dim texte As String
nLignes = 100
couleur = 4
texte = "VERT"
colonne = 1
For ligne = 1 To nLignes
    If Cells( ligne , colonne ).value = texte Then
        Cells( ligne , colonne ).Interior.ColorIndex = couleur
    End If
Next
' etc...
```

Diviser pour régner

- **Procédures paramétrées**
  - Une procédure, c'est une série d'actions
    - Une procédure peut être appelée par une autre procédure
    - Techniquement, une macro est une procédure
  - Une procédure, c'est un bloc **Sub**...**End Sub**
  - Une procédure paramétrée a des, heu, paramètres
    - Valeurs passées à la procédure lors de son appel
    - Se comportent (presque) comme des variables

```
Sub MaProcedure( parametre1 As type , parametre2 As type ... )
    ' Du code !
End Sub
```

Diviser pour régner

- **L'exemple, de nouveau**
  - On extrait la boucle dans une procédure paramétrée
  - La procédure a besoin du nombre de lignes, du texte à trouver, de la colonne et de la couleur

```
Sub ColoriageTexte( colonne As Integer ,
                    nLignes As Integer ,
                    couleur As Integer ,
                    texte As String )
   For ligne = 1 To nLignes
      If Cells( ligne , colonne ).value = texte Then
         Cells( ligne , colonne ).Interior.ColorIndex = couleur
      End If
   Next
End Sub
```

Diviser pour régner

- **L'exemple, de nouveau**
  - Dans la macro, on se contente d'appeler la procédure 3 fois, avec 3 jeux de paramètres

```
Sub MacroIdiote7()
    Call ColoriageTexte( 1 , 100 , 4 , "VERT" )
    Call ColoriageTexte( 2 , 200 , 3 , "ROUGE" )
    Call ColoriageTexte( 3 , 150 , 5 , "BLEU" )
End Sub
```

Diviser pour régner

- **Procédures parfois insuffisantes**
  - Pas de communication depuis la procédure appelée vers la procédure appelante
  - Problème si le but de la procédure est par exemple de calculer quelque chose

Diviser pour régner

- **Fonction**
  - Une fonction est une procédure qui renvoie une valeur
  - Du coup, une fonction a un type de retour
    - i.e. le type de la valeur que renvoie la fonction
- **Exemple**
  - « Je veux savoir si une cellule contient un certain texte »
  - Paramètres :
    - Coordonnées de la cellule
    - Texte à examiner
  - Type de retour : booléen
    - « oui elle contient ça » ou « non elle ne contient pas ça »

Diviser pour régner

- **Fonction**
  - Bloc **Function**...**End Function**
  - Indique le type de retour

```
Function ContientTexte( ligne As Integer ,
                        colonne As Integer ,
                        texte As String ) As Boolean
    ' ...
End Function
```

Diviser pour régner

- **Fonction**
  - Doit contenir un « assignement » au nom de la fonction elle-même
  - C'est ce qui détermine la *valeur renvoyée*

```
Function ContientTexte( ligne As Integer ,
                        colonne As Integer ,
                        texte As String ) As String
   ContientTexte = ( Cells( ligne, colonne ).Value = texte )
End Function
```

Diviser pour régner

- **Fonction**
  - Utilisation dans une expression
    - Par exemple assignement à une variable

```
Dim tutu As Boolean
tutu = ContientTexte( 1 , 1 , "TUTU" )
```

    - Ou dans une instruction conditionnelle

```
Sub MacroIdiote6Bis()
    Dim ligne As Integer
    ligne = 1
    While … Not ContientTexte( ligne , 1 , "TUTU" )
        Cells( ligne , 1 ).Borders.LineStyle = xlContinuous
        ligne = ligne + 1
    Wend
End Sub
```

Diviser pour régner

- **Si on se limitait à ça...**
  - ~~Seulement des tâches simples~~
  - ~~Obligation de lister toutes les étapes manuellement~~
  - ~~Aucune interaction avec le contenu du classeur~~
  - Aucune interaction directe avec l'utilisateur
  - ~~Macros potentiellement énormes~~

Plan du cours

- **Qu'est-ce que la programmation ?**
- **Les langages de programmation**
- **Visual Basic for Applications**
- **Dans la macro il y a…**
- **Diviser pour régner**
- **Le problème derrière le clavier**

Le problème derrière le clavier

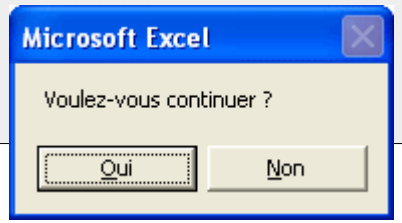
- **« Parler » à l'utilisateur**
  - On pourrait écrire dans le classeur Excel, par exemple
    - Difficile à voir pour lui
    - S'il y a plusieurs messages ?
  - On utilise des boîtes de dialogue
    - La plus élémentaire : la boîte de message

```
Sub Yo()
    MsgBox( "Bonjour" )
End Sub
```

Le problème derrière le clavier

- **Demander son avis à l'utilisateur**
  - En lui proposant des choix précis
    - → Boîte de message avec boutons personnalisés

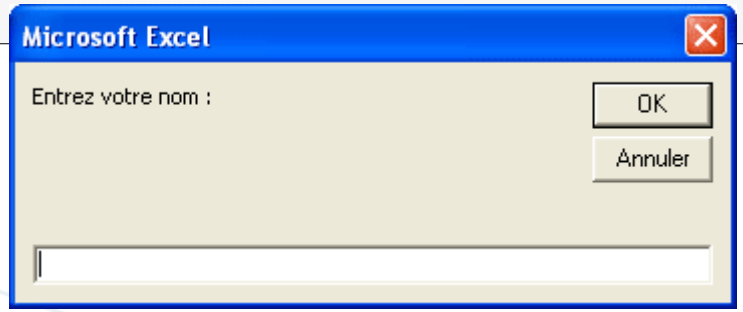
```
Sub PoserUneQuestionStupide()
    Dim reponse As Integer
    reponse = MsgBox( "Voulez-vous continuer ?" , vbYesNo )
    If reponse = vbYes Then
        MsgBox( "Z'êtes bien urbain." )
    End If
End Sub
```



Le problème derrière le clavier

- **Permettre à l'utilisateur de saisir du texte**
  - Boîte de saisie

```
Sub DetecteurDHorreursCosmiques()
    Dim reponse As String
    reponse = InputBox( "Entrez votre nom :" )
    If reponse = "Cthulhu" Then
        MsgBox( "Iä !" )
    End If
End Sub
```



Le problème derrière le clavier

- **Événements**
  - Certaines actions de l'utilisateur sont détectées par Excel
  - Excel essaie alors d'appeler certaines procédures dans le code VBA associé au classeur
  - Actions détectées ?
    - Sélection d'une plage de cellules
    - Saisie dans des cellules
    - Clic sur des boutons, etc
  - L'action détermine le nom de la procédure appelée

Le problème derrière le clavier

- **Quelques exemples pratiques**
  - Ajouter une question à l'utilisateur pour lui demander un numéro de couleur (indexée) à mettre sur les cellules dont la commission est <40€
  - Ajout d'un bouton pour déclencher votre commande

Quelques notions importante pour VBA sous Excel

- **Cellules actives ou sélectionnées**
  - Sélectionnées : sur fond bleu
  - Actives : bordures surlignées en gras
  - Elle peuvent être définies et mobilisées facilement
    - Fonctions **.Select** / **Selection.**
    - Fonctions **.Activate** / **ActiveCell.**