Programmation VBA sous Excel

Université Paris-Est Créteil
Serge Lhomme

Maître de conférences en Géographie
http://sergelhomme.fr
serge.lhomme@u-pec.fr

Les premiers pas

Objectif

Visual Basic pour Application (VBA) est un langage de programmation commun à toutes les applications de la suite Microsoft Office. Historiquement, VBA était aussi beaucoup utilisé pour programmer sous d'autres logiciels, comme par exemple le SIG ArcGIS. Aujourd'hui, le langage Python est privilégié.

Dans Excel, ce langage permet :

- D'automatiser des tâches répétitives : succession de tri et de sélection...
- De créer des formulaires personnalisés : créer des interfaces simplifiant l'utilisation d'Excel...
- De communiquer avec d'autres applications : notamment les autres applications Microsoft Office...
- De créer des algorithmes : compléter les manques d'Excel...

Les premiers pas

Utilisation de VBE

Ces programmes VBA sont souvent appelés macros. Pour faire simple, ces macros doivent être programmées dans des modules qui contiendront des procédures. Il existe deux possibilités pour créer des procédures VBA :

- Générer automatiquement du code à partir de l'enregistreur de macros ;
- Saisir directement sa procédure dans l'environnement Visual Basic Editor.

La première solution est plus simple, mais plus limitée. Ce cours se focalise donc sur la deuxième solution.

Pour avoir accès à VBE, il faut aller dans « Fichier -> Option -> Personnaliser le ruban », puis cocher la case « Développeur ».

Pour ouvrir VBE, allez dans l'onglet « Développeur », puis cliquez sur l'onglet « Visual Basic ».

Les premiers pas

Hello World comme première procédure

Pour créer un module, il est possible de faire un clic droit sur « VBA Project », puis de cliquer sur « Insertion », puis « Module ». Ensuite, un module nécessite de créer une procédure qui va contenir les instructions à suivre par le programme (Sub).

Message Box Hello World

```
Sub Hello_World()
MsgBox ("Bonjour le Monde")
End Sub
```

Cliquez sur l'icône « Enregistrer », puis choisissez un format autorisant les modules complémentaires VBA.

Vous pouvez fermer le module, puis cliquer dans l'onglet « Développeur » sur « Macros », vous allez alors voir apparaître la macro « Hello_World ». Vous pouvez cliquer sur « Executer » !

Les premiers pas

Premier calcul et notion d'affectation

Afin de ne pas refermer l'éditeur VBE lors de chaque test d'une procédure, vous pouvez cliquer sur le triangle vert. Attention, en cas d'erreur, il peut être nécessaire de réinitialiser le code en cliquant sur le carré.

Dans la plupart des cas, les programmes nécessitent de créer des variables auxquelles il convient d'affecter des valeurs. On utilise alors le signe $=$.

Les noms des variables doivent toujours commencer par une lettre.

Afficher que $2 + 3 = 5$

```
Sub Hello_World()
a = 2
b = 3
c = a + b
MsgBox ("Le résultat est de : " & c)
End Sub
```

Les premiers pas

Premier calcul et notion de déclaration

Attention

En théorie, avant de procéder à l'affectation d'une variable, il convient de la déclarer de manière explicite.

Pour cela, on utilise une instruction de déclaration (Dim, Public ou Private).

La déclaration permet à Excel de connaître le type de la variable et l'espace mémoire qu'il doit lui associer pour aller plus vite par la suite : entier (Integer), réel (Double), texte (String), date (Date), divers (Variant), Cellules (Range), Tableau (Array)...

Précédemment, la déclaration était implicite. Par défaut, les variables sont alors considérées de type Variant, ce qui est couteux en termes de temps de calcul, car il s'agit du type de variable le plus gourmand en espace mémoire...

Les premiers pas

Premier calcul et notion de déclaration

Afficher que $2 + 3 = 5$ avec déclaration des variables

```
Sub Hello_World()
Dim a As Integer
Dim b As Integer
Dim c As Integer
a = 2
b = 3
c = a + b
MsgBox ("Le résultat est de : " & c)
End Sub
```

Déclarer de manière explicite les variables prend du temps. Sachant que les programmes que l'on va développer par la suite restent simples, nous déclarerons les variables de manière implicite. C'est un choix très critiquable !

Les premiers pas

Première fonction

Une fonction est une procédure qui renvoie une variable. Les fonctions permettent d'éviter de réécrire plusieurs fois les mêmes séquences de code et d'en simplifier l'écriture et la lecture.

Afficher d'une manière plus élégante que $2 + 3 = 5$

```
Function Calcul(val1,val2)
Calcul = val1 + Val2
End Function

Sub Hello_World()
a = 2
b = 3
c = Calcul(a,b)
MsgBox ("Le résultat est de : " & c)
End Sub
```

Les premiers pas

Interagir avec les cellules

Si l'on programme en VBA sur Excel, c'est notamment parce que de nombreuses données sont directement disponibles dans des classeurs Excel, qui possèdent des feuilles, elles-mêmes composées de cellules.

Pour accéder aux valeurs d'une ou plusieurs cellules, on peut simplement utiliser l'objet Range en utilisant le référencement classique d'Excel (Absolu) et les règles d'affectation. De même, pour affecter une valeur à une cellule.

Utiliser Range

```
Sub Hello_Range()
a = Range("A1")
b = Range("A2")
c = Calcul(a, b)
Range("A3") = c
End Sub
```

Les premiers pas

Interagir avec les cellules

Tous les objets d'Excel, comme Range, possèdent des propriétés et des méthodes. En tapant le nom de l'objet suivi d'un point, vous avez accès aux propriétés et aux méthodes de l'objet.

Ainsi, l'objet Range possède des propriétés comme Borders ou Interior et des méthodes comme Cut ou Select. Certaines propriétés peuvent aussi être considérées comme des objets... On consultera l'aide pour savoir comment elles fonctionnent.

Utiliser Range et ses propriétés

```
Sub Hello_Prop()
c = 5
Range("A3") = c
Range("A3").Borders.LineStyle = xlContinuous
Range("A3").Interior.Color = RGB(255, 0, 0)
End Sub
```

Les premiers pas

Interagir avec les cellules

Dans les faits, il est aussi possible d'interagir simplement avec une cellule en utilisant l'objet Cells(nligne,ncolonne). Néanmoins, l'objet Range permet d'interagir directement avec plusieurs cellules.

Au même titre qu'avec les cellules, on peut interagir avec les feuilles de calcul à l'aide de l'objet Worksheets. Pour changer de feuille, on utilisera la méthode Activate.

Range Vs Cells

```
Sub Hello_Cellule()
Worksheets("Feuil2").Activate
Range("A1 :A2").Borders.LineStyle = xlContinuous
Range("A1 :A2").Interior.Color = rgb(255,140,0)
Worksheets("Feuil1").Cells(3, 1) = 7
End Sub
```

Les premiers pas

Interagir avec l'utilisateur par les cellules : les sélections

Un des moyens les plus simples pour interagir avec un utilisateur est de s'appuyer sur les cellules activées et sur ses sélections.

Pour cela, on peut s'appuyer sur les objets suivants : ActiveCell et Selection.

ActiveCell

```
Sub Hello_Active()
a = ActiveCell.Value
b = ActiveCell.Row
c = ActiveCell.Column
MsgBox ("Valeur de la cellule active : " & a)
MsgBox ("Ligne de la cellule active : " & b)
MsgBox ("Colonne de la cellule active : " & c)
End Sub
```

Les premiers pas

Interagir avec l'utilisateur par les cellules : les sélections

L'objet Selection fonctionne comme l'objet Range, c'est un objet Range...

Selection

```
Sub Hello_Selection()
nr = Selection.Rows.Count
nc = Selection.Columns.Count
Selection.Interior.Color = RGB(255, 0, 0)
MsgBox ("Nbr lignes sélectionnées : " & nr)
MsgBox ("Nbr colonnes sélectionnées : " & nc)
End Sub
```

Selection.Row et Selection.Column permettent de récupérer les valeurs de la cellule située en haut à gauche de la sélection.

Les premiers pas

Interagir directement avec l'utilisateur : les boîtes de dialogue

Néanmoins, pour interagir (échanger, communiquer) avec l'utilisateur, on privilégiera les boîtes de dialogue.

Il existe deux types de boîtes de dialogue : prédéfinies et personnalisées (les formulaires). Il est très simple d'utiliser les boîtes de dialogue prédéfinies, d'ailleurs MsgBox en est une.

Utiliser InputBox()

```
Sub Hello_Utilisateur()
a = InputBox("Quelle valeur 1 ?")
b = InputBox("Quelle valeur 2 ?")
c = Calcul(a, b)
Range("A3") = c
End Sub
```

Les premiers pas

Interagir directement avec l'utilisateur : les boîtes de dialogue

InputBox invite l'utilisateur à saisir une information.

Attention

Par défaut, on récupère des variables de type texte. Il faut donc les transformer en nombre pour effectuer un calcul. Pour cela, on peut utiliser la commande CDbl.

Utiliser InputBox()

```
Sub Hello_Utilisateur()
a = CDbl(InputBox("Quelle valeur 1 ?"))
b = CDbl(InputBox("Quelle valeur 2 ?"))
c = Calcul(a, b)
Range("A3") = c
End Sub
```

Les fondamentaux

Créer des boutons et des formulaires

VBA va être principalement utilisé pour personnaliser Excel et faciliter son fonctionnement afin de répondre aux besoins d'utilisateurs pressés ou pas forcément très compétents !

Ainsi, il convient de faciliter au maximum l'accès aux macros développées. Dans ce cadre, l'ajout d'un simple bouton au sein d'une feuille Excel permettant de lancer une macro est très courant.

Pour cela, dans l'onglet « Développeur », on pourra cliquer sur « Insérer », la première icône en haut à gauche permet alors de placer librement un bouton à l'aide d'un cliquer-glisser. Une fois relâché, il est demandé de lui associer (affecter) une macro (ici, on choisira Hello_World).

Au passage, on remarquera qu'il est possible de rajouter d'autres commandes.

Il faudra écrire un texte explicite sur le bouton. Pour cela (et pour modifier à tout moment une macro), effectuez un clic droit sur le bouton.

Les fondamentaux

Créer des boutons et des formulaires

Néanmoins, ce sont surtout les formulaires qui permettent de personnaliser au mieux Excel en permettant de créer des interfaces conviviales. Ce sont des boîtes de dialogue personnalisées, des boîtes de dialogue complexes.

Pour créer un formulaire, il suffit dans VBE de cliquer dans « Insertion -> UserForm ». Dans la fenêtre « Propriétés » à gauche, vous pouvez notamment le renommer « Pascaline » en changeant le texte de la propriété (Name).

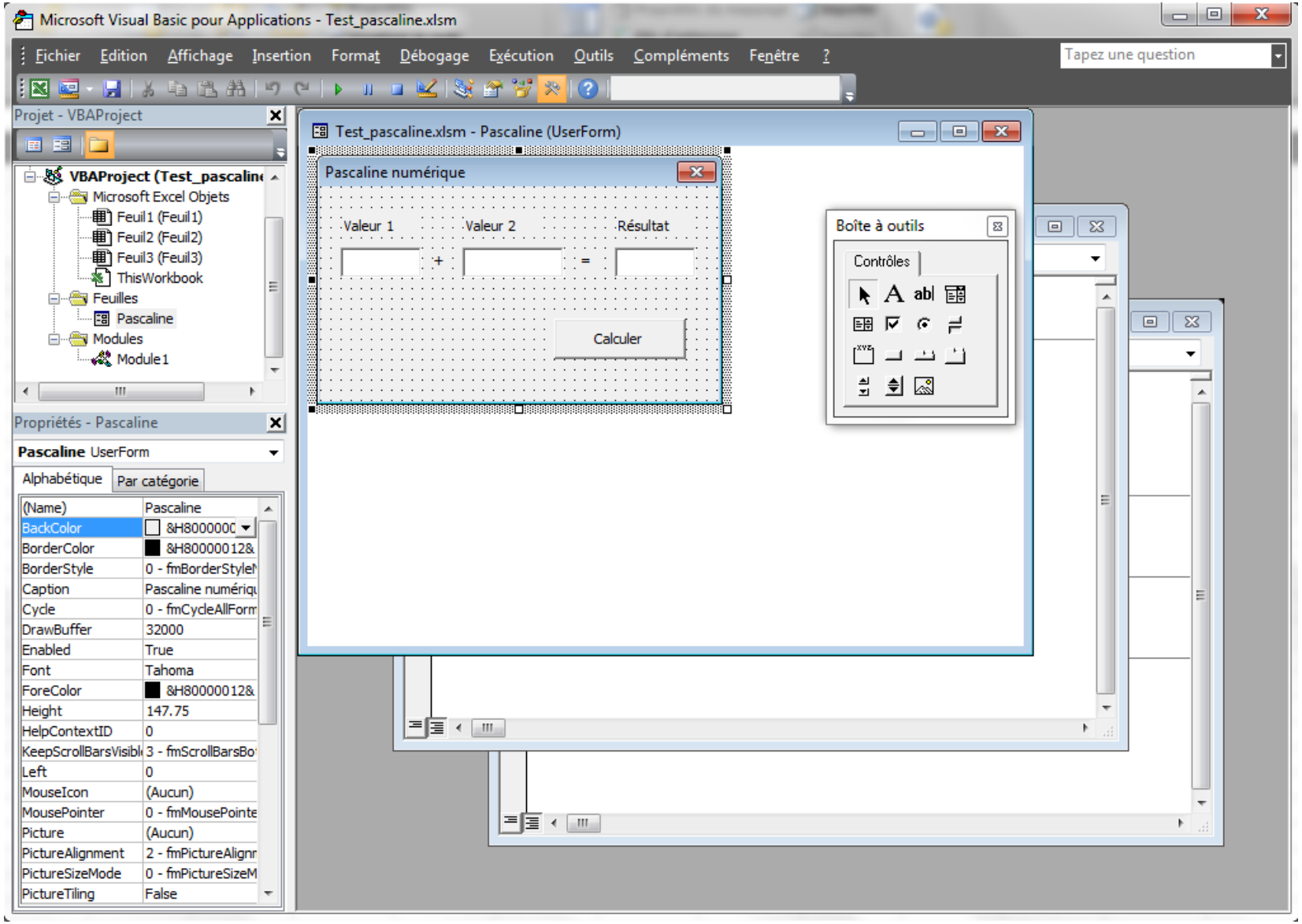
La propriété Caption permet de changer le texte de l'en-tête du formulaire.

La boîte à outils permet d'insérer simplement du texte (Label), des commandes (InputBox, CheckBox, ComboBox), des boutons (CommandButton) ou encore des images...

A l'aide des propriétés de chaque contrôle, il est possible de les placer et de les dimensionner précisément (Top, Left, Height, Width), de définir leur texte (Caption) ou leur disponibilité (Lock ou Enabled)...

Les fondamentaux

Créer des boutons et des formulaires



Pour que votre formulaire soit utile, il doit faire quelque chose... En l'occurrence, lorsque l'on appuie sur le bouton « Calculer », il faut que cela lance le calcul.

Pour cela, il suffit de double cliquer sur le bouton dans VBE. Une procédure NomDuBouton_Click() est créée. Il convient ensuite de mettre dans cette procédure le code correspondant.

A noter que cette procédure est privée (Private Sub). Elle n'apparaîtra donc pas dans les macros depuis le fichier Excel. De fait, ce n'est pas une macro, c'est une simple procédure.

Toutes les commandes créées sont des objets qui possèdent des propriétés. La propriété Value sera utile pour récupérer les valeurs des TextBox et afficher le résultat du calcul dans un TextBox.

Les fondamentaux

Créer des boutons et des formulaires

Procédure privée NomduBouton_Click()

```
Private Sub Button1_Click()
a = CDbl(TextBox1.Value)
b = CDbl(TextBox2.Value)
c = Calcul(a, b)
TextBox3.Value = c
End Sub
```

Pour afficher le formulaire, il suffit d'appliquer la méthode Show au nom du formulaire dans une procédure.

Procédure privée NomduBouton_Click()

```
Sub Hello_Pascal()
Pascaline.Show
End Sub
```

Les fondamentaux

Les boucles et les tests

Les structures de test sont au fondement de la programmation informatique. Elles permettent d'exécuter des instructions en fonction du résultat d'une condition.

Dans VBA elles suivent le protocole suivant : IF <Condition> Then <Instruction> (ELSE <Instruction>) END IF.

Les tests s'appuient sur des opérateurs de comparaison : égal à (=) ; supérieur à (>) ; différent de (<>)...

Et parfois sur des opérateurs logiques : et (And) ; ou (Or)...

Dans le cas de la pascaline, le test peut être utile afin d'éviter de générer une erreur incompréhensible pour l'utilisateur lorsque celui-ci oublie par exemple de remplir une valeur avant d'appuyer sur Calculer.

NomduBouton_Click() avec des tests utiles

```
Private Sub Button1_Click()
If TextBox1.Value <> "" And TextBox2.Value <> "" Then
   a = CDbl(TextBox1.Value)
   b = CDbl(TextBox2.Value)
   c = Calcul(a, b)
   TextBox3.Value = c
End If
If TextBox1.Value = "" Or TextBox2.Value = "" Then
   MsgBox("Attention à bien remplir Val1 et Val2")
End If
End Sub
```

Les fondamentaux

Les boucles et les tests

Les boucles permettent de répéter l'exécution d'un ensemble d'instructions. Il existe plusieurs types de boucle. La boucle For permet de réaliser une boucle associée à une variable entière (ou un pointeur) qui sera incrémentée à chaque itération.

Boucle For

```
Sub Boucle()
For i = 1 To 3 Step 1
   MsgBox ("Bonjour le monde pour la " & i & "eme
   fois")
Next i
End Sub
```

A noter que l'argument Step est facultatif ($= 1$) et qu'il est aussi possible d'utiliser l'instruction For Each lorsque l'on veut parcourir une collection d'objets.

Les fondamentaux

Boucle For

```
Sub Boucle()
a = Array(1,4,5,2)
For i = LBound(a) To UBound(a)
   MsgBox (a(i))
Next i
End Sub
```

Boucle For Each

```
Sub Boucle_ForEach()
a = Range("A1 :A2")
For Each i In a
   MsgBox (i)
Next i
End Sub
```

Les fondamentaux

Les boucles et les tests

La boucle While, francisée en boucle tant que, permet d'exécuter un ensemble d'instructions de façon répétée sur la base d'une condition booléenne. Si cette condition est respectée on continue, sinon on s'arrête.

Boucle While

```
Sub JeuCapitale()
Cap = ""
While Cap <> "Paris"
   Cap = InputBox("Quelle est la capitale de la
   France ? ")
Wend
End Sub
```

On utilise While à la place de For, lorsque l'on ne connaît pas le nombre d'itérations nécessaires à leur arrêt ou lorsque ce calcul s'avère complexe.

TD

Chaque programme doit être exécutable à l'aide d'un bouton spécialement créé à cet effet.

1. A l'aide d'une boucle, faire un programme qui demande de saisir une ville, puis renvoie le numéro de la ligne et la population de cette ville dans une seule fenêtre.

2. Faire un programme qui insère après le nom des villes une colonne intégrant le nom des villes en majuscule. Un deuxième clic sur le bouton supprime la colonne créée.

3. Faire un programme qui affiche dans une fenêtre les coordonnées géographiques d'une ville saisie en majuscule ou en minuscule.

4. Faire un formulaire avec 2 zones de texte permettant de saisir un nom de ville dans chaque zone. Une troisième zone de texte permet alors d'afficher la distance séparant ces deux villes après avoir appuyé sur un bouton Calcul.

TD

Réaliser un distancier

1. Créez un formulaire contenant notamment un bouton et deux grandes zones de texte permettant à l'utilisateur de saisir plusieurs noms de villes de départ et d'arrivée (indiquez à l'utilisateur que ces villes doivent être séparées par un ;).
2. Dans la procédure associée au bouton qui lancera le calcul du distancier, commencez par développer une procédure qui permettra de parcourir l'ensemble des villes rentrées par l'utilisateur, puis placez les comme titre du distancier final dans une nouvelle feuille.
3. Récupérez dans des tableaux les coordonnées de ces villes.
4. Calculez alors les distances qui séparent ces villes, puis placez ces valeurs dans le distancier et fermez le formulaire.
5. Placez un bouton dans une nouvelle feuille qui lancera le formulaire et recueillera le résultat du distancier sous le bouton.