A. Objectifs du chapitre

Objet élémentaire d'Excel, les cellules seront votre première étape sur la programmation VBA des objets Excel.

Sans parcourir l'intégralité des actions possibles avec les cellules en VBA, ce chapitre vous permettra de connaître les principales propriétés et méthodes des objets élémentaires que sont les **cellules** et **plages de cellules** sous Excel.

Vous verrez ensuite des exemples de code les plus fréquents avant de terminer en validant vos nouveaux acquis au travers d'exercices.

1. Objet et variable Range

Lorsque vous avez créé votre première macro avec l'Enregistreur de macros dans le chapitre L'enregistreur de macros, vous avez déjà fait la connaissance de l'objet **Range**.

a. Objet Range

Le type de données `Range` est le premier que vous découvrez dans la trousse à objets de VBA Excel.

Cet objet représente aussi bien une cellule seule, une plage de cellules ou encore une série de cellules non contiguës. Lorsque vous utilisez cet objet, la syntaxe générale est la suivante :

```
Range(Adresse)
```

Exemple 1 : syntaxe générale de l'objet Range

Selon vos besoins, l'adresse passée sous forme de chaîne de caractères peut représenter la ou les cellules que vous souhaitez manipuler :

```
Range("A1")
Range("C2:D5")
Range("A1,B2,G3")
Range("CelluleNommee")
```

Exemple 2 : possibilité de syntaxe pour l'objet Range

2. Variable de type Range

Tout comme vous l'avez fait pour des chaînes de caractères, des valeurs numériques ou des booléens, il est possible de travailler avec l'objet **Range** au travers d'une variable. Ce paragraphe vous indique comment déclarer, affecter et manipuler ensuite cet objet.

a. Déclaration

La déclaration d'une variable de type **Range** est similaire à celles vues jusqu'à présent.

```
Dim VotreCellule As Range
```

Exemple 3 : déclaration d'une variable de type Range

Il en est autrement pour l'affectation de valeurs.

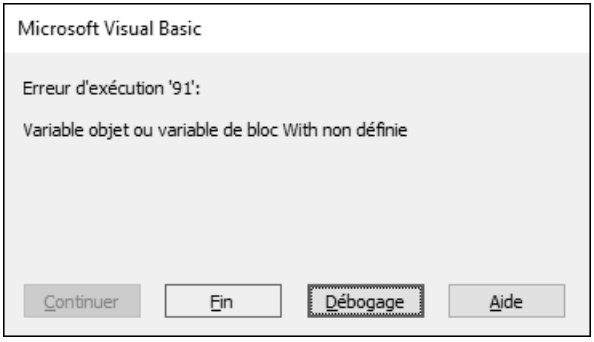
b. Affectation

Contrairement aux types de données élémentaires comme les chaînes de caractères ou les valeurs numériques, l'affectation pour le type de donnée Range a pour syntaxe générale la suivante :

```
Dim rCellule As Range
Set rCellule = Range("DateDuJour")
```

Exemple 4 : affectation d'une variable de type Range

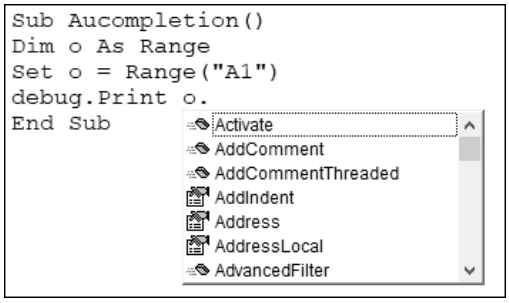
Vous pouvez constater la présence du mot-clé **Set** avant le nom de la variable **rCellule**. Si vous oubliez le mot-clé Set, une erreur 91 apparaîtra lors de l'exécution de la ligne de code.



L'usage du mot-clé Set devient nécessaire pour tous les types de données **Objet**.

c. Utilisation

Une fois que la variable est instanciée, vous pourrez profiter de l'autocomplétion lorsque vous commencerez à vouloir accéder aux propriétés et méthodes de l'objet `Range`.



Vous pourrez saisir les premières lettres de la propriété ou de la méthode et utiliser au choix la touche ⇥ ou la combinaison Ctrl Espace pour que la ligne se renseigne automatiquement.

B. Objet Cells

Il existe un objet plus petit encore que l'objet `Range`, qui permet lui aussi de manipuler les cellules ; il s'agit de l'objet `Cells`. Il est plus petit car là où l'objet `Range` permet de gérer une plage de cellules, l'objet `Cells` se limite à une seule cellule. Il n'est par exemple ainsi pas possible de faire référence à la plage de cellules A1:D3 avec un objet `Cells`.

Pour utiliser l'objet **`Cells`**, il existe deux syntaxes possibles, chacune fournissant à l'objet `Cells` une ligne et une colonne, coordonnées de la cellule.

```
'Syntaxe générale
Cells(indiceLigne, indiceColonne)
'Syntaxe avec des valeurs numériques uniquement
Debug.Print Cells(3,4) 'affiche le contenu de la cellule D3
'Syntaxe avec une valeur numérique pour la ligne et une chaîne de
caractères pour la colonne
Debug.Print Cells(5, "F") 'affiche le contenu la cellule F5
```

Exemple 5 : différentes syntaxes avec l'objet Cells

L'objet **`Cells`** est utilisable avec une variable de type **`Range`**.

```
Dim o As Range
Set o = Cells(1,3)
```

Exemple 6 : utilisation d'une variable de type Range pour l'objet Cells

Les types de données `Range` et `Cells` partageront des propriétés et méthodes communes.

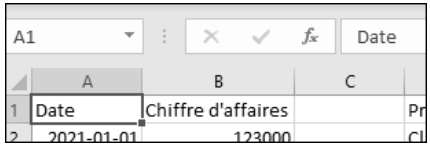
C. Quelques cellules particulières : ActiveCell, Selection et Target

Lorsque vous utilisez l'Enregistreur de macros également, certains mots-clés spécifiques peuvent apparaître pour définir une cellule ou une plage de cellules. Cette courte section a pour objectif de vous en expliquer les grandes lignes.

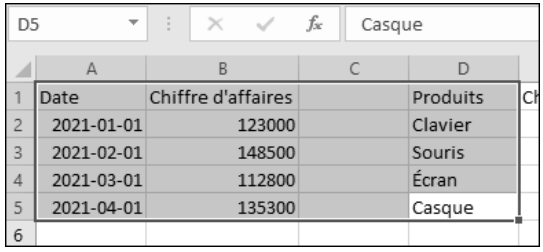
1. Cellule active : ActiveCell

Lorsque vous cliquez sur une cellule, on dit que vous l'activez. Cette **cellule active** est représentée par un objet natif en VBA : **`ActiveCell`**. Cet objet, de type `Range`, représente la cellule active dans votre classeur.

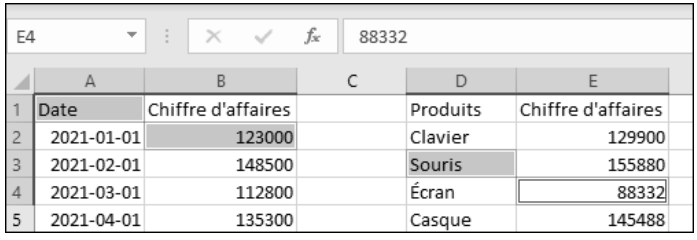
Ci-dessous un exemple de cellule active :



Lorsque vous sélectionnez une plage de cellules, la cellule à partir de laquelle vous effectuez votre sélection est la cellule active.



Enfin, si vous effectuez une série de sélections de cellules non adjacentes, la dernière cellule activée est la cellule active.



Vous pouvez voir le mot-clé `ActiveCell` dans l'exemple suivant :

```
Sub RemplirCellule()
   ActiveCell.FormulaR1C1 = "Bonjour"
End Sub
```

Exemple 7 : apparition du mot-clé ActiveCell depuis l'Enregistreur de macros.

2. Sélection active : Selection

Quand vous travaillez dans Excel et que vous sélectionnez une ou plusieurs cellules, VBA utilise un objet spécifique : **`Selection`**. Cet objet, de type `Range` comme `ActiveCell`, couvre un spectre similaire de fonctionnalités à celui-ci. Ce mot-clé pourra apparaître lorsque vous utilisez l'Enregistreur de macros.

Lorsque vous travaillez avec une seule cellule, l'objet `Selection` est identique à l'objet `ActiveCell`. Cependant, dès l'instant que vous utilisez une plage de cellules ou des cellules éparpillées, l'objet `Selection` vient trouver toute sa place, pour représenter l'ensemble des cellules sélectionnées.

Bien que la sélection puisse représenter plusieurs cellules, vous n'aurez toujours qu'une seule cellule active.

```
Sub UsageDeSelection()
   Range("D3").Select
   With Selection.Interior
       .Pattern = xlSolid
       .PatternColorIndex = xlAutomatic
       .Color = 255
       .TintAndShade = 0
       .PatternTintAndShade = 0
   End With
End Sub
```

Exemple 8 : cas d'utilisation du mot-clé Selection par l'Enregistreur de macros

3. Cellule(s) impliquée(s) dans les évènements Excel : Target

Comme vous le verrez dans le chapitre suivant Manipuler les feuilles Excel, le code VBA peut se déclencher automatiquement à partir d'événements qui ont lieu dans votre feuille ou votre classeur. Que cela soit un changement dans une feuille ou un double clic sur une cellule, ces évènements utilisent le mot-clé **`Target`** pour représenter la référence de l'objet qui est ciblé dans la procédure événementielle. Dans ces événements qui ont trait à des cellules, le type de l'objet **`Target`** sera **`Range`**.

```
Private Sub Worksheet_BeforeDoubleClick(ByVal Target As Range, Cancel As Boolean)
   If Target.Address = "$C$3" Then
       MsgBox "Interdiction de modifier cette cellule", vbCritical + vbOKOnly
   End If
End Sub
```

Exemple 9 : cas d'utilisation du mot clé Target dans un évènement de feuille Excel

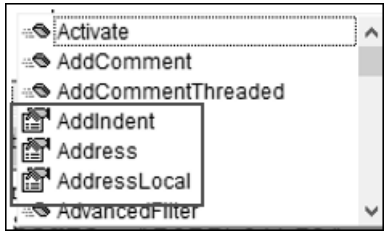
D. Les propriétés des cellules

Dans un premier temps, nous verrons ce qu'on appelle une **propriété** puis nous verrons les principales propriétés qui peuvent servir avec les cellules.

1. Définition d'une propriété

En informatique, les propriétés d'un objet sont ses caractéristiques, ce qui le définit. Prenons exemple d'une voiture, vous aurez parmi ses propriétés sa marque, son modèle, sa couleur, son année de sortie. Les propriétés peuvent être de tout type, (numérique, date, chaîne de caractères). Certaines propriétés seront modifiables, d'autres seulement lisibles.

En cours de programmation, vous pouvez reconnaître une propriété à l'icône de doigt pointé :



2. Le contenu d'une cellule : Value, Value2

Lorsque vous commencez à programmer avec Excel, la première chose qui vous intéresse avec une cellule sera la valeur qu'elle contient. Dans ce paragraphe, vous retrouverez une partie des lignes de code que vous avez découvertes avec l'Enregistreur de macros.

La valeur contenue dans une cellule est obtenue grâce à la propriété **`Value`**. VBA considère d'ailleurs cette propriété comme celle par défaut.

La syntaxe générale est la suivante :

```
Debug.Print Range("A1").Value 'Affiche la valeur contenue dans la cellule A1.
```

Exemple 10 : syntaxe générale de la propriété Value d'un objet Range

Cette propriété peut aussi bien être lue que modifiée par votre programme.

```
Range("A1").Value = 3 'Affecte la valeur 3 dans la cellule A1
MsgBox Range("A1").Value 'Affiche la valeur de la cellule A1
```

Exemple 11 : usage de la propriété Value

Selon le type de valeur contenue dans la cellule, le type de donnée retourné par la propriété **`Value`** sera automatiquement adapté par VBA. Il conviendra donc d'utiliser les bons types de données pour vos variables, au risque d'avoir une mauvaise interprétation des valeurs par votre programme.

Dans le cas de figure où vous avez les données suivantes dans les cellules :

	A	B	C
1	123	exemple de texte	2021-08-17

Le code suivant permettra de récupérer chacune des valeurs avec le bon type de donnée :

```
Dim iNumerique As Integer, sTexte As String, dtDate As Date
iNumerique = Range("A1").Value '123
sTexte = Range("B1").Value '"exemple de texte"
dtDate = Range("C1").Value '2021-08-17
```

Exemple 12 : récupération des valeurs au travers de la propriété Value

De la même façon, il est possible d'écrire des valeurs dans les cellules à partir des variables :

```
Dim strTexte As String
strTexte = "Bonjour"
Range("A1").Value = strTexte
```

Exemple 13 : affectation de valeur à une cellule par la propriété Value

Il existe une autre propriété, **`Value2`**, plus rarement utilisée mais que vous pourrez rencontrer dans les macros d'autres développeurs ; elle est similaire à `Value`, à ceci près qu'elle ne retourne pas de valeur de type **`Currency`** ou **`Date`** (ce type étant remplacé par le numéro de série, par exemple 2021-08-17 qui retournera 44425).

En reprenant le cas de l'illustration précédente, la valeur récupérée sera la suivante :

```
?Range("C1").Value2
44425
```

Exemple 14 : usage de la propriété Value2