



A1	BUREAUTIQUE	
	TP13	

Objectif : Utiliser les variables en VBA

Les variables permettent de stocker toutes sortes de données.

Voici un premier exemple :

```
'Affichage de la valeur de la variable dans une boîte de dialogue
Sub variables()

    'Déclaration de la variable
    Dim maVariable As Integer

    'Attribution d'une valeur à la variable
    maVariable = 12

    'Affichage de la valeur de maVariable dans une MsgBox
    MsgBox maVariable

End Sub
```

Cette première ligne de code est la déclaration de la variable (généralement placée en début de procédure).

```
Dim maVariable As Integer
```

- Dim : déclaration de la variable
- maVariable : nom choisi pour cette variable (sans espaces)
- As : déclaration du type de la variable
- Integer : type de la variable

Déclarer ses variables n'est pas obligatoire mais recommandé. Cela permet de s'y retrouver plus facilement, peut aider dans certains cas à résoudre plus facilement les problèmes, etc. Mieux vaut donc prendre l'habitude de déclarer correctement ses variables.

Le type de la variable indique la nature de son contenu (texte, nombres, date, etc.).

Une valeur est ensuite donnée à cette variable :

```
maVariable = 12
```

Et enfin, la valeur de la variable est affichée dans une boîte de dialogue :

```
MsgBox maVariable
```

MsgBox affiche une valeur dans une boîte de dialogue (les [boîtes de dialogue](#) seront détaillées dans quelques leçons).

Le résultat de ce code :



Si pour le moment vous ne comprenez pas bien l'intérêt d'utiliser des variables, soyez rassuré, les exemples abordés au cours des prochaines leçons vous en démontreront l'utilité.

LES TYPES DE VARIABLES

Nom	Type	Détails	Symbole
Byte	Numérique	Nombre entier de 0 à 255.	
Integer	Numérique	Nombre entier de -32'768 à 32'767.	%
Long	Numérique	Nombre entier de -2'147'483'648 à 2'147'483'647.	&
Currency	Numérique	Nombre à décimale fixe de -922'337'203'685'477.5808 à 922'337'203'685'477.5807.	@
Single	Numérique	Nombre à virgule flottante de -3.402823E38 à 3.402823E38.	!
Double	Numérique	Nombre à virgule flottante de -1.79769313486232E308 à 1.79769313486232E308.	#
String	Texte	Texte.	\$
Date	Date	Date et heure.	
Boolean	Boolean	True (vrai) ou False (faux).	
Object	Objet	Objet.	
Variant	Tous	Tout type de données (type par défaut si la variable n'est pas déclarée).	

Quelques exemples avec différents types :

```
'Exemple : nombre entier
Dim nbEntier As Integer
nbEntier = 12345

'Exemple : nombre à virgule
Dim nbVirgule As Single
nbVirgule = 123.45

'Exemple : texte
Dim varTexte As String
varTexte = "Texte"
```

```

'Exemple : date
Dim varDate As Date
varDate = "21/12/2021"

'Exemple : vrai/faux
Dim varBoolean As Boolean
varBoolean = True

'Exemple : objet (objet Worksheet pour cet exemple)
Dim varFeuille As Worksheet
Set varFeuille = Sheets("Feuil2") 'Set => attribution d'une valeur à une variable objet

'Exemple d'utilisation de la variable objet : activation de la feuille
varFeuille.Activate

```

Les symboles indiqués dans le tableau ci-dessus permettent de raccourcir les déclarations de variables.

Par soucis de lisibilité, ils ne seront pas utilisés dans les leçons mais voici tout de même un exemple :

```

Dim exemple As Integer
Dim exemple%

```

Ces deux lignes sont identiques.

Il est possible de forcer les déclarations de variables en plaçant Option Explicit tout au début du module (une erreur sera ainsi générée en cas d'oubli de déclaration).

LES TABLEAUX

Les variables permettent de stocker une seule valeur par variable, les tableaux permettent de stocker une multitude de valeurs par tableau (leur utilisation est proche de celle des variables).

Voici quelques exemples de déclarations :

```
'Exemple de déclaration de variable
Dim var1 As String

'Exemple de déclaration de tableau à 1 dimension
Dim tab1(4) As String

'Exemple de déclaration de tableau à 2 dimensions
Dim tab2(4, 3) As String
```

LE TABLEAU À 1 DIMENSION

```
'Exemple de déclaration de tableau à 1 dimension
Dim tab1(4) As String
```

Dans cette déclaration, il n'y a qu'un chiffre entre parenthèses, il s'agit donc d'un tableau à une dimension.

Ce chiffre indique également le nombre de cases du tableau. Dans le cas présent, tab1(4) est un tableau dont les cases vont de 0 à 4, il s'agit donc d'un tableau de comportant 5 cases :

0

1

2

3

4

Et voici comment attribuer des valeurs aux 5 cases de ce tableau :

```
tab1(0) = "Valeur de la case 0"
tab1(1) = "Valeur de la case 1"
tab1(2) = "Valeur de la case 2"
tab1(3) = "Valeur de la case 3"
tab1(4) = "Valeur de la case 4"
```

La première case d'un tableau est le 0.

LE TABLEAU À 2 DIMENSIONS

```
'Exemple de déclaration de tableau à 2 dimensions
Dim tab2(4, 3) As String
```

0,0	0,1	0,2	0,3
1,0	1,1	1,2	1,3
2,0	2,1	2,2	2,3
3,0	3,1	3,2	3,3
4,0	4,1	4,2	4,3

Et voici comment attribuer des valeurs aux cases d'un tableau à 2 dimensions :

```
tab2(0, 0) = "Valeur de la case rouge"
tab2(4, 1) = "Valeur de la case verte"
tab2(2, 3) = "Valeur de la case bleue"
```

Nous reviendrons sur les tableaux plus tard dans ce cours.

LES CONSTANTES

Les constantes permettent de stocker des valeurs comme les variables, à la différence près qu'on ne peut pas les modifier (d'où leur nom) après les avoir déclarées.

Par exemple, ces quelques lignes calculent le montant de la TVA en fonction d'un taux de TVA de 12.34% :

```
Sub exemple()  
  
Cells(1, 1) = Cells(1, 2) * 0.1234  
Cells(2, 1) = Cells(2, 2) * 0.1234  
Cells(3, 1) = Cells(3, 2) * 0.1234  
Cells(4, 1) = Cells(4, 2) * 0.1234  
Cells(5, 1) = Cells(5, 2) * 0.1234  
  
End Sub
```

Pour éviter les répétitions et faciliter la lecture de ce code, il est possible de déclarer le taux de TVA sous forme de constante :

```
Sub exemple()  
  
'Déclaration de la constante + attribution de sa valeur  
Const TAUX_TVA As Double = 0.1234  
  
Cells(1, 1) = Cells(1, 2) * TAUX_TVA  
Cells(2, 1) = Cells(2, 2) * TAUX_TVA  
Cells(3, 1) = Cells(3, 2) * TAUX_TVA  
Cells(4, 1) = Cells(4, 2) * TAUX_TVA  
Cells(5, 1) = Cells(5, 2) * TAUX_TVA  
  
End Sub
```

En utilisant une constante, le jour où le taux de TVA changera, il vous suffira de modifier une seule fois la valeur de la constante dans le code (au lieu de rechercher et remplacer toutes les valeurs 0.1234 dans le code).

Par convention, une constante se nomme en majuscules en séparant les mots par un _ (par exemple : EXEMPLE_DE_NOM).

LA PORTÉE DES VARIABLES

Si la variable est déclarée au début d'une procédure (Sub), elle ne peut être utilisée que dans cette même procédure. La valeur de la variable n'est pas conservée après l'exécution de la procédure.

```
Sub procedure1()  
    Dim var1 As Integer  
    '=> Utilisation de la variable dans la procédure uniquement  
End Sub  
  
Sub procedure2()  
    '=> Impossible d'utiliser var1 ici  
End Sub
```

Pour pouvoir utiliser une variable dans toutes les procédures d'un module, il suffit de la déclarer en début de module. De plus, cela permet de conserver la valeur de la variable jusqu'à la fermeture du classeur.

```
Dim var1 As Integer  
  
Sub procedure1()  
    '=> Utilisation de var1 possible  
End Sub  
  
Sub procedure2()  
    '=> Utilisation de var1 possible  
End Sub
```

Même principe pour utiliser une variable dans tous les modules, à la différence près que Dim est remplacé par Public :

```
Public var1 As Integer
```

Pour conserver la valeur d'une variable à la fin d'une procédure, remplacez Dim par Static :

```
Sub procedure1()  
    Static var1 As Integer  
End Sub
```

Pour conserver les valeurs de toutes les variables d'une procédure, ajoutez Static devant Sub :

```
Static Sub procedure1()  
    Dim var1 As Integer  
End Sub
```

CRÉER SON PROPRE TYPE DE VARIABLE

Nous n'allons pas nous attarder sur ce point, voici juste un exemple :

```
'Création d'un type de variable
Type Utilisateur
    Nom As String
    Prenom As String
End Type

Sub exemple()

    'Déclaration
    Dim user1 As Utilisateur

    'Attributions des valeurs à user1
    user1.Nom = "Smith"
    user1.Prenom = "John"

    'Exemple d'utilisation
    MsgBox user1.Nom & " " & user1.Prenom
End Sub
```