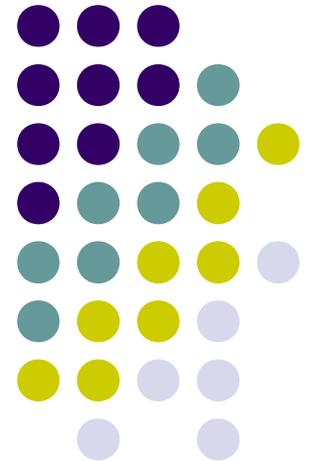
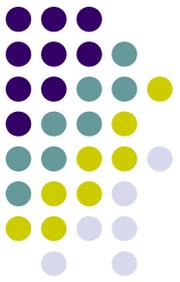


# Outils informatiques 2

## cours n°2: types et expressions

Claire Hanen  
Juliette Arnal





# Types de base

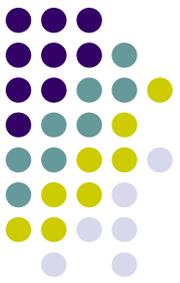
- **Byte** : nombre entier de 0 à 255
- **Integer** : nombre entier de  $-32768$  à  $32767$
- **Boolean** : True ou False
- **String** : chaîne de caractères
- **Double** : nombre décimal en double précision
- **Variant** : lorsqu'il peut y avoir plusieurs types
- Il y a d'autres types qui ne seront pas utilisés dans le cadre du cours.

# Pourquoi distinguer les entiers des décimaux?

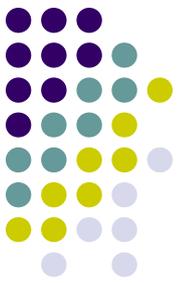


- Sur un nombre de bits fixé, on peut représenter un nombre fini de nombres.
- Ainsi, les types représentant des entiers correspondent à des intervalles de valeurs (ex le type integer)
- Or, dans un intervalle, il y a une infinité de décimaux!
- On a recours à une représentation des nombres à virgule flottante (ex le type double)

# Nombres à virgule flottante



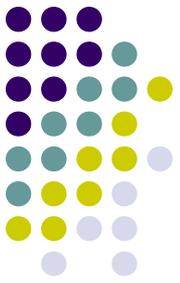
- Représentation d'un nombre sous **forme normalisée** (en binaire)
  - Équivalent décimal pour mieux comprendre: 345,67 s'écrit  $0.34567 * 10^3$
  - 0,0012 s'écrit  $0.12 * 10^{-2}$
  - Ce qu'il y a après la virgule s'appelle la mantisse
  - L'entier exposant de 10 s'appelle l'exposant
- Un nombre décimal peut être représenté par la mantisse sur un nombre fixe de bits, et l'exposant (nombre entier)
  - Plus les nombres représentés ont des **grands exposants**, plus ils sont **éloignés** les uns des autres.
  - Lorsqu'on fait des calculs avec deux nombres, on commence par « aligner les virgules » puis on réalise l'opération. Parfois le résultat n'est pas représentable.
  - Exemple: si on a 5 chiffres de mantisse,  $345,67 + 0,0012 = 345,6712$  n'est pas représentable; Le résultat serait interprété par l'ordinateur comme = 345,67
- **Tous les calculs faits sur les nombres décimaux sont potentiellement faux!**



# Rappel sur l'affectation

- La syntaxe d'une affectation est :
  - **nomdevariable=expression**
- Expression est soit
  - Une constante
  - Une variable
  - Une combinaison d'opérateurs et d'expressions
  - Un appel à une fonction
- Il doit y avoir concordance des types:
  - Le type de l'expression peut être traduite dans le type de la variable (si le type est différent)

# Expressions élémentaires



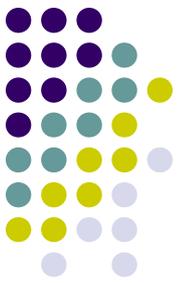
- Constantes
  - Constantes String: une suite de caractères entre guillemets.: " ceci est une phrase "
  - Constantes entières (Byte ou integer): écriture habituelle d'un nombre entier: 25, -68
  - Constantes Boolean: True, False
  - Constantes double: écriture habituelle d'un nombre décimal avec un point comme séparateur
  - Constantes prédéfinies dans les applications qui possèdent des noms Exemple : vbOK de type Byte, qui vaut 1

# Expressions élémentaires



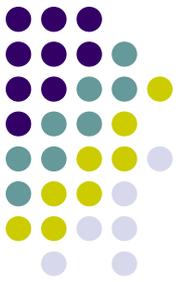
- Variables
- Un nom de variable (ou identificateur) est une expression. La valeur de l'expression est alors la valeur de la variable **au moment du calcul.**

# Concordance de types



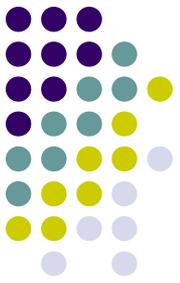
- VBA fait pour vous des conversions de type (sans le dire, en modifiant le codage des données) **nomvariable=expression**
  - Si la variable est de type integer, l'expression peut-être byte ou Integer
  - Si la variable est de type double, l'expression peut-être byte, Integer ou double
  - Si la variable est de type String, toutes les valeurs de l'expression sont transformées en chaînes de caractères.

# Exemples



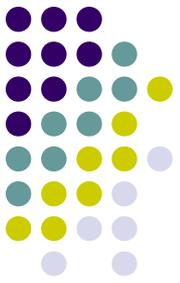
- Dim a As Integer
- Dim b As Double
- Dim s As String
- Dim v As boolean
- b=3.5
- s=b 's vaut la chaîne de caractères "3,5"
- a=5
- b=a 'b vaut 5- mais codé en double
- b=a+b
- v=True
- s=v 's a pour valeur le mot "True"

# Concordance de types : effets de bords (à éviter)



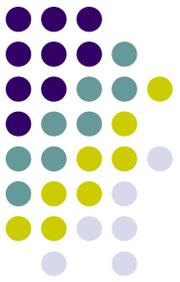
- VBA fait aussi des conversions plus inattendues:
- Si la variable est de type entier et l'expression de type double, il calcule la partie entière ou l'arrondi:
  - Exemple:
  - `Dim x As Integer`
  - `x=3.5` 'x vaut 4 (arrondi)
- Si la variable est Integer et l'expression de type boolean, il transcrit True en -1 et False en 0

# Concordance de types: effets de bord (à éviter)



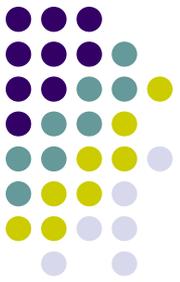
- Si la variable est entier ou double et l'expression de type string
  - il y a une erreur sauf si la chaîne de caractères est l'écriture d'un nombre:
    - Dim x As double
    - Dim s As string
    - S= "3,5" 'attention, séparateur = virgule à vérifier
    - x=s 'pas d'erreur d'exécution, x vaut 3,5
    - s= "bonjour"
    - x=s 'erreur d'exécution

# Expressions avec opérateurs binaires



- Syntaxe générale:
  - **expression1 opérateur expression2**
- On distinguera différents opérateurs:
  - Opérateurs arithmétiques (+, -, \*, /, Mod)
  - Opérateurs de comparaison (<, >, =, <=, >=, <>)
  - Opérateurs logiques (And, Or) + opérateur unaire not
  - Opérateurs sur les textes (&, +)

# Opérateurs arithmétiques



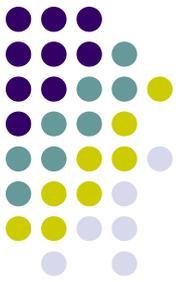
- +, -, \*, /, Mod
- Rappel sur l'opérateur Mod (modulo)
- Si  $X$  et  $Y$  entiers,  $X > Y$ , on peut exprimer de manière unique
  - $X = q * Y + r$ , avec  $r$  compris entre 0 et  $Y - 1$ .
  - $X \text{ Mod } Y = r$
- Exemple:  $18 \text{ Mod } 5 = 3$

# Opérateurs de comparaison



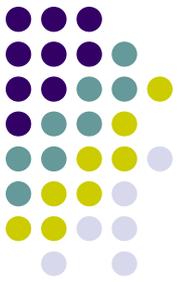
- $<$ ,  $>$ ,  $<=$ ,  $>=$ ,  $=$ ,  $<>$
- **Expression1 opérateur Expression2**
  - Expression 1 et expression 2 doivent être de type identique, ou pouvant être convertis l'un dans l'autre.
  - Le résultat du calcul est de type Boolean (True, False)

# Opérateurs logiques



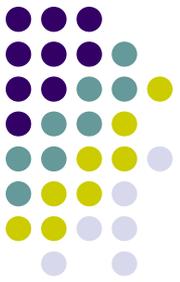
- And, Or
- **Expression1 opérateur Expression2**
  - Expression 1 et expression 2 doivent être de type boolean
  - Le résultat du calcul est de type Boolean (True, False)
- Not : not Expression1
  - Expression1 est de type boolean, et le résultat l'est aussi

# Exemple



- Sub essai()
- Dim note as double
- Dim ab as boolean
- note=InputBox("entrez votre note")
- ab=(note>=12) And not (note>=14)
- 'note comprise entre 12 et 13,999
- MsgBox(ab)
- 'affichage de True ou False
- End Sub

# Opérateurs sur les textes



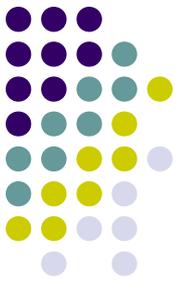
- &, +
- Expression1 opérateur Expression2
- Les deux opérateurs réalisent une concaténation, le résultat est de type string
  - Dim s As string
  - s="bonjour"
  - s=s & s 's vaut bonjourbonjour
  - s=s+s 's vaut bonjourbonjourbonjourbonjour

# Fonctions élémentaires



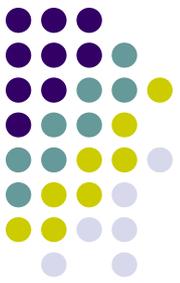
- VBA comporte un très grand nombre de fonctions prédéfinies. La syntaxe de l'appel:
  - **Nomdefonction (paramètre1,paramètre2,..etc)**
  - Les paramètres sont des expressions de type indiqué dans le mode d'emploi de la fonction (ou compatible)
  - Le type du résultat est également indiqué dans le mod d'emploi de la fonction.

# Fonctions utiles sur les nombres



- **Int(nombre )**: un seul paramètre de type double, calcule la partie entière inférieure
- **Rnd()**: sans paramètres, tire un nombre au hasard dans  $[0, 1[$  avec une loi uniforme.
- **sqr(nombre)**: un seul paramètre de type double, calcule la racine carrée.
- Autres fonctions mathématiques: **log** (logarithme népérien), **abs** (valeur absolue)

# Exemple



Un programme qui propose une multiplication et vérifie que la réponse de l'utilisateur est

correcte

```
Sub multiplication()
```

```
Dim x As Integer
```

```
Dim y As Integer
```

```
Dim res As Integer
```

```
Dim gagne As boolean
```

```
x=Int(Rnd()*9)+1 'Calcul d'un nombre aléatoire entier entre 1 et 10
```

```
y=Int(Rnd()*9)+1
```

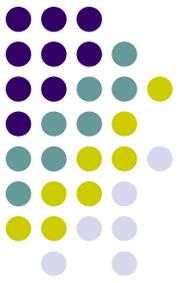
```
res=InputBox("Combien font " & x & " fois " & y & "?")
```

```
gagne=(res=x*y)
```

```
MsgBox("votre calcul est "& gagne)
```

```
End Sub
```

# Fonctions utiles sur les textes et autres.



- **Len( chaîne)**: un paramètre de type string, calcule le nombre de caractères de la chaîne.
- **Mid(chaîne, début,longueur)**: trois paramètres
  - Chaîne de type string
  - Début, longueur de type byte ou integer
  - Extrait de la chaîne longueur caractères à partir de la position début:
    - Exemple Mid("Bonjour ",2,4) vaut "onjo"
- Fonctions de date: **Date()**- donne la date du système sous forme variant, **Year( unedate)** donne l'année, **Month(unedate)** le mois, **Day(unedate)** le jour, etc.
  - Unedate peut être donné sous forme chaîne ou de nombre.