1.3 Les types des variables

1.3.1 Les variables numériques : le type réel et le type entier

Les variables de type numérique utilisées dans les algorithmes, ont comme domaine usuel, ceux fournis par les mathématiques : entiers ou réels

Remarque 8 :

1. Jamais les réels ne sont présents dans les algorithmes informatiques ; ce sont **toujours des approximations décimales.** [2]
2. On démontre que l'ensemble des décimaux et l'ensemble des dyadiques sont des ensembles denses dans $\mathbb{R}$; l'utilisation de ces nombres n'est donc pas injustifiée.
3. Les opérateurs utilisables sur ces éléments sont les opérateurs mathématiques classiques, ainsi que les opérateurs de comparaison $<$,$>$,$\leqslant$,$\geqslant$,$\neq$
4. Pourquoi 2 types de nombres ? Parce qu'en informatique, les opérations entre entiers et les opérations entre réels suivent des processus différents
5. Les réels, en informatique, sont aussi appelés **flottants**
6. En `Python`, en plus de l'addition et de la multiplication, d'autres opérations sur les entiers sont possibles :
   ⇒ **La division entière**, c'est à dire la recherche du quotient de m par n dans la division euclidienne ; elle s'exprime par un double slash `//`

   ```
   >>> A=15
   >>> B=9
   >>> A//B
   1
   ```

   ⇒ **Le modulo** ou la recherche du reste de la division euclidienne de m par n ; elle s'exprime par un pourcentage `%`

   ```
   >>> A=15
   >>> B=9
   >>> A % B
   6
   ```

   On peut remarquer que nous avons :

   ```
   >>> A=(A//B)*B+(A % B)
   >>> A
   15
   ```

   C'est donc une « reconstitution »de A

Exemple 4 :

```
>>> nombre1=1.2
>>> nombre2=15
>>> var1= 2
>>> type(nombre1)
<class 'float'>
>>> type(nombre2)
<class 'int'>
>>> type(var1)
<class 'int'>
```

Nous avons donc affecté à des variables `nombre1`, `nombre2` et `var1` des nombres et avons appelé la fonction `type` qui nous précise le type de ces variables ; le type `float` désigne les flottants ou réels et le type `int` désigne les entiers *(Integer en anglais)*
Faisons, maintenant, des opérations avec ces nombres

2. Aller voir l'annexe « Structure des nombres en machine » dans le cours de L_0

```
>>> nombre1=12
>>> nombre2=15
>>> var1= 2
>>> resultat=(nombre1/nombre2)*var1
>>> print(resultat)
1.6
>>> type(nombre1)
<class 'int'>
>>> type(resultat)
<class 'float'>
```

Remarque 9 :

Une remarque sur les conversions :

1. On peut convertir un naturel en réel : cette opération ne fait pas perdre d'information : ainsi, 15 deviendra 15.0

```
>>> nombre1=12
>>> type(nombre1)
 <class 'int'>
>>> nombre1=float(nombre1)
>>> nombre1
 12.0
```

2. Par contre convertir un réel en entier fait perdre de l'information : 15.75 deviendra 15

```
>>> Z=15.75
>>> Z=int(Z)
>>> Z
15
```

3. D'autre part, en `Python`, si nous additionnons un réel à un entier, nous obtenons un réel. Exemple :

```
>>> Z=15.75
>>> type(Z)
<class 'float'>
>>> Y=225
>>> type(Y)
<class 'int'>
>>> X=Y+Z
>>> X
240.75
>>> type(X)
 <class 'float'>
```

Exercice 2 :

Petit exercice : prévoyez le résultat affiché dans les opérations réalisées en `Python`

1. $\dfrac{2\times 3}{2+1}$
2. $2\times\dfrac{3}{2}+1$
3. $2^{(3^4)}$

Exercice 3 :

Que fait cet algorithme si nous lui demandons d'imprimer B ?

```
>>> A = input('Donner une valeur à A: ')
>>> A = int(A)
>>> B=A*A
>>> B=B*B
>>> B=B*A
>>> B=B*B
>>> print("Le résultat donne ",B)
```

Remarque 10 :

En `Python`, il est possible d'utiliser des opérateurs pour une variable de type caractère ; ce type s'appelle, en `Python`, `str` *(De l'anglais « String » caractère)*.

1. **Longueur**
   `len` est la fonction qui donne la longueur d'une chaîne de caractères.

```
>>> Caractere="Chaine de caractères"
>>> Caractere
'Chaine de caractères'
>>> type(Caractere)
<class 'str'>
>>> len(Caractere)
20
```

2. **Concaténation**
   + est l'opérateur de concaténation

```
>>> S1="Bonjour"
>>> S2=" tout le monde"
>>> S3=S1+S2
>>> S3
'Bonjour tout le monde'
```

3. **Répétition**
   * est l'opérateur de répétition

```
>>> S4=S3*3
>>> S4
'Bonjour tout le mondeBonjour tout le mondeBonjour tout le monde'
```

Remarque 11 :

On ne peut pas faire n'importe quoi avec les types ! Par exemple, il est impossible d'additionner ou concaténer des variables de différents types.

1. Dans le premier exemple, nous additionnons un entier `Nombre` à une variable de type caractère `"Les résidents"`, et nous obtenons un message d'erreur

```
>>> Nombre= 6
>>> type(Nombre)
<class 'int'>
>>> Expression = "Les résidents" + Nombre
Traceback (most recent call last):
  File "<pyshell#9>", line 1, in <module>
    Expression = "Les résidents" + Nombre
TypeError: can only concatenate str (not "int") to str
```

2. Que faire ? ? ...On peut transformer la variable entière `Nombre` en une variable caractère `str`

```
>>> Nombre= 6
>>> Nombre=str(Nombre)
>>> Expression = "Les résidents" + Nombre
>>> Expression
'Les résidents6'
```

1.3.2 Le type booléen

Les variables de type booléen utilisées dans les algorithmes ont, comme domaine usuel un ensemble formé de deux seules valeurs : {**VRAI**, **FAUX**}

Remarque 12 :

1. Les opérateurs admissibles sur les variables de ce type, sont celles que l'on trouve dans la logique : **ET**, **OU**, **NON**, dont il faut aller voir les tables de vérité dans le cours de mathématiques
2. En `Python` les valeurs affichées pour les valeurs prises par les variables booléennes sont {**TRUE**, **FALSE**}

Exemple 5 :

Il est intéressant de faire cet algorithme, pas à pas, sur le papier.

```
>>> booleen1=6<5
>>> booleen1
False
>>> booleen2 = not( 3>8)
>>> booleen2
True
>>> booleen3 =(5<6) or (3>8)
>>> booleen3
True
>>> booleen4 = (5<6) and (3>8)
>>> booleen4
False
```