Types et Arithmétique

Plan du T.P.

Retour au sommaire

1 Les types de variables

Les variables ne contiennent pas forcément des nombres. Elles peuvent aussi stocker des chaînes de caractères et bien d'autres types de données : listes de nombres, valeurs booléennes (vrai ou faux) ...

Saisissons par exemple dans la console :

```
>>> message = "Hello World !"
>>> message
'Hello World !'
```

En Python, toutes les variables ont un type qui correspond à la nature de la valeur stockée dans la variable : nombre entier, nombre à virgule flottante, chaîne de caractères, booléen (Vrai ou Faux) ...

La fonction `type` permet de connaître le type d'une variable :

```
>>> a = 5
>>> type(a)
int
>>> b = "Bonjour"
>>> type(b)
str
```

	Type	**Notation Python**
Nombre entier	Integer	`int`
Nombre à virgule flottante	Float	`float`
Chaîne de caractères	String	`str`
Booléen	Boolean	`bool`

1.1 Les entiers

Processeur et transistor

Le processeur [1] pourrait être qualifié de cerveau de l'ordinateur. Il est constitué de milliards de transistors. Un transistor se comporte comme un robinet de courant électrique :

- le courant passe,
- ou, le courant ne passe pas.

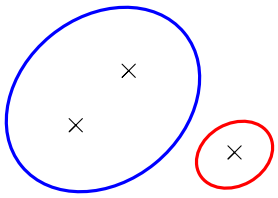
Pour ces raisons électriques, les ordinateurs n'utilisent que des 0 et des 1 :

- 0 : le courant ne passe pas,
- 1 : le courant passe.

L'ordinateur compte en **base 2** [2] : il n'utilise que des 0 et des 1 pour représenter les nombres. Ceci ne pose pas de problème pour les entiers puisque tout nombre entier peut s'écrire en base 2, avec un nombre fini de 0 et de 1.

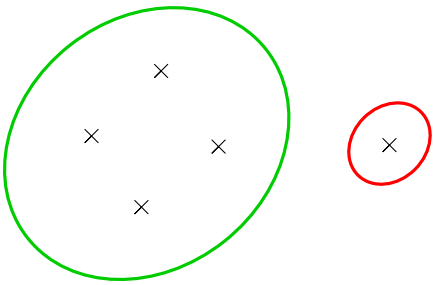
Par exemple :

- **3** s'écrit en base 2 : **11**



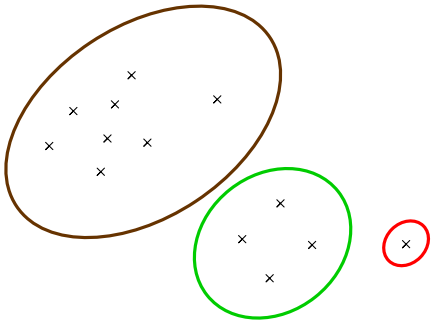
une deuzaine, et une unité

- **5** s'écrit en base 2 : **101**



1 quatraine, 0 deuzaine, 1 unité

- **11** s'écrit en base 2 : **1011**



1 huitaine, 1 quatraine, 0 deuzaine, 1 unité

Rappel :
En Python, les **nombres entiers** sont des objets de type **integer** (abrégé : `int`).

(1). https://fr.wikipedia.org/wiki/Microprocesseur
(2). http://pythonlandais.fr/codage/codage4.php

1.2 Les chaînes de caractères

Objectif :

Écrire une fonction `salutation(prenom)`, qui produira le résultat suivant lors de son appel en console :

```
>>> salutation("Muhammad")
'Bonjour Muhammad'
```

Exercice 1 : des chaînes de caractères dans la console

a. Deviner le résultat de l'instruction suivante. Vérifier dans la console.

```
>>> "Bonjour"
```

b. Deviner le résultat de l'instruction suivante. Vérifier dans la console.

```
>>> "Bonjour " + "Muhammad"
```

c. Deviner le résultat des instructions suivantes. Vérifier dans la console.

```
>>> prenom = "Muhammad"
>>> "Bonjour " + prenom
```

Synthèse :
- En Python, les **chaînes de caractères** peuvent être notées **entre guillemets**.
- Il est possible de mettre "bout à bout" deux chaînes de caractères à l'aide de l'opérateur **+**. On dit alors qu'on a **concaténer** les deux chaînes de caractères.

Exercice 2 : fonction de salutation

Écrire une fonction `salutation(prenom)` :
- prenant en argument une chaîne de caractères `prenom`,
- renvoyant à l'aide de l'instruction `return`, le message : Bonjour suivi du prénom fourni en argument :

```
>>> salutation("Muhammad")
'Bonjour Muhammad'
```

Exercice 3 : fonction majeur

Écrire une fonction `majeur(age)` :
- prenant en argument un entier `age`,
- renvoyant le message :
  - `'Vous êtes mineur'`, si l'âge est inférieur à 18,
  - `'Vous êtes majeur'`, sinon.

1.3 Les nombres à virgule flottante

Nous avons vu avec les entiers que les nombres manipulés par l'ordinateur sont exprimés en base 2. Cela soulève un certain nombre de problèmes pour les décimaux, qui ne peuvent pas tous être écrits en base 2 avec un nombre fini de 0 et de 1. Certains s'écrivent en base 2 avec une infinité de 0 et de 1 !

Python est alors amené à les tronquer et à utiliser une valeur approchée. Nous constaterons ainsi parfois une approximation lors des calculs avec des nombres à virgule flottante. Testons par exemple l'instruction suivante dans la console :

```
>>> 0.1 + 0.2
0.30000000000000004
```

Il en résulte que le test d'égalité "==" n'est pas adapté à la comparaison de nombres à virgules flottantes :

```
>>> 0.1 + 0.2 == 0.3
False
```

Exercice 4 :

a. Quel très célèbre théorème de géométrie, étudié au collège, pourrait nous amener à tester l'égalité de deux nombres à virgules flottantes ?

b. Un tel programme fournirait-t-il un résultat satisfaisant ?

Synthèse :
L'égalité entre nombres flottants n'a, pour ainsi dire, aucun sens. (3)

1.4 Les variables booléennes

Définition :
Une **expression booléenne** est une expression qui ne peut prendre que deux valeurs : **Vrai ou Faux**.

Exemple :

```
>>> 7 < 4
False
```

Définition :
Une **variable booléenne** est une variable qui ne peut prendre que deux valeurs : **Vrai ou Faux**.

Exemple : la variable `majeur` ci-dessous est une variable booléenne :

```
>>> age = 16
>>> majeur = (age < 18)
>>> majeur
True
```

Exercice 5 :
Pour chacune des séquences d'instructions du tableau suivant :

- prévoir le résultat renvoyé, ou ce que contiendra la variable écrite en dernière ligne,
- vérifier en saisissant ces instructions dans la console.

`>>> 3 + 1 == 4`	`>>> 3 + 1 = 5`	`>>> -20 + 12 != 8`
`>>> test = 5 < 3`	`>>> age = 16` `>>> tarif_reduit = age < 18`	`>>> age = 35` `>>> tarif_reduit = age < 18`

(3). Jean-Pierre Becirspahic : http://pc-etoile.schola.fr/wp-content/uploads/pdf-cours-info/nombres.pdf

2 L'opérateur modulo %

2.1 Les opérateurs

Nous avons déjà rencontré des opérateurs réalisant des **opérations entre deux nombres** :
- **x + y** : réalise l'addition de x et de y,
- **x - y** : réalise la soustraction de $x - y$,
- **x * y** : réalise la multiplication de x et de y„
- **x / y** : réalise la division de x par y„
- **x ** y** : élève x à la puissance y,

Nous avons aussi rencontré des opérateurs réalisant des **comparaisons entre deux nombres** :
- **x == y** : teste si x est égal à y (en Python, "le conditionnel se conjugue en double égal"),
- **x != y** : teste si x est différent de y,
- **x > y** : teste si x est supérieur strictement à y,
- **x < y** : teste si x est inférieur strictement à y,
- **x >= y** : teste si x est supérieur ou égal à y,
- **x <= y** : teste si x est inférieur ou égal à y.

2.2 L'opérateur modulo %

Il existe une infinité de nombres entiers naturels : 0, 1, 2, 3 ...

En mathématiques et en informatique, il sera souvent utile de travailler dans des structures ne contenant, par exemple, que 10 entiers : 0, 1, 2, 3, 4, 5, 6, 7, 8, 9.

Après, nous considérerons que :
- 10, c'est 0,
- 11, c'est 1,
- 12, c'est 2
- 13, c'est 3 ...

Cela peut sembler curieux ou inutile. Cependant, il est intéressant de remarquer que nous comptons déjà modulo 24 depuis la petite enfance. En effet, nous ne numérotons les heures que de 0 à 23, et revenons à 0 toutes les 24 heures.

Nous comptons ainsi modulo 24 :
- au bout de 24 heures, il est 0 heure,
- au bout de 25 heures, il est 1 heure,
- au bout de 26 heures, il est 2 heures,
- au bout de 48 heures, il est 0 heure ...

Exercice 6 :
Prévoir les résultats renvoyés par les instructions suivantes. Vérifier si besoin dans la console.

do.	`3 % 2`	**ré.**	`4 % 2`	**mi.**	`5 % 2`	**fa.**	`6 % 2`	**sol.**	`0 % 2`
la.	`1 % 2`	**si.**	`2 % 2`	**do.**	`7 % 2`				
a.	`13 % 10`	**b.**	`8 % 10`	**c.**	`27 % 10`	**d.**	`3 % 10`	**e.**	`134 % 10`
f.	`4 % 5`	**g.**	`25 % 5`	**h.**	`53 % 5`				
1.	`11 % 13`	**2.**	`13 % 13`	**3.**	`14 % 13`	**4.**	`15 % 13`	**5.**	`25 % 13`
6.	`26 % 13`	**7.**	`27 % 13`	**8.**	`28 % 13`				

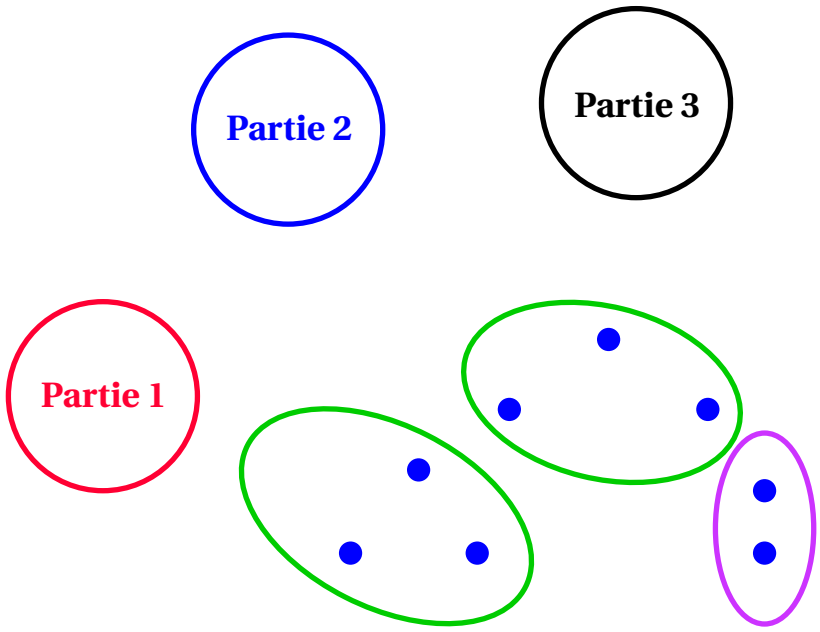
3 Où l'on retrouve ...

Exercice 7 :

a. À quelle notion mathématique célèbre associer l'opérateur modulo ?

b. En cas de doute, vous trouverez la réponse en haut de la page suivante.

```
>>> 'Euclide, bien sûr !'
```



Nous avons revu dans le chapitre 1 la division euclidienne d'un entier naturel a, par un entier naturel b non nul. Il s'agit de déterminer :

- le nombre de paquets de b unités que nous pouvons prélever dans a,
- le nombre d'unités restantes à l'issue du partage.

Définition :

- Soient a et b deux nombres entiers naturels tels que $b \neq 0$.
  Effectuer la **division euclidienne** de a par b, c'est trouver les deux entiers naturels q et r tels que :
  $$\boldsymbol{a = b \times q + r} \quad \text{et} \quad \boldsymbol{0 \leq r < b}.$$
- Le nombre $\boldsymbol{q}$ est appelé le **quotient** de la division euclidienne de $\boldsymbol{a}$ par $\boldsymbol{b}$. .
- Le nombre $\boldsymbol{r}$ est appelé le **reste** de la division euclidienne de $\boldsymbol{a}$ par $\boldsymbol{b}$.

Syntaxe :

En Python, le nombre $\boldsymbol{a\%b}$ est le **reste de la division euclidienne** de a par b.

4 Applications en arithmétique

Exercice 8 :
Soit n un entier naturel.

a. Compléter l'équivalence suivante en utilisant l'opérateur % : n est divisible par 3 $\Leftrightarrow$ .....

b. Écrire une fonction `divisible_par_3(n)` :

- prenant en argument un entier naturel `n`,
- et renvoyant : • `True`, si n est divisible par 3,
  • `False`, sinon.

c. Tester votre fonction avec les nombres suivants : 9, 7, 21, 341, 351.

d. Rappeler le critère de divisibilité par 3 étudié au collège.

e. Vérifier que nous obtenons bien les mêmes résultats que dans la question **c.** à l'aide de ce critère.

Exercice 9 :

a. Écrire une fonction `pair(n)`, prenant en argument un entier naturel `n`, et renvoyant :
- `True`, si n est pair,
- `False`, sinon.

b. Rappeler le critère de divisibilité par 2 étudié au collège.

Exercice 10 :
Soit n un entier naturel.

a. Compléter l'équivalence suivante en utilisant l'opérateur % : a est divisible par b $\Leftrightarrow$ .....

b. Écrire une fonction `divisible(a,b)` :
- prenant en argument un entier naturel `a` et un entier naturel `b` non nul,
- et renvoyant :
  - `True`, si a est divisible par b,
  - `False`, sinon.

c. Utiliser la fonction de la question **b.** pour tester si :
- 12 est divisible par 4 ?
- 144 est divisible par 12 ?
- 8043 est divisible par 17 ?
- 9 est divisible par 4 ?
- 169 est divisible par 13 ?
- 8041 est divisible par 17 ?
- 81 est divisible par 9 ?
- 400 est divisible par 20 ?