1 Les différents types de données

Dans tous les exemples donnés ici, on utilise des *commentaires* qui ne sont pas exécutés par python. Tout texte qui suit un dièse (#) est considéré comme un commentaire.

1.1 Les valeurs spéciales

Il y en a 3

```
True, False  # ce sont les 2 booléens
```

```
(True, False)
```

```
None  # "rien"
```

1.1.1 Les nombres

```
# Nombres entiers
a = 42
b = 5
```

On peut effectuer sur les entiers certaines opérations qui n'ont du sens qu'entre deux entiers :

```
a % b  # reste dans la division de a par b
a // b # quotient de cette division
```

Plus généralement, sur les nombres entiers, flottants ou même complexes, on peut effectuer les opérations :

```
a + b
a * b
a / b  # noter la simple barre
a ** b  # a puissance b
```

Les nombres complexes sont notés de la manière suivante :

```
1j  # le nombre i
2 + 3.14j
```

```
(2+3.14j)
```

1.2 Les types "conteneurs"

Ce sont des types de données un peu à part, qui permettent de stocker plusieurs informations

```
s = "Ceci est un texte. Noter les guillements pour le délimite
```

s est maintenant une *chaine de caractères*. Elle contient

```
len(s)
```

```
58
```

caractères. On peut accéder à chacun de ces caractères séparément. Ils sont numérotés à partir de 0 (et donc jusqu'à 57)

```
s[0], s[57]
```

```
('C', 'r')
```

```
L = [1, 2, True, "chaine"]  # L est une liste, noter les croch
```

Les listes en python peuvent contenir tout type d'objets, qui sont numérotés comme pour les chaînes, à partir de 0.

```
L[2]
```

```
True
```

```
len(L)
```

```
4
```

Une opération qui nous servira très souvent sur les listes :

```
L.append(3.14159)  # ajoute à la fin de L
L  # cette ligne sert juste à afficher la valeur de L
```

```
[1, 2, True, 'chaine', 3.14159, 3.14159]
```

Sur les deux types précédents, on peut effectuer les opérations suivantes :

```
"Une " + "opération"  # concaténation
```

```
'Une opération'
```

```
[1, 2] + [1, 2]  # idem
```

```
[1, 2, 1, 2]
```

```
"a"*7  # multiplication par un entier
```

```
'aaaaaaa'
```

```
[0]*4
```

```
[0, 0, 0, 0]
```

```
"a"*0, [0]*0  # chaîne et liste vides.
```

```
('', [])
```

2 Les structures de contrôle en python

Noter, dans tous les exemples suivants, **l'indentation** c'est à dire le nombre d'espace qui commence chaque ligne

2.1 Tests

Il s'agit de la structure la plus simple, qui premet de tester d'effectuer une action seulement à une certaine condition.

```
if condition:
    # instructions à exécuter si la condition est vraie
else:
    # à exécuter dans le cas où la condition est fausse.
```

```
if condition1:
    # instructions à exécuter si la condition est vraie
elif condition2:
    # la condition1 est fausse mais condition2 est vraie
# on peut ajouter autant de elif que l'on souhaite.
else:
    # à exécuter dans le cas où la condition est fausse.
```

La où les conditions sont des expressions python dont la valeur doit être un booléen (True ou False).

Les opérations suivantes retournent un booléen :

```
a == b # teste l'égalité
a < b
a <= b
a != b # teste si a et b sont différents
a is None  # teste si a est la valeur spéciale None
a in liste  # teste si a est dans la liste
```

2.2 Boucles

```
while condition:
    # répète les instructions tant que la condition
    # est vraie. Les instructions doivent modifier
    # la condition pour pouvoir sortir de la boucle.
```

```
for i in conteneur:
    # instructions exécutées une fois par valeur contenue
    # dans le conteneur.
    # Plus précisément, la variable i prend successivement
    # comme valeur chaque élément du conteneur et les instruct
    # sont exécutées une fois par valeur de i
```