

Range :

```
range(deb,fin [,pas])
```

La fonction "range" crée un objet non mutable (non modifiable) qui est une sequence sur laquelle on peut ITÉRER. Cette fonction ne retourne rien (on ne peut pas affecter range(...) a une variable

```
>>> range(1,4)      # crée la suite d'entiers [1,2,3]
>>> range(1,7,2)    # crée la suite d'entiers [1,3,5]
```

La boucle "for":

On peut ajouter un else (optionnel) a la fin du for qui sera exécuté a la fin du "for"

```
for <element> in <sequence> :
    instruction1
else :
    instruction2
```

L'instruction "break" sort de la boucle "for" sans passer par le "else". En revanche "continue" passe à l'iteration suivante et execute le "else" s'il est présent

ex :

```
>>> for elt in range(0,7) :
....   print(elt)

>>> L = ['a', 'b','c']      # 1
>>> for c in L :
....   print(c)
>>> for i in range(0,3) : #2
....   print(c)
```

ATTENTION : en Python on privilègue la première methode

PROBLEME ITERATION

```
>>> L = [1,2,3,4,6,5,7,8,9]
>>> for c in L :
....   if c % 2 == 0 :
....       L.remove(c)

[1,3,6,5,7,9] # ERREUR
```

Il vaut mieux utiliser une copie de L

```
>>> L = [1,2,3,4,6,5,7,8,9]
>>> for c in L[:]:
....   if c % 2 == 0 :
....       L.remove(c)

[1,3,5,7,9] # OK
```

Compréhension de listes :

Mécanisme qui permet d'appliquer (to map) une liste A dans une autre en appliquant une fonction sur chaque élément de A

```
[ f(e) for e in A ]
```

Où f est une fonction et A une sequence

```
>>> L = ['a' , 'b' , 'c']
>>> L2 = [2*e for e in L ]           # ['aa' , 'bb' , 'cc']
```

La sequence L n'est pas modifiée

```
>>> L = ['a' , 'b' , 'c']
>>> L2 = [3*e for e in L if e != 'a' ]           # ['bbb' , 'ccc']
```