Les boucles itératives ou boucles for en Python

1. La boucle for pour parcourir un élément

Une boucle `for` en python permet de parcourir l'ensemble des valeurs d'un objet si celui-ci le permet. Un tel objet est dit **itérable**.

Pour la fonction suivante :

```
def boucle(objet):
    for element in objet:
        print(element)
```

On a par exemple les retours suivants en console :

```
>>> boucle("AZER")
A
Z
E
R
>>> boucle(12)
Traceback (most recent call last):
  File "<pyshell>", line 1, in <module>
  File "/Users/davidlaignel/Desktop/2022-
23/nsi/tp/revisions_boucles/boucles_for_exemples_fiche.py", line 2, in boucle
    for element in objet:
TypeError: 'int' object is not iterable
```

Les objets itérables que nous verront cette année sont : les chaînes de caractères, les tuples et les listes.

```
>>> boucle("abcde") # un chaîne de caractère
a
b
c
d
e
>>> boucle((1,2,3,4)) # un tuple
1
2
3
4
>>> boucle([1,2,3,4]) # une liste
1
2
3
```

```
4
```

2. La boucle for .... in range ()

La boucle `for .... in range ()` permet de parcourir un ensemble de valeurs entières.

```
>>> for i in range(5): # parcourt les entiers successifs de 0 à 4
    print(i)

0
1
2
3
4
>>> for i in range(5,8): # parcourt les entiers successifs de 5 à 7
    print(i)

5
6
7
>>> for i in range(5,15,2): # parcourt les entiers successifs de 5 à 14 avec un pas de
2
    print(i)

5
7
9
11
13
>>> for i in range(5,14,2):# parcourt les entiers successifs de 5 à 13 avec un pas de 2
    print(i)

5
7
9
11
13
>>> for i in range(15,0,-1): # parcourt les entiers successifs de 15 à 1 avec un pas de
-1
    print(i)

15
14
13
12
11
10
9
8
```

```
7
6
5
4
3
2
1
>>> for i in range(8,7): # la boucle n'est pas effectuée puisque 8>7
    print(i)

>>>
```

La syntaxe générale est :

```
for valeur in range (val_depart,val_finale,pas):
   instructions à effectuer à chaque itération
```