Python 4 – **Boucles bornées en Python : for**

I – Boucle BORNEE

La boucle POUR est dite **bornée** car on peut connaître **à l'avance** le nombre de fois où elle va se dérouler.

II – Exactement pareil plusieurs fois

La syntaxe permettant de réaliser plusieurs fois la même chose est :

```
01 print("Avant boucle")    ← pas tabulée : avant la boucle
02
03 for _ in range(3):        ← for : déclaration d'une boucle
04     print("A")            ← tabulé : dans la boucle
05     print("-- B")         ← tabulé : dans la boucle
06
07 print("Après boucle")     ←  pas tabulée : après la boucle
```

La **tabulation** (4 espaces) permet à l'interpréteur de savoir où se trouvent les instructions à réaliser en boucle.

L’interpréteur Python exécute les lignes dans cet ordre :

L01

L03(1er tour)-L04-L05 - L03(2e tour)-L04-L05 - L03(2e tour)-L04-L05

L07

Il provoque cet affichage :

```
Avant boucle
A
-- B
A
-- B
A
-- B
Après boucle
```

III – Presque pareil plusieurs fois

On peut utiliser une **variable de boucle** dont la valeur varie à chaque tour de boucle pour réaliser presque la même chose.

En notant **for v in range(4)**, la variable de boucle **v** :

→ référence **0** lors du premier tour

→ référence **3** lors du dernier tour.

On aura bien fait 4 tours : 0, 1, 2, 3.

```
01 print("Avant boucle")
02
03 for k in range(3):          ← pour k variant de 0 à 2
04     print("A")              ← affiche le string "A"
05     print(k)                ← affiche le contenu de k
06
07 print("Après boucle")
```

L’interpréteur Python exécute les lignes dans cet ordre :

L01

L03(k=0)-L04-L05 - L03(k=1)-L04-L05 - L03(k=2)-L04-L05

L07

Il provoque cet affichage :

```
Avant boucle
A
0
A
1
A
2
Après boucle
```

IV – Problème typique : somme de 1 à n

La variable de boucle permet de faire la somme des entiers de 1 à 100 par exemple.

0 + 1 + 2 + 3 + 4 + 5 + 6 + ... + 99 + 100.

```
01 s = 0                          # Somme s = 0 au début
02 for x in range(101):           # Pour x variant de 0 à 100 !
03     s = s + x                  # Incrémente s de x
04 print(s)                       # Affiche la somme s
```

```
L2(x=0)-L3): s = 0 + 0 = 0
L2(x=1)-L3): s = 0 + 1 = 1
L2(x=2)-L3): s = 1 + 2 = 3
L2(x=3)-L3): s = 3 + 3 = 6
L2(x=4)-L3): s = 6 + 4 = 10
ect ...
```