Les boucles

8. Ce qu'il faut retenir

Une boucle sert à répéter une portion de code en fonction d'une condition (ou prédicat).

On peut parcourir une séquence grâce à la syntaxe : for element in sequence

On peut créer une boucle grâce au mot-clé while suivi d'une condition (ou prédicat)

La fonction range()

La fonction **range()** permet de créer une **liste**. Elle peut être utilisée avec 3 paramètres **range(valeur_debut, valeur_fin, pas).**

- **valeur_debut** : contient le numéro de départ de la liste,
- **valeur_fin :** le dernier nombre de la liste et finalement
- **pas :** l'incrément entre chaque nombre généré

Exemple :

```
range(10,40,5)
Out[9]: [10, 15, 20, 25, 30, 35]
```

Lorsqu'elle est omise, la valeur **pas** est implicitement égale à 1. Exemple range(10,40) est identique à range(10,40,1).
Si la **valeur_debut** n'est pas spécifiée, elle est implicitement mise à zéro. Exemple dans range(10) est identique à range(0,10).

Boucle for avec range()

Syntaxe :

```
for i in range(valeur_debut,valeur_fin,pas):
    instruction
```

- La variable **i** prendra successivement toutes les valeurs contenues dans (**in**) la liste crée par la fonction **range()**. On dit que cette variable est un **compteur** car, elle sert à numéroter les **itérations** de la boucle. On réalise une **itération** de la boucle à chaque fois que le corps de la boucle est exécuté.

- Les expressions **valeur_debut** *et* **valeur_fin** constituent les bornes de la boucle, d'où la notion de **boucle bornée**.
- La boucle inclut toujours ***valeur_debut*** et exclut ***valeur_fin*** pendant l'itération.
- Le nombre **d'itérations** dans une boucle = ***valeur_fin - valeur_debut***

Boucle For et chaîne de caractère

Il arrive très souvent qu'on traite des variables de chaîne de caractère, caractère par caractère, du premier jusqu'au dernier, pour effectuer à partir de chacun d'eux une opération quelconque. Nous appellerons cette opération un **parcours**.

Syntaxe :

```
for caracter in variable_chaine_de_caractere :
     Instruction
```

Boucle while

Synthaxe :

```
while condition:
     Instruction A
```

Le mot-clé **while** signifie **tant que** en anglais. Le corps de la boucle (c'est-à-dire le bloc d'instructions indentées) sera répété tant que la **condition** est vraie.

L'instruction break

Synthaxe :

```
Debut_boucle:
    instruction
    Si Condition
        break      # fin itération
    Instruction
instruction_apres_boucle
```

L'instruction **break** permet de « casser » l'exécution d'une boucle (**while** ou **for**). Elle fait sortir de la boucle et passer à l'instruction suivante.

L'instruction continue

Synthaxe :

```
Debut_boucle:
    instruction
    Si Condition
        continue  # passer à l'itération suivante
    Instruction
instruction_apres_boucle
```

L'instruction **continue** permet de passer prématurément au tour de boucle suivant. Elle fait continuer sur la prochaine itération de la boucle.

Comment choisir entre boucle for et boucle while ?

En général, si on connaît avant de démarrer la boucle le nombre d'itérations à exécuter, on choisit une boucle **for**. Au contraire, si la décision d'arrêter la boucle ne peut se faire que par un **test**, on choisit une boucle **while**.

Il est toujours possible de remplacer une boucle **for** par une boucle **while**.