TP N°3: Les structures itératives

Les *structures de contrôle itératives* ou les *boucles* permettent d'exéecuter plusieurs fois une ou plusieurs instructions. Python propose deux types de structures itératives : `for` et `while`.

1 La boucle `for ... in range(...):`

La boucle `for` est une boucle *inconditionnelle* sert à exécuter une ou plusieurs instructions un *nombre connu* de fois. En python, la syntaxe est la suivante :

```
for compteur in range(debut, fin, pas):
    bloc instructions
```

range retourne une séquence de nombres à partir de `debut` jusqu'à `fin` (exclu) : $[debut, fin[$

debut c'est la première valeur de l'intervalle crée par le fonction `range`, sert à initialiser la variable `compteur`. Par défaut : `debut = 0`

fin sert à donner une borne supérieure exclue de l'intervalle crée par le fonction `range` et dont la variable `compteur` ne doit pas y arriver.

pas C'est la valeur à ajouter à la variable `compteur` à chaque itération. Par défaut le `pas = 1`. On peut lui affecter une valeur négative, dans ce cas le parcours se fait dans le sens inverse, et par conséquant `debut` doit être supérieure à `fin`.

for sert à exécuter un bloc d'instructions un certain nombre de fois selon cette démarche :

1. Intialise la variable `compteur` par la valeur `debut`
2. Exécute son *corps* (le bloc d'instructions après le ":")
3. Incrémente la variable `compteur` par la valeur `pas`
4. Si la variable `compteur` est toujours strictement inférieur (ou strictement supérieur si le `pas` est négatif) à la valeur `fin` alors ré-itérer à partir du point 2, sinon la boucle s'arrête.

2 La boucle `while ... :`

La boucle `while` est une boucle *conditionnelle* sert à exécuter une ou plusieurs instructions un *nombre inconnu* de fois. La boucle `while` permet d'exécuter des instructions seulement sous une certaine condition. En Python ,la syntaxe est la suivante :

```
while condition:
    bloc instructions
```

condition Expression logique, aide la boucle `while` à décider si elle va exécuter son `corps` ou non.

while sert à exécuter un bloc d'instructions un certain nombre de fois selon cette démarche :

1. Evalue la `condition`
2. Si l'évaluation égale à `True`, alors elle exécute son *corps* (le bloc d'instructions après le ":") et ré-itérer à partir du point 1
3. Si l'évaluation égale à `False`, alors la boucle s'arrête.

3 Remarques

On peut utiliser les commandes suivantes avec les deux boucles `for` et `while` :

break On utilise l'instruction `break` lorsque l'on désire sortir d'une boucle itérative.
Lorsqu'on utilise `break` dans une boucle imbriquée, elle permet d'arrêter une seule boucle itérative à la fois (la boucle dont elle appartient à son corps).

continue On utilise l'instruction `continue` dans une boucle itérative lorsqu'on désire arrêter l'exécution de l'itération en cours et passer à l'itération suivante.

else On ajoute la clause `else` lorsqu'on désire exécuter un autre bloc d'inscturctions le cas où la boucle n'est pas interompue par `break`.

Exemple :

```
#Exemple 1
for i in  range (5):
    if i % 2 == 0:
        continue # ne pas afficher les nombres pairs
    print(i)

#Exemple 2
while True:
    n = eval(input("n = "))
    if n > 0:
        break # arrêter la boucle while
else: # ce bloc ne sera pas exécuté si la boucle while est arrêtée avec la commande break
    print(n)
```

4 Exercices

Exercice 1: La boucle `for`

1. Afficher tous les multiples de 7 inférieurs à 100.
2. Donner la somme des carrés de tous les entiers de 1 à 1000.
3. A l'aide d'une boucle `for` et de la commande `print(arg1,arg2,...)`, écrire un programme qui affiche sur 10 lignes la table des carrés des 10 premiers entiers non-nuls.
4. Écrire un programme qui calcule et affiche $\sum_{k=p}^{n^3} \frac{1}{k}$,où p et n sont deux entiers choisis par l'utilisateur.
5. On considère les deux suites récurrentes de *Mycielski* définies par : $m_1 = 2$; $m_k = 2m_{k-1} + 1$; $c_1 = 1$; $c_k = 3c_{k-1} + m_{k-1}$. Ecrire un script Python nommém *mycielski.py* , qui permet de saisir un entier n strictement positif , puis calcule et affiche le n^{eme} terme de la suite c_n

Exercice 2: La boucle `while`

1. Écrire un programme demandant un entier N, et qui affiche à l'écran tous les diviseurs de N. Combien 540 a-t-il de diviseurs ?
2. Écrire un programme demandant un entier naturel N (supérieur à 2) et qui affiche à l'écran son plus petit diviseur (supérieur à 2).

Exercice 3

On appelle *persistance* d'un nombre, le nombre d'itérations (répétitions) nécessaires pour le réduire à un seul chiffre, en multipliant tous ceux qui le composent et en recommençant avec le résultat obtenu.

Ainsi la persistance de 6788 est 6 car :
$6 \times 7 \times 8 \times 8 = 2688 \rightarrow 1^{ere} itration \mid 2 \times 6 \times 8 \times 8 = 768 \rightarrow 2^{eme} itration \mid 7 \times 6 \times 8 = 336 \rightarrow 3^{eme} itration$
$3 \times 3 \times 6 = 54 \rightarrow 4^{eme} itration \mid 5 \times 4 = 20 \rightarrow 5^{eme} itration \mid 2 \times 0 = 0 \rightarrow 6^{eme} itration$

Ecrire un script Python nommé *persistance.py* permettant de calculer et afficher la persistance d'un entier n strictement positif saisi au clavier.