Informatique - Notes de cours 3

Structures itératives - Boucles *for*

M. Marmorat

16 octobre 2024

Lorsqu'une opération doit être répétée plusieurs fois dans un programme, on utilise une boucle. Les boucles sont des structures de contrôle fondamentales en informatique. Elles offrent une grande richesse dans l'écriture de programmes. Elles sont divisées en deux types : boucles `for` et boucles `while`.

Nous abordons dans ce cours la syntaxe et les utilisations classiques des boucles `for`.

1 Syntaxe des boucles `for`

La boucle `for` permet de répéter un bloc d'instructions un certain nombre de fois, à condition que ce nombre de fois soit connu à l'avance.

Pour cela, on utilise un compteur qui va parcourir une liste des valeurs. La syntaxe est la suivante :

```
for compteur in liste_de_valeurs:
    #Instructions a repeter dans la boucle
#Instructions hors de la boucle
```

où `liste_de_valeurs` est un objet dont on peut parcourir les éléments. Nous verrons cette année plusieurs objets de ce type : listes, chaînes de caractères... On dit que ces éléments sont des itérables.

Voici un premier exemple d'utilisation d'une boucle `for`.

Exercice 1. Que fait le programme suivant ?

```
for n in [0, 1, 2, 3, 5, 10]:
    print(n)
print("C'est fini !")
```

Remarque 1.

- Dans le script ci-dessus, `[0, 1, 2, 3, 5, 10]` est une liste d'entiers, contenant les valeurs 0, 1, 2, 3, 5 et 10.
- On utilise ici la fonction `print` pour afficher dans la console Python la liste des valeurs parcourue par l'entier `n`. Attention à ne pas confondre les instructions `print` et `return` !

2 La fonction `range`

Pour utiliser l'instruction `for`, il est pratique de savoir créer des listes d'entiers. Pour ce faire, on utilise la fonction `range` :

- `range(a,b)` est la liste de tous les entiers de a à $b-1$ (inclus).

- `range(a,b,r)` est la liste de tous les entiers de a à $b-1$ (inclus), parcourus avec un pas de r. Autrement dit, il s'agit de la liste des termes de la suite arithmétique de premier terme a et de raison r, *jusqu'à la dernière valeur strictement plus petite que* b.
- `range(b)` est la liste de tous les entiers de 0 à $b-1$ (donc `range(b)` renvoie la même chose que `range(0, b)`).

Exercice 2. Que contiennent les listes suivantes ?

1. `range(2, 7)` :

2. `range(1, 20, 6)` :

3. `range(5)` :

Exercice 3. Ecrire un programme qui affiche tous les entiers pairs entre 0 et $2n$ (on proposera 3 solutions)

3 Sommes et produits

La boucle `for` permet de calculer des objets définis par récurrence, comme par exemple les sommes et les produits.

Ainsi, le script suivant

```
somme = 0
for n in range(10):
    somme += n
```

somme tous les entiers compris entre 0 et 9 et stocke cette valeur dans la variable `somme`.

Après exécution du script, la variable `somme` vaut donc .

Remarque 2.
- **Rappel :** on aurait aussi pu écrire

```
somme = 0
for n in range(10):
    somme = somme + n
```

pour obtenir le même résultat.

- Si l'on souhaite calculer un produit, on peut utiliser le même type de script, par exemple

```
produit = 1
for n in range(1,7):
    produit *= n
```

calcule le produit de tous les entiers compris entre 1 et 6 et stocke cette valeur dans la variable `produit`. Ainsi après exécution, la variable `produit` vaut .

Exercice 4. Écrire une fonction Python `somme1` qui prend en argument un entier n et qui renvoie

$$\sum_{k=1}^{n} k^3 = 1^3 + 2^3 + 3^3 + \cdots + n^3.$$

Exercice 5. Écrire une fonction Python `somme2` qui prend en argument un entier n et qui renvoie

$$\sum_{k=1}^{n} \cos\left(\frac{2k\pi}{n}\right) = \cos\left(\frac{2\times 1\times \pi}{n}\right) + \cos\left(\frac{2\times 2\times \pi}{n}\right) + \cdots + \cos\left(\frac{2\times n\times \pi}{n}\right)$$

Exercice 6. Écrire une fonction Python `factorielle` qui prend en argument un entier n et qui renvoie $n! = 1 \times 2 \times \cdots \times (n-1) \times n$.

4 Suites récurrentes

Une autre utilisation importante des boucles `for` est le calcul des termes d'une suite récurrente. Ainsi, après exécution du script suivant

```
u = 1
for n in range(100):
    u = 2*u+3
```

la variable `u` vaut le 100-ème terme de la suite (u_n) définie par $u_0 = 1$ et la relation de récurrence

$$\forall n \in \mathbb{N}, \quad u_{n+1} = 2u_n + 3.$$

Exercice 7. Écrire un script Python `suite1` qui calcule le 10-ème terme de la suite (u_n) définie par $u_0 = 0$ et

$$\forall n \in \mathbb{N}, \quad u_{n+1} = -2u_n + 3.$$

Exercice 8. Écrire une fonction Python `suite2` qui prend en argument un entier n et qui renvoie le n-ème terme de la suite (u_n) définie par $u_0 = 0$ et

$$\forall n \in \mathbb{N}, \quad u_{n+1} = u_n^2 + 3.$$

Exercice 9. Écrire une fonction Python `suite3` qui prend en argument un entier n et qui renvoie le n-ème terme de la suite (u_n) définie par $u_0 = 6$ et

$$\forall n \in \mathbb{N}, \quad u_{n+1} = u_n + n^2.$$

On peut aussi calculer les termes d'une suite récurrente double avec une boucle `for`. Par exemple, supposons que l'on souhaite calculer les termes de la suite récurrente linéaire double (u_n) définie par $u_0 = 1$, $u_1 = 2$ et

$$\forall n \in \mathbb{N}, \quad u_{n+2} = 2u_{n+1} - u_n.$$

Après exécution du script suivant,

```
u = 1
v = 2
for n in range(9):
    v, u = 2*v-u, v
```

la variable `v` vaut u_{10} et `u` vaut u_9. On utilise les deux variables `u` et `v` pour stocker u_n et u_{n+1}, et il y a 9 éléments dans la liste `range(9)` !

Exercice 10. Écrire une fonction Python `fibo` qui prend en argument un entier n et qui renvoie le n-ème terme de la suite de Fibonacci définie par (F_n) définie par $F_0 = 0$, $F_1 = 1$ et

$$\forall n \in \mathbb{N}, \quad F_{n+2} = F_{n+1} + F_n.$$

Exercice 11. Écrire une fonction Python `suite2` qui prend en argument un entier n et qui renvoie le n-ème terme de la suite (u_n) définie par $u_0 = 2$, $u_1 = 3$ et

$$\forall n \in \mathbb{N}, \quad u_{n+2} = \sqrt{u_n + u_{n+1}^2}.$$

5 Entraînement

Exercice 12. 1. Ecrire une fonction Python `somme3` qui prend en argument un entier n et qui renvoie $\displaystyle\sum_{k=1}^{n} \frac{1}{k}$.

2. Ecrire une fonction Python `somme4` qui prend en argument un entier n et qui renvoie $\sum_{k=1}^{n} \frac{1}{k+n}$.

3. Ecrire une fonction Python `produit1` qui prend en argument un entier n et un nombre réel x et qui renvoie $\prod_{k=1}^{n} x^k$ (on fera comme si l'on ne savait pas calculer ce produit à la main).