Listes Python

INFORMATIQUE COMMUNE - TP n° 3 - Olivier Reynet

À la fin de ce chapitre, je sais :

- ☞ créer une liste in extenso, avec une boucle ou en compréhension
- ☞ manipuler une liste pour ajouter, ôter ou sélectionner des éléments
- ☞ tester l'appartenance d'un élément à une liste
- ☞ utiliser la concaténation et le tronçonnage sur une liste
- ☞ itérer sur les éléments d'une liste avec une boucle for
- ☞ coder les algorithme incontournables (count, max, min, sum, avg)

Les listes Python constituent le couteau suisse du langage et permettent de modéliser une collection d'informations. Selon l'épreuve que vous passerez au concours, elles répresenteront des points GPS d'une trajectoire, l'orientation de spins dans un matériau ferromagnétique, la population d'un pays, la durée de vie d'un isotope, les nombres d'une conjecture, un jeu de dominos, un plateau de jeu de dames ou de solitaire...

A Jouons avec les listes

Toute utilisation d'une structure de données commence par une initialisation. C'est l'objet de cette section : initialiser correctement une liste [1].

A1. **Création et manipulation :**

(a) Créer **in extenso** [2] une liste contenant les 15 premiers entiers,

(b) Créer une liste contenant les 15 premiers entiers en utilisant une boucle `for`,

(c) Créer en compréhension une liste contenant les 15 premiers entiers,

(d) Sélectionner et afficher le cinquième et le neuvième élément,

(e) Tester l'appartenance des nombres 5 et 42 à la liste,

(f) Ajouter un élément à l'aide de la méthode append,

(g) Retirer le dernier élément,

(h) Afficher un élément sur deux à partir du deuxième,

(i) Concaténer la liste à elle même,

(j) Démultiplier la liste par cinq et afficher la longueur de la liste.

On suppose dans ce qui suit que n et h sont des variables telles que `h = 5` et `n = h * h`.

1. C'est aussi souvent le début des épreuves de concours...
2. C'est à dire en explicitant tous les éléments.

A2. Créer la liste des n premiers éléments pairs.

A3. Créer la liste des n premiers éléments impairs.

A4. Créer une liste de n éléments selon le modèle [1, 4, 7, 10, ...].

A5. Créer une liste de n éléments ne contenant que des zéros de deux manières différentes.

A6. Créer une liste de n éléments ne contenant alternativement que 1 et -1 de deux manières différentes. C'est à dire : [1, -1, 1, -1 ...].

A7. Créer une liste de `n = h * h` éléments constituée de l'alternance de h fois 1 et de h fois -1 de deux manières différentes. La liste commence par un 1. [1, 1, 1,...,-1, -1, -1,..., 1, 1, 1, ...]

A8. Créer une liste de liste de `h` éléments constituée de l'alternance d'une liste de h fois 1 et d'une liste de h fois -1 de deux manières différentes. [[1, 1, 1, ... 1], [-1, -1, -1, ...,-1], [1, 1, 1, ..., 1], ...]

A9. Créer un jeu de dominos. On représente un domino par un tuple (2,3), le zéro marque l'absence de numéro. Le jeu contenant toutes les pièces est une liste de tuple. Le jeu de dominos en contient que 28 éléments : `[(0, 0), (1, 0), (1, 1), (2, 0), (2, 1), (2, 2), (3, 0), (3, 1), (3, 2), (3, 3), (4, 0), (4, 1), (4, 2), (4, 3), (4, 4), (5, 0), (5, 1), (5, 2), (5, 3), (5, 4), (5, 5), (6, 0), (6, 1), (6, 2), (6, 3), (6, 4), (6, 5), (6, 6)]`

B Algorithmes incontournables

B1. Créer une liste L de dix nombres aléatoirement choisis dans l'intervalle $[0,1[$.

B2. Créer une fonction qui renvoie le nombre d'éléments strictement supérieurs à la valeur `v` dans une liste. Le prototype de cette fonction est `count_if_sup(L, v)`, où L est un type `list` et `v` un type `float`. Elle renvoie un type `int`.

B3. Créer une fonction qui renvoie l'élément maximal d'une liste s'il existe (liste non vide) et `None` sinon. Le prototype de cette fonction est `max_val(L)`, où L est un type `list`.

B4. Créer une fonction qui renvoie l'indice de l'élément maximal d'une liste s'il existe (liste non vide) et `None` sinon. Le prototype de cette fonction est `max_index(L)`, où L est un type `list`.

B5. Créer une fonction qui renvoie les indices des éléments minimal et maximal d'une liste sous la forme d'un tuple. Le prototype de cette fonction est `min_max_index(L)`, où L est un type `list`.

B6. Créer une fonction qui renvoie la moyenne des éléments d'une liste. Le prototype de cette fonction est `average(L)`, où L est un type `list`. Elle renvoie un type `float`.

C Syracuse

On considère la suite u définie par $u_0 \in \mathbb{N}^\star$ et :

$$\forall n \in \mathbb{N}, \quad u_{n+1} = \begin{cases} 3u_n + 1 & \text{si } u_n \text{ est impair} \\ \frac{u_n}{2} & \text{si } u_n \text{ est pair} \end{cases} \tag{1}$$

Compléter à la main :

n	0	1	2	3	4	5	6	7	8	9	10	11	12	13
u_n	3													

La conjecture de Syracuse fait l'hypothèse qu'il existe un rang N tel que $u_N = 1$, quel que soit l'entier u_0 choisi. Pour un entier u_0 donné, on nomme :

durée du vol l'entier défini par : $\min\{N \in \mathbb{N}^\star, u_N = 1\}$

altitude maximale du vol l'entier défini par : $\max\{u_n, n \in \mathbb{N}\}$

durée de vol en altitude l'entier défini par : $\max\{n \in \mathbb{N}, \forall k \in [\![0, n]\!], u_k \geqslant u_0\}$

C1. Que valent la durée du vol, l'altitude maximale et la durée de vol en altitude pour $u_0 = 3$?

C2. Écrire une fonction `suivant(u)` prenant en entrée un type `int` et renvoyant le terme suivant. Par exemple `suivant(8)` renverra 4 et `suivant(3)` renverra 10.

C3. En utilisant la fonction précédente, écrire une fonction `syracuse(u0)` prenant comme paramètres d'entrée le premier terme `u0` de la suite de type `int`. Cette fonction renvoie la liste `vol` contenant de tous les termes u_i jusqu'à la durée du vol N, c'est à dire $u_N = 1$.

C4. Comment peut-on calculer la durée du vol à partir de la fonction précédente ?

C5. Modifier la fonction `syracuse(u0)` afin qu'elle renvoie également la durée de vol en altitude et l'altitude maximale. La valeur renvoyée sera un tuple `(vol, vol_alt, alt_max)`.

C6. Compléter :

u_0	7	26	27	28	703
Durée du vol					
Altitude maximale					
Durée de vol en altitude					