Les listes en Python

Planche de TD pour un cours dispensé à l'université de Rennes 2

Romain Tavenard

Le but de cette séance est de commencer à manipuler les listes en Python. Il est fortement conseillé de tester les éléments de syntaxe présentés dans le cours avant de vous lancer dans ce TD. Comme d'habitude, un rappel : la documentation Python est de très bonne qualité, utilisez-la :

— pour les listes : https://docs.python.org/3/tutorial/datastructures.html ;

1 Échauffement

1. Écrivez, en pseudo-code, un algorithme calculant la différence minimale (en valeur absolue) entre éléments consécutifs d'une liste fournie en entrée. Par exemple, pour la liste `[1, 3, 5, 4]`, la différence retournée devra être `1` (la différence entre les deux derniers éléments de la liste).

2 Organisation de votre code

Pour ce TD, vous créerez un nouveau fichier `td3.py` dans le répertoire que vous avez créé à la première séance.

Dans ce nouveau nouveau fichier, votre code sera organisé de la manière suivante :

```
# Section 1 : les imports
[...]

# Section 2 : les définitions de fonctions
[...]

# Section 3 : les tests (un ou plusieurs par fonction codée)
[...]
```

Notamment, vous définirez vos fonctions en début de fichier et les appels seront listés en fin de fichier. De cette manière, vous pourrez, d'une question à l'autre, réutiliser les fonctions déjà codées au besoin. Si vous souhaitez, à un certain

moment, ne plus ré-exécuter vos tests de début de TD, il suffira de commenter les appels de fonction correspondants (mais pas les fonctions elles-mêmes !).

3 Les listes

3.1 Recherche d'élément dans une liste

2. Écrivez une fonction qui prend en entrée une liste et un élément (par exemple un entier) et retourne l'indice de la première occurrence de cet élément dans la liste (ou `None` si l'élément n'est pas présent dans la liste) sans faire appel à la méthode `index`.
3. Écrivez une fonction qui prend en entrée une liste et un élément (par exemple un entier) et retourne une version de la liste passée en argument dans laquelle toutes les occurrences de cet élément ont été supprimées.

3.2 Copie de liste

La copie de liste en Python est un exercice délicat. En effet, si l'on crée une première liste `liste1` puis que l'on écrit

```
liste2 = liste1
```

le contenu de `liste1` ne sera pas recopié dans `liste2`, les deux objets ne feront qu'un. Cela signifie notamment que si l'on modifie l'une des deux listes, la modification sera répercutée sur l'autre liste.

4. Pour vous en convaincre, écrivez une fonction nommée `copie_fausse` qui prend une liste en argument, la copie à l'aide d'une simple affectation du type `liste2 = liste1` et affiche le contenu des deux listes après modification de la liste passée en argument.

Ainsi, si l'on veut copier le contenu d'une liste dans une autre, on devra utiliser une astuce syntaxique. Deux approches sont possibles. La première utilise une liste en compréhension, la deuxième la fonction `list` qui effectue une copie de la liste passée en argument.

5. Écrire une fonction nommée `copie_comprehension` qui prend une liste en argument, la copie à l'aide d'une liste en compréhension et affiche le contenu des deux listes après modification de la liste passée en argument.
6. Écrire une fonction nommée `copie_cast` qui prend une liste en argument, la copie à l'aide de la fonction `list(.)` et affiche le contenu des deux listes après modification de la liste passée en argument.

4 Exercices de synthèse

7. Implémentez en Python la fonction de la question 1. **N.B.** : Pour calculer la valeur absolue d'une quantité numérique, on pourra utiliser la fonction `abs`.
8. Écrivez une fonction qui prend en entrée une liste `li` et un nombre `a` et retourne une liste ne contenant que les éléments de `li` supérieurs ou égaux à `a`.
9. **En utilisant la fonction de la question précédente**, écrivez une fonction qui prend en entrée une liste `li` et retourne une liste ne contenant que les éléments positifs de `li`.