Session d'exercices – Listes

Dans cette série d'exercices, vous devrez créer, manipuler, et modifier des listes python. Pour certains exercices, il vous faudra parcourir une liste tout en utilisant l'index d'un élément. Il existe plusieurs façons de faire cela. La première consiste à utiliser la fonction **len()**, qui retourne la taille d'une liste. Vous pouvez essayer le code suivant:

```python
"""
Example 1
"""
l = [1, 2, 3, 4]
i = 0
while i < len(l):
    print(f"l[{i}] = {l[i]}", end="; ")
    i += 1
print()
# Output:
# l[0] = 1; l[1] = 2; l[2] = 3; l[3] = 4;
```

Une autre solution consiste à utiliser la fonction **enumerate**, qui permet l'utilisation de boucles `for` et retourne à la fois l'index (i dans l'exemple) et la valeur dans la liste à cet index (v dans l'exemple).

```python
"""
Example 2
"""
l = [1.4, 2.0, -3.7, 4.9]
for i, v in enumerate(l):
    print(f"l[{i}] = {l[i]}", end="; ")
print()
# Output:
# l[0] = 1.4; l[1] = 2.0; l[2] = -3.7; l[3] = 4.9;
```

Deux autres fonctions qui vous seront utiles pour cet exercice sont **append** et **extend**. La première permet de concaténer une valeur (un element) à une liste, comme cela:

```python
"""
Example 3
"""
l = [False, 2, -9.999]
l.append(True)
print(l)
# Output:
# [False, 2, -9.999, True]
```

La seconde fonction permet de concaténer deux listes ensemble:

```python
"""
Example 4
"""
l1 = [1, 2, 3]
l2 = [4, 5, 6]
```

```
6     l1.extend(l2)
7     print(f"l1 = {l1}")
8     print(f"l2 = {l2}")
9     # Output:
10    # l1 = [1, 2, 3, 4, 5, 6]
11    # l2 = [4, 5, 6]
```

Enfin, python a une syntaxe, appelée «list comprehension», permettant de créer une liste en en parcourant une autre. Notez que la fonction **enumerate** peut aussi être utilisée dans le `for` ci-dessous (ligne 12).

```
1     """
2     Example 5
3     """
4     l1 = [1, 5, 9, 12]
5
6     l2 = [x + 1 for x in l1]
7     print(f"l2 = {l2}")
8
9     l3 = [x for x in l1 if x % 2 == 0]
10    print(f"l3 = {l3}")
11
12    l4 = [x for i, x in enumerate(l1) if i % 2 == 0]
13    print(f"l4 = {l4}")
14    # Output:
15    # l2 = [2, 6, 10, 13]
16    # l3 = [12]
17    # l4 = [1, 9]
```

Exercices

Dans les exercices suivants, il vous sera demandé de modifier une liste existante ou d'en créer une nouvelle. N'oubliez pas qu'une liste doit d'abord être créée avant de pouvoir être remplie ou modifiée. Dans de nombreux cas, il suffit de créer une liste vide. Parfois, il est également nécessaire de l'initialiser (attribuer des valeurs aux éléments de la liste).

1. (a) [Difficulté: *] Créez et remplissez (initialisez) une liste `l` de n éléments: `l = [e0, e1, e2, ..., en]` (ici, `e0` doit être remplacé par la valeur de tout premier élément de la liste, etc.). Ensuite, écrivez le code permettant de créer une seconde liste, `l_swap_adjacent`, où les éléments adjacents de `l` sont échangés: `l_swap_adjacent = [e1, e0, e3, e2, ...]`.
   Par exemple, la liste `l = [1, 2.5, 11, 4, 0.5]` doit donner `l_swap_adjacent = [2.5, 1, 4, 11, 0.5]`.
   Avant de coder, élaborez l'algorithme et testez-le sur papier pour vérifier qu'il fonctionne.

   (b) [Difficulté: **] Écrivez le code permettant de faire la même chose, mais cette fois sans créer une seconde liste. Vous devrez directement modifier la liste `l`.
   Avant de coder, élaborez l'algorithme et testez-le sur papier pour vérifier qu'il fonctionne.

2. [Difficulté: *] Soit une liste `l` de n éléments: `l = [e0, e1, e2, ..., en]`. Écrivez le code permettant de créer une nouvelle liste avec les mêmes éléments mais dans un ordre différent: `l_shuffle = [e0, en, e1, e(n-1), ...]`.

   Par exemple, une liste `l = [1, 2, 33, 24, 15]` donnera `l_shuffle = [1, 15, 2, 24, 33]`.

   Avant de coder, élaborez l'algorithme et testez-le sur papier pour vérifier qu'il fonctionne.

3. (a) [Difficulté : **] Soit une liste `l` de n éléments: `l = [e0, e1, e2, ..., en]`. Écrivez le code permettant de créer une seconde liste `l_even_index_first = [e0, e2, ..., e1, e3,...]` telle que tous les éléments aux index pairs apparaissent avant les éléments aux index impairs. Ici, ous devrez utiliser des boucles `while`.

   Par exemple, pour la liste `l = [1, 12, 23, 34, 55]`, le résultat sera `l_even_index_first = [1, 23, 55, 12, 34]`.

   Avant de coder, élaborez l'algorithme et testez-le sur papier pour vérifier qu'il fonctionne.

   (b) [Difficulté : *] Même question, mais cette fois vous devez utiliser la fonction `enumerate`.

   (c) [Difficulté : *] Pouvez-vous résoudre le même exercice en peux lignes de code en utilisant une «list comprehension» et la fonction `extend` ?

4. Soit une liste d'entiers `l = [1, 7, 3, 2, 4]`.

   (a) [Difficulté : *] Calculez la moyenne et la variance des nombres dans cette liste. Il n'est pas permis de faire appel aux fonctions python `statistics.mean` et `statistics.variance`.

   Pour la liste `l = [1, 7, 3, 2, 4]`, la moyenne est 3.4 et la variance est 5.3.

   Avant de coder, élaborez l'algorithme et testez-le sur papier pour vérifier qu'il fonctionne.

   (b) [Difficulté : *] Trouvez le maximum et le minimum dans cette liste. Il n'est pas permis de faire appel aux fonctions python `max` et `min`.

   Pour la liste `l = [1, 7, 3, 2, 4]`, le maximum est 7 et le minimum 1.

   Avant de coder, élaborez l'algorithme et testez-le sur papier pour vérifier qu'il fonctionne.