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:

```
    """
    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).

```
    """
    Example 2
    """
    l = [1.4, 2.0, -3.7, 4.9]
    for i, v in enumerate(l):
        print(f"l[{i}] = {v}", 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:

```
    """
    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:

```
    """
    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]
```

Python offre de multiples méthodes sur les listes, dont voici quelques exemples du cours en application. La méthode **copy** permet de copier une liste, puis de modifier la copie sans que la liste de base ne soit modifiée. La méthode **remove(x)** supprime la première occurence de x dans la liste, alors que **pop([i])** supprime l'élément à l'index i, ou si i n'est pas précisé le dernier élément de la liste. Les accolades autour de l'argument i montrent que i est un argument optionnel, c'est une notation souvent utilisée dans la documentation python.

```
1   """
2   Example 6
3   """
4   l1 = [1, 2, 3, 4, 5, 6]
5
6   l2 = l1.copy()
7   l2.reverse()
8   print(f"l1 = {l1}")
9   print(f"l2 = {l2}")
10
11  l3 = l1.copy()
12  l3.remove(2)
13  print(f"l3 = {l3}")
14
15  l4 = l1.copy()
16  l4.pop(3)
```

```python
print(f"l4 = {l4}")

l5 = l1.copy()
l5.pop()
print(f"l5 = {l5}")

# Output:
# l1 = [1, 2, 3, 4, 5, 6]
# l2 = [6, 5, 4, 3, 2, 1]
# l3 = [1, 3, 4, 5, 6]
# l4 = [1, 2, 3, 5, 6]
# l5 = [1, 2, 3, 4, 5]
```

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 $l = [e_0, e_1, e_2, ..., e_{n-1}]$ de n éléments (ici, e_0 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: $l\_swap\_adjacent = [e_1, e_0, e_3, e_2, ...]$.

   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: $l = [e_0, e_1, e_2, ..., e_{n-1}]$. Écrivez le code permettant de créer une nouvelle liste avec les mêmes éléments mais dans un ordre différent: $l\_swap\_symmetric = [e_0, e_{n-1}, e_1, e_{n-2}, ...]$. C'est à dire que la nouvelle liste doit avoir le premier élément de la liste originale, puis le dernier, puis le deuxième, puis l'avant-dernier, et ainsi de suite.

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

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

3. [Difficulté : **] Soit une liste l de n éléments: $l = [e_0, e_1, e_2, ..., e_{n-1}]$. Écrivez le code permettant de créer une seconde liste $l\_even\_index\_first = [e_0, e_2, ..., e_1, e_3, ...]$ telle que tous les éléments aux index pairs apparaissent avant les éléments aux index impairs.

   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.

   (a) Utiliser des boucles `while`.

   (b) Même question, mais cette fois vous devez utiliser la fonction `enumerate()`.

   (c) Pouvez-vous résoudre le même exercice en deux 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 de l'échantillon ajustée des nombres dans cette liste. L'expression pour caluler la variance de l'échantillon ajustée :

   $$s_n^2 = \frac{1}{n-1} \sum_{i=0}^{n-1} (X_i - \overline{X}_n)^2,$$

   où $\overline{X}_n$ est la moyenne. Il n'est pas permis de faire appel aux fonctions python telles que:

   - `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.