Session d'exercices : Fonctions

En informatique, la structure des données joue un rôle crucial dans l'écriture de programmes efficaces. En Python, une liste normale n'est pas forcément triée, pourtant, avoir une liste triée a de nombreux avantages, notamment, pour la recherche d'éléments à l'intérieur de celle-ci.

Différents algorithmes de tri et de recherche d'un élément dans une liste vous ont été présentés lors du cours théorique d'ICC. Le but de cet exercice est de vous faire coder en Python l'algorithme de tri par sélection.

Tri par sélection [Difficulté: **]

Le tri par sélection, ou *Selection Sort*, est un algorithme de tri par comparaison. Il est facile à comprendre mais pas des plus efficaces, sa complexité étant de $O(n^2)$.

Il consiste à parcourir la liste dans son ensemble et à trouver le plus petit élément. Cet élément est ensuite déplacé à la position 0, et l'élément à la position 0 est déplacé à la position qu'occupait précédemment le plus petit élément. Ensuite, on cherche le deuxième plus petit élément, donc le plus petit élément à partir de la position 1, et on échange sa position avec celle de l'élément à la position 1. Et ainsi de suite pour les positions 3, 4, ..., $n-2$. Le dernier élément, à la position $n-1$, se retrouve automatiquement à être le plus grand.

Faisons ceci étape par étape. Pour commencer, entrez la ligne de code suivante afin d'importer les outils nécessaires à l'exécution du programme :

```
from random import randint
```

La fonction importée sert à générer des nombres aléatoires. En effet, l'expression `randint(x, y)` retourne un entier aléatoire entre x et y (compris).

1. Construisez une liste `numbers` de 20 nombres entiers aléatoirement choisis entre 1 et 100.

2. Définissez et codez la fonction `find_min_index()` qui prend comme paramètres une liste d'entiers `l` et un entier `index_start` et qui retourne l'**index** de l'élément le plus petit de la liste, en commençant la recherche non pas forcément à 0, mais à l'index `index_start`. Commencez par :

   ```
   def find_min_index(l, index_start) :
   ```

3. Modifiez légèrement la déclaration de votre fonction de manière à ce que, lors d'un appel de fonction, l'indication d'une valeur pour `index_start` soit optionnelle et que la valeur par défaut 0 soit utilisée.

   Utilisez les lignes de code suivantes pour tester votre fonction :

   ```
   print(find_min_index([3, 1, 4, 1, 18])) # sortie attendue : 1
   print(find_min_index([3, 1, 4, 1, 18], 0)) # sortie attendue : 1
   print(find_min_index([3, 1, 4, 1, 18], index_start=2)) # sortie attendue : 3
   print(find_min_index([3, 1, 4, 1, 18], index_start=4)) # sortie attendue : 4
   ```

4. Définissez et coder la fonction `selection_sort()` qui doit directement trier la liste d'entiers `l` passée en paramètre selon le principe du tri par sélection. Commencez par :

```
def selection_sort(l) :
```

   Utilisez des appels répétés à `find_min_index()` et des déplacements d'éléments.

Enfin, testez votre fonction en exécutant les commandes suivantes dans l'interpréteur. Votre liste `numbers` va s'afficher deux fois à l'écran : d'abord sous sa forme aléatoire et ensuite triée.

```
print(numbers)
selection_sort(numbers)
print(numbers)
```