Activité : algorithmes de tris par sélection et par insertion

1 Tri par sélection

1.1 Présentation de l'algorithme

1.2 Description de l'algorithme

1. **Dans la vidéo, quel nom, très simple, est donné au tableau (ou à la liste) ? Regarder au dessus des danseurs.**
   Le nom est a. En effet, les danseurs représentent les éléments du tableau, repérés par leurs indices : $a[0], a[1], a[2], \ldots$

2. **Décrire cet algorithme avec des phrases, en détaillant la suite des instructions des différentes étapes.**
   - On parcourt toute la liste pour trouver le plus petit élément. On le place en première position. *Ce premier élément est trié.*
   - On cherche le plus petit élément parmi les données restantes (du deuxième au dernier élément) et on le place à la deuxième position. *Les deux premiers éléments sont triés.*
   - On recommence en cherchant à chaque fois le plus petit élément parmi les données restantes, et on le place au début de ces données.

1.3 Programmation

3. **Programmer en Python une fonction *tri_selection* qui prend en argument une liste non triée et renvoie la liste triée.**

```
def tri_selection(liste):
    for i in range(len(liste) - 1):
        i_mini = i  # indice initial du minimum
        mini = liste[i] # valeur initiale du minimum
        for j in range(i + 1, len(liste)):
            if liste[j] < mini:
                i_mini = j # nouvel indice temporaire du minimum
                mini = liste[j] # nouvelle valeur temporaire du minimum
        liste[i], liste[i_mini] = liste[i_mini], liste[i]
```

4. **Tester votre fonction en l'appliquant sur les listes suivantes :**
   On peut utiliser le code suivant :

```
liste1 = [4, 2, 8, 10, 6]
liste2 = [10, 8, 6, 4, 2]
liste3 = []
liste4 = [5, 5, 4, 3, 3]

tri_selection(liste1)
tri_selection(liste2)
tri_selection(liste3)
tri_selection(liste4)
print(liste1, liste2, liste3, liste4)
```

2 Tri par insertion

2.1 Présentation de l'algorithme

2.2 Description de l'algorithme

5. Dans la vidéo, quel nom, très simple, est donné au tableau (ou à la liste) ? Regarder au dessus des danseurs.
Le nom est a. En effet, les danseurs représentent les éléments du tableau, repérés par leurs indices : $a[0], a[1], a[2], \ldots$

6. Décrire cet algorithme avec des phrases, en détaillant la suite des instructions des différentes étapes.

- On commence par trier la liste à partir de la gauche. *Au départ, l'élément le plus à gauche est trié.*
- Le premier élément non trié est appelé la clef. Ici c'est le deuxième élément. On insère alors la clef à la bonne place (à gauche ou à droite du premier élément selon sa valeur). *Les deux premiers éléments sont alors triés.*
- La clef est maintenant le troisième élément. On insère alors la clef à la bonne place (tout à gauche, entre les deux premiers éléments, ou à droite des deux premiers éléments, selon sa valeur). *Les trois premiers éléments sont alors triés.*
- On continue ainsi de suite. La partie de gauche de la liste est triée. La clef est toujours le premier élément non trié, elle doit être insérée à la bonne place.

2.3 Programmation

7. Programmer en Python une fonction *tri_insertion* qui prend en argument une liste non triée et renvoie la liste triée.

```
def tri_insertion(liste):
    for i in range(1, len(liste)):  # on parcourt la liste du deuxième élément jusqu'au dernier
        k = i    # indice initial de la clef
        clef = liste[i]    # valeur à insérer
        while k > 0 and clef < liste[k - 1]:
            liste[k] = liste[k - 1]    # on décale la valeur vers la droite
            k -= 1   # on diminue de 1 l'indice temporaire de la clef i.e. on la décale d'un cran vers la gauche
        liste[k] = clef   # on sort de la boucle while lorsqu'on a trouvé la bonne position pour la clef
```

8. Tester votre fonction en l'appliquant sur les listes suivantes :
On peut utiliser le code suivant :

```
liste1 = [4, 2, 8, 10, 6]
liste2 = [10, 8, 6, 4, 2]
liste3 = []
liste4 = [5, 5, 4, 3, 3]

tri_insertion(liste1)
tri_insertion(liste2)
tri_insertion(liste3)
tri_insertion(liste4)
print(liste1, liste2, liste3, liste4)
```