

Exercices 2

Philipp Ahrendt

November 28, 2025

Contents

1	Exercice 1: Recherche de motif	1
1.1	Question a:	1
1.2	Question b:	2
2	Exercice 2: Opérations de base sur les listes	2
2.1	Question a:	2
2.2	Question b:	2
3	Exercice 3: Algorithmes de tri	2
3.1	Question a: Tri par insertion	2
3.2	Question b: Tri par sélection	3
4	Exercice 4: Points au QCM	3

1 Exercice 1: Recherche de motif

1.1 Question a:

Écrire une fonction `motif` qui prend en entrée deux chaînes de caractères, et teste si la première apparaît dans la deuxième.

Exemples:

- `motif("gori", "algorithme") = True`
- `motif("agor", "algorithme") = False`

1.2 Question b:

Écrire une fonction `motif_occ` qui prend en entrée deux chaînes de caractères et renvoie le nombre de fois que la première chaîne apparaît dans la deuxième. Deux apparitions distinctes peuvent partager des caractères.

Exemple:

- `motif_occ("aba", "ababa") = 2`

2 Exercice 2: Opérations de base sur les listes

2.1 Question a:

Écrire des fonctions `l_max` et `l_min`, qui prennent en entrée une liste de nombres et renvoient, respectivement son plus grand et plus petit élément. Ces fonctions ne doivent pas utiliser les fonctions `max` et `min` de python.

2.2 Question b:

Écrire une fonction `moy` qui prend en entrée une liste de nombres et renvoie la moyenne de ses éléments.

3 Exercice 3: Algorithmes de tri

Dans cet exercice, nous allons implémenter différents algorithmes pour trier une liste de nombres, de l'élément le plus petit au plus grand.

3.1 Question a: Tri par insertion

Dans le tri par insertion, l'idée est de parcourir de parcourir la liste en la triant progressivement. Pour une liste de longueur n , à l'étape i , on cherche la place du i -ième élément dans la partie entre 0 et i . Pour $i = 0$, cela revient à ne rien faire; on ne peut pas déplacer en arrière l'élément en position 0. Pour $i = 1$, il faut comparer `L[1]` avec `L[0]` et les échanger si `L[0] > L[1]`. En continuant cette procédure sur $0 \leq i < n$, on assure qu'à la i -ième étape, les éléments $0 \leq j < i$ sont déjà triés, et qu'il suffit d'insérer le i -ième élément à la bonne place pour que les éléments $0 \leq j < i+1$ soient triés. Après l'étape $i = n-1$, toute la liste est triée.

Écrire une fonction `tri_insertion`, qui prend en entrée une liste, et la trie en faisant le tri par insertion.

3.2 Question b: Tri par sélection

Dans le tri par sélection, on cherche successivement les éléments croissants de la liste, en commençant par le plus petit, et les mets dans le bon ordre au début de la liste. A l'étape i , on cherche le plus petit élément parmi les éléments $i \leq j < n$. Quand on a trouvé cet élément en position j_m , on le déplace vers i , et on décale les autres éléments entre i et j_m vers l'avant. Quand on arrive à la fin, la liste est triée.

Écrire une fonction `tri_selection`, qui prend en entrée une liste, et la trie en faisant le tri par sélection.

4 Exercice 4: Points au QCM

On souhaite créer une liste des points obtenus par chaque étudiant lors d'un QCM. Le QCM contient 10 questions avec des réponses a,b,c,d. Les étudiants peuvent ne pas répondre à une question. Les résultats sont sauvegardés sous forme d'un dictionnaire `reponses_etudiants` avec une entrée pour chaque étudiant contenant ses réponses sous forme d'un dictionnaire. Si l'étudiant n'a pas répondu à une question, celle-ci ne figure pas dans son dictionnaire. On a aussi sauvegardé le corrigé dans un dictionnaire `corrige`. Voici un exemple avec une classe de 5 étudiants.

```
etudiants = ['Marc', 'Julien', 'Alice', 'Grégoire', 'Caroline']
```

```
corrige = {
    'Q1': 'd',
    'Q2': 'b',
    'Q3': 'c',
    'Q4': 'd',
    'Q5': 'c',
    'Q6': 'b',
    'Q7': 'b',
    'Q8': 'c',
    'Q9': 'd',
    'Q10': 'd',
}
```

```
reponses_etudiants = {
    'Marc': {'Q1': 'a', 'Q2': 'b', 'Q3': 'c', 'Q4': 'c', 'Q6': 'b', 'Q7': 'b', 'Q8': 'c', 'Q9': 'd', 'Q10': 'd'},
    'Julien': {'Q1': 'd', 'Q2': 'a', 'Q3': 'd', 'Q5': 'c', 'Q6': 'b', 'Q7': 'c', 'Q8': 'c', 'Q9': 'd', 'Q10': 'd'},
    'Alice': {'Q1': 'a', 'Q2': 'b', 'Q3': 'c', 'Q4': 'c', 'Q6': 'b', 'Q7': 'b', 'Q8': 'c', 'Q9': 'd', 'Q10': 'd'},
    'Grégoire': {'Q1': 'a', 'Q2': 'b', 'Q3': 'c', 'Q4': 'c', 'Q6': 'b', 'Q7': 'b', 'Q8': 'c', 'Q9': 'd', 'Q10': 'd'},
    'Caroline': {'Q1': 'a', 'Q2': 'b', 'Q3': 'c', 'Q4': 'c', 'Q6': 'b', 'Q7': 'b', 'Q8': 'c', 'Q9': 'd', 'Q10': 'd'},
}
```

```
'Alice': {'Q1': 'b', 'Q2': 'b', 'Q3': 'b', 'Q4': 'd', 'Q5': 'a', 'Q6': 'a', 'Q7':  
'Grégoire': {'Q2': 'b', 'Q3': 'c', 'Q4': 'd', 'Q5': 'c', 'Q6': 'b', 'Q7': 'b', 'Q8  
'Caroline': {'Q1': 'd', 'Q2': 'd', 'Q3': 'c', 'Q4': 'a', 'Q5': 'c', 'Q6': 'b', 'Q7  
}
```

Les points sont attribués de la manière suivante:

- Chaque réponse correcte donne 3 points
- Chaque réponse fausse donne -1 points
- Une question sans réponse donne 0 points

Écrire une fonction `correction_QCM` prenant en entrée un dictionnaire de réponses étudiants et le corrigé, et renvoie en sortie un dictionnaire avec une entrée pour chaque étudiant contenant son résultat au QCM.