Recherche séquentielle dans une liste

On désigne par le terme *recherche séquentielle* l'ensemble des méthodes de recherche qui consistent à énumérer un à un tous les éléments d'une liste (ce qui nécessite donc un *parcours* complet de la liste). D'autres méthodes plus performantes et plus complexes existent, elles seront étudiées ultérieurement.

Toutes les fonctions de ce TP doivent être sauvegardées dans un script `seq_search`. Toutes les fonctions doivent être documentées.

1 Recherche d'un élément

Exercice 1. Écrire une fonction `search(li, n)` qui retourne l'indice de la valeur `n` si celle-ci est présente dans la liste `li` d'entiers, et `None` si elle ne l'est pas.

Exemple. `search([3, 2, 6, 1, 7], 1)` doit retourner `3`, tandis que `search([3, 2, 6, 1, 7], 4)` doit retourner `None`.

On adopte dans la suite le modèle suivant :

- l'affectation d'une variable, un affichage, une opération arithmétique ou un test de comparaison coûtent une unité de calcul.
- une boucle qui fait n itérations coûte n fois le coût d'une itération.
- On va s'intéresser à la forme mathématique du coût (la famille de fonction de n associée) plus qu'aux détails de son calcul.

  Exemple. la fonction ci-dessous a un coût **proportionnel à n** (taille de la liste `li`) :

```
def affiche(li):
    for i in range(len(li)):
        print(li[i])
```

Exercice 2. On suppose que la liste contient n éléments. Quel est le coût de la fonction précédente, exprimé comme une fonction de n ?

2 Recherche du maximum (ou du minimum)

Exercice 3. Écrire une fonction `maxi(li)` qui retourne le maximum de la liste d'entiers `li`. Écrire de même une fonction `mini`.

Exemple. `maxi([3, 2, 6, 7, 1)` doit retourner `7`.

Exercice 4. On suppose que la liste contient n éléments. Quel est le coût de la fonction précédente ?

3 Recherche de l'indice du maximum (ou du minimum)

Exercice 5. Écrire une fonction `imax(li)` qui retourne l'indice du maximum de la liste `li`. Écrire de même une fonction `imin`.

Exemple. `imax([3, 2, 6, 7, 1])` doit retourner `3`.

Exercice 6. On suppose que la liste contient n éléments. Quel est le coût de la fonction précédente ?

4 Comptage des éléments à l'aide d'un dictionnaire

Lire le chapitre du cours sur les dictionnaires.

Remarque. On peut tester l'appartenance d'une valeur `val` à la liste des clés d'un dictionnaire `d` avec le mot-clé `in` :

```
>>> d = {2:1, 1:4}
>>> 2 in d
True
>>> 3 in d
False
```

Exercice 7. Écrire une fonction `dans(val, li)->bool:` qui retourne `True` si `val` est dans `li` et `False` sinon, comme le ferait le mot-clé `in`. La fonction est en fait valide pour toutes les séquences (listes, tuples et chaînes de caractères), la tester sur une liste puis sur un dictionnaire (qui est alors considéré comme la liste de ses clés).

Exercice 8. Écrire une fonction `count(li)` qui retourne un dictionnaire qui contient les valeurs de la liste et leur nombre d'occurrences. Utiliser au choix le mot-clé `in` ou la fonction `dans` écrite précédemment.

Exemple. `count([2, 1, 3, 1, 1, 2, 3, 1])` doit retourner `{2:2, 1:4, 3:2}`

Exercice 9. Quel est le coût de la fonction précédente ? Justifier soigneusement la réponse.

5 Recherche du second maximum

Exercice 10. Écrire une fonction `maxs(li)` qui retourne le premier et le second maximum et qui utilise la fonction `maxi` écrite précédemment.

Exemple. `maxs([3, 2, 5, 7, 1, 9])` doit retourner `9, 7`.

Exercice 11. Quel est le coût de la fonction précédente ? Écrire une nouvelle version de la fonction `maxs` qui ne nécessite qu'un parcours de la liste.