❑ **Exercice 1** : *Appartient*

Ecrire une fonction `appartient` qui prend en paramètre un entier `n` et une liste d'entiers `entiers` et renvoie `True` si `n` apparaît dans `entiers` et `False` sinon.

```
>>> appartient(10,[4,17,11])
False
>>> appartient(10,[])
False
>>> appartient(2,[8,5,1,1,1,2,7])
True
>>> appartient(0,[0,1,1,0,0,0,1])
True
>>> appartient(2,[0,1,1,0,0,0,1])
False
```

❑ **Exercice 2** : *Palindrome*

Une chaine de caractères est un *palindrome* lorsqu'elle peut être lue indifféremment de gauche à droite ou de droite à gauche, par exemple « KAYAK »ou « ANNA »sont des palindromes. Ecrire une fonction `est_palindrome` qui prend en argument une chaîne de caractères et renvoie `True` si cette chaîne est un palindrome et `False` sinon.

Exemples :

```
>>> est_palindrome("toto")
False
>>> est_palindrome("ressasser")
True
>>> est_palindrome("")
True
>>> est_palindrome("radar")
True
>>> est_palindrome("p")
True
>>> est_palindrome("retirer")
False
```

❑ **Exercice 3** : *Bonus*

1. Si vous avez proposé une solution récursive à l'exercice 2 alors donner une solution itérative et inversement.
2. Pour l'exercice 1, lorsqu'un la liste `entiers` est triée, un algorithme plus rapide existe pour déterminer si un entier `n` y figure ou non.
   - **a)** Rappeler le nom de cet algorithme et expliquer rapidement son fonctionnement.
   - **b)** Expliquer pourquoi cet algorithme est *plus rapide*.
     - ✪ On pourra indiquer simplement leur complexité en temps.
   - **c)** Ecrire une implémentation en Python de cet algorithme