❑ **Exercice 1** : *Somme des éléments d'une liste*

1. Ecrire une fonction `somme_iteratif` qui prend en paramètre une liste d'entiers et renvoie la somme des éléments de cette liste en faisant un parcours de liste. Si la liste est vide, cette fonction renvoie 0. Voici quelques exemples d'appel à cette fonction dans une console Python :

```
>>> somme_iteratif([4,17,11])
32
>>> somme_iteratif([])
0
>>> somme_iteratif([8,5,1,1,1,2])
18
>>> somme_iteratif([10])
10
```

2. Ecrire une fonction `somme_recursif` qui prend en paramètre une liste d'entiers et renvoie la somme des éléments de cette liste. Cette fonction doit s'appeler elle-même. De même que pour `somme_iteratif` si la liste est vide, on renvoie 0.
   Aide : On pourra recopier et utiliser la fonction suivante qui renvoie la "*queue*" d'une liste, c'est à dire cette liste privée de son premier élément. Par exemple `queue[4,17,11]` renvoie `[17,11]`.

```
    def queue(liste):
            return liste[1:]
```

3. **Bonus :** La fonction `queue` donnée ci-dessus peut s'écrire sans utiliser la notation en tranche comme dans `liste[1:]`. Donner une façon d'écrire la fonction `queue` sans utiliser cette notation.

❑ **Exercice 2** : *Partitions d'un entier*

Attention, exercice **difficile**, donné uniquement en bonus.

L'entier 4 peut s'écrire de diverses façons comme somme d'entiers :

- $4 = 4$
- $4 = 3 + 1$
- $4 = 2 + 2$
- $4 = 2 + 1 + 1$
- $4 = 1 + 1 + 1 + 1$

Ce sont les **partitions** de l'entier 4.

Ecrire une fonction `partitions` qui prend en argument un entier `n` et renvoie la liste de ses partitions. Par exemple :

```
>>> partitions(3)
>>> [[3],[2,1],[1,1,1]]
```