Sujet 12

❑ **Exercice 1** : *Consécutifs égaux*

Ecrire une fonction `consecutifs_egaux` qui prend en argument une liste et renvoie `True` si cette liste contient deux éléments consécutifs égaux et `False` sinon.

Exemples :

```
>>> consecutifs_egaux([2, 7, 5, 5, 11, 3])
True
>>> consecutifs_egaux([13])
False
>>> consecutifs_egaux([2, 3, 5, 7, 11, 13, 17])
False
>>> consecutifs_egaux([2, 1, 2, 1, 2, 1])
False
```

❑ **Exercice 2** : *Inversion des éléments situés au sommet d'une pile*

Dans cet exercice, on dispose de la classe `Pile` suivante qui implémente les méthodes de base sur la structure de pile : création d'une pile, test si la pile est vide, empile un élément et depile. De plus une pile dispose d'un attribut `longueur` qui donne son nombre d'éléments. Toutes les opérations sur les piles se feront par l'intermédiaire des méthodes de cette classe, il est donc *interdit* de modifier l'attribut valeurs d'une pile en dehors de la classe.

Compléter la fonction `inverse_au_sommet` qui prend en paramètre une pile et inverse les deux éléments situés au sommet de cette pile lorsque cette pile contient au moins deux éléments, sinon la fonction ne fait rien. Par exemple, si les valeurs contenus dans la pile sont `["N","S","I"]` après appel de `inverse_au_sommet` sur cette pile, elle devient `["N","I","S"]`

```
    class Pile:
        def __init__(self):
            self.contenu = []
            self.longueur = 0

        def get_longueur(self):
            return self.longueur

        def est_vide(self):
            return self.longueur == 0

        def empiler(self, v):
            self.contenu.append(v)
            self.longueur += 1

        def depiler(self):
            if not self.est_vide():
                self.longueur -= 1
                return self.contenu.pop()

    def inverse_au_sommet(pile):
        if pile.get_longueur() >= ...:
            sommet = pile.......
            sous_sommet = ......
            pile.........(.....)
            pile.........(.....)
```