Exercice 1

Ecrire une fonction `max_et_indice` qui prend en paramètre un tableau non vide `tab` de nombres entiers et qui renvoie la valeur du plus grand élément de ce tableau ainsi que l'indice de sa première apparition. **Sans utiliser la fonction max**

```
>>> max_et_indice([1, 5, 6, 9, 1, 2, 3, 7, 9, 8])
(9,3)
>>> max_et_indice([-2])
(-2, 0)
>>> max_et_indice([-1, -1, 3, 3, 3])
(3, 2)
>>> max_et_indice([])
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
  File "C:\Users\vanot\Desktop\Watteau.py", line 2, in max_et_indice
    assert tab, "Le tableau ne doit pas etre vide"
AssertionError: Le tableau ne doit pas etre vide
```

Exercice 2

Ecrire une fonction `renverse`, prenant en paramètre une chaine de caractères `mot` et renvoyant cette chaine en ordre inverse.

```
>>> renverse("")
""
>>> renverse("abc")
"cba"
>>> renverse("informatique")
"euqitamrofni"
```

Exercice 3

Je veux développer un jeu de morpion grâce à Python, pour ce faire j'ai déjà besoin d'un plateau de jeu.

Le plateau d'un morpion est un tableau de 3 cases par 3 cases

En une seule ligne, créer une variable `tableau` et lui affecté un tableau rempli de `'0'` de taille 3x3 sans écrire soit même la liste comme suit.

```
tab = [[0, 0, 0], [0, 0, 0], [0, 0, 0]]
```

Exercice 4

Ecrire une fonction `ecriture_binaire_entier_positif` qui prend en paramètre un entier positif `n` et renvoie une chaine de caractères correspondant à son écriture binaire.

On rappelle que:

- l'écriture binaire de 25 est 11001 car
  25 = 1 * 2**4 + 1 * 2**3 + 0 * 2**2 + 0 * 2**1 + 1 * 2**0
- `n%2` vaut 1 si `n` est impair, 0 sinon
- `n//2` donne le quotient de la division euclidienne de `n` par 2.

```
>>> ecriture_binaire_entier_positif(25)
'11001'
>>> ecriture_binaire_entier_positif(8)
'1000'
>>> ecriture_binaire_entier_positif(255)
'11111111'
```

Exercice 5

On considère dans cet exercice, la suite de nombre suivante : 1, 11, 21, 1211, 111221, ...
Cette suite est construite ainsi: pour passer d'une valeur à la suivante, on la lit et on écrit ce qu'on lit.

Compléter la fonction `nombre_suivant` qui prend en entrée un nombre sous forme de chaine de caractères et qui renvoie le nombre suivant par ce procédé, encore sous forme de chaine.

```
def nombre_suivant(s):
    """
    Renvoie le nombre suivant de celui représenté par s en appliquant le procédé
de lecture.
    """

    res = ""
    chiffre = s[0]
    compte = 0
    for i in range(...):
        if s[i] == chiffre:
            compte += ...
        else:
            res += ... + ...
            chiffre = ...
            compte = ...
    res += ...
    return res
```