Exercice 1 - Nombre d'occurrences d'un caractère.

Programmer la fonction `recherche`, qui prend en paramètre `caractere`, un caractère, et `mot`, une chaîne de caractère, et qui renvoie le **nombre d'occurrences** de `caractere` dans `mot`, c'est à dire le nombre de fois où `caractere` apparaît dans `mot`.

Tests :

- `>>> recherche('e', "sciences")` renvoie 2.
- `>>> recherche('i', "mississippi")` renvoie 4.
- `>>> recherche('a', "mississippi")` renvoie 0.

Exercice 2 - Indice de la dernière occurrence.

Programmer la fonction `recherche`, prenant en paramètre un tableau non vide `tab` (type `list`) d'entiers et un entier n, et qui renvoie **l'indice de la dernière occurrence de l'élément cherché**. Si l'élément n'est pas présent, la fonction renvoie -1.

Tests :

- `>>> recherche([5, 3],1)` renvoie -1.
- `>>> recherche([2,4],2)` renvoie 0.
- `>>> recherche([2,3,5,2,4],2)` renvoie 3.

Exercice 3 - Valeur maximale et son indice dans une liste.

Programmer la fonction `maxi`, prenant en paramètre une liste `tab` (type `list`) d'entiers et qui renvoie un **couple donnant le plus grand élément de cette liste, ainsi que l'indice de la première apparition** de ce maximum dans la liste.

Tests :

- `>>> maxi([1,5,6,9,1,2,3,7,9,8])` renvoie $(9,3)$.
- `>>> maxi([1,5,6,9,1,2,3,57,9,8])` renvoie $(57,7)$.

Exercice 4 - Valeurs minimale et maximale dans une liste.

Programmer la fonction `RechercheMinMax`, prenant en paramètre une liste `tableau` (type `list`) de nombres non triés, et qui renvoie la plus petite et la plus grande valeur du `tableau` sous la forme d'un dictionnaire à deux clés `'min'` et `'max'`.

Tests :

```
>>> tableau = [0, 1, 4, 2, -2, 9, 3, 1, 7, 1]
>>> resultat = rechercheMinMax(tableau)
>>> resultat
{'min': -2, 'max': 9 }
```

```
>>> tableau = []
>>> resultat = rechercheMinMax(tableau)
>>> resultat
{'min': None, 'max': None }
```

Exercice 5 - Indice du minimum.

Programmer la fonction `indice_du_min`, prenant en paramètre une liste `tab` (type `list`) de nombres non triés, et qui renvoie l'indice de la première occurrence du minimum de ce tableau.

Tests :

- `>>> indice_du_min([5])` renvoie 0.
- `>>> indice_du_min([2, 4, 1])` renvoie 2.
- `>>> indice_du_min([5, 3, 2, 2, 4])` renvoie 2.

Exercice 6 - Recherche température minimale et année correspondante.

On a relevé les valeurs moyennes annuelles des températures à Paris pour la période allant de 2013 à 2019. Les résultats ont été récupérés sous la forme de deux listes : l'une pour les températures, l'autre pour les années :

```
t_moy = [14.9, 13.3, 13.1, 12.5, 13.0, 13.6, 13.7]
annees = [2013, 2014, 2015, 2016, 2017, 2018, 2019]
```

Écrire la fonction `mini` qui prend en paramètres le tableau `releve` des relevés et le tableau `date` des dates et qui renvoie la plus petite valeur relevée au cours de la période et l'année correspondante.

Exemple :

```
 >>> mini(t_moy, annees)
12.5, 2016
```

Exercice 7 - Dictionnaire d'occurrence.

On rappelle que l'occurrence d'un caractère dans un phrase est le nombre de fois où ce caractère est présent.

On cherche les occurrences des caractères dans une phrase. On souhaite stocker ces occurrences dans un dictionnaire dont les clefs seraient les caractères de la phrase et les valeurs l'occurrence de ces caractères.

Par exemple : avec la phrase `'Hello world !'` le dictionnaire est le suivant :

```
 {'H': 1,'e': 1,'l': 3,'o': 2,' ': 2,'w': 1,'r': 1,'d': 1,'!': 1 }
```

(l'ordre des clefs n'ayant pas d'importance).

Écrire une fonction `occurence_lettres` avec prenant comme paramètre une variable `phrase` de type `str`. Cette fonction doit renvoyer un dictionnaire de type constitué des occurrences des caractères présents dans la phrase.

Test : vérifier que `occurrence_lettres('Hello world !')` renvoie bien le dictionnaire ci-dessus.

Exercice 8 - Recherche de couples consécutifs.

Programmer la fonction `recherche` qui prend en paramètre un tableau de nombres entiers `tab`, et qui renvoie la liste (éventuellement vide) des couples d'entiers consécutifs successifs qu'il peut y avoir dans `tab`.

Tests :

```
>>> recherche([1, 4, 3, 5])
[]
>>> recherche([1, 4, 5, 3])
 [(4, 5)]
>>> recherche([7, 1, 2, 5, 3, 4])
 [(1, 2), (3, 4)]
>>> recherche([5, 1, 2, 3, 8, -5, -4, 7])
 [(1, 2), (2, 3), (-5, -4)]
```

Exercice 9 - Recherche de l'occurrence maximale.

Écrire une fonction `occurrence_max` prenant en paramètres une chaîne de caractères `chaine` et qui renvoie le caractère le plus fréquent de la chaîne. La chaine ne contient que des lettres en minuscules sans accent.

On pourra s'aider du tableau :

```
alphabet = ['a','b','c','d','e','f','g','h','i','j','k','l','m','n',
'o,','p','q','r','s','t','u','v','w','x','y','z']
```

et du tableau `occurrence` de 26 éléments où l'on mettra dans `occurrence[i]` le nombre d'apparitions de `alphabet[i]` dans la chaine. Puis on calculera l'indice k d'un maximum du tableau occurrence et on affichera `alphabet[k]` .

Exemple :

```
>>> ch = 'je suis en terminale et je passe le bac et je souhaite poursuivre
des etudes pour devenir expert en informatique'
>>> occurrence_max(ch)
 'e'
```

Exercice 10 - Recherche de palindromes.

Un mot palindrome peut se lire de la même façon de gauche à droite ou de droite à gauche : *bob*, *radar*, et *non* sont des mots palindromes.

De même certains nombres sont eux aussi des palindromes : 33 , 121, 345543.

L'objectif de cet exercice est d'obtenir un programme Python permettant de tester si un nombre est un nombre palindrome.

Pour remplir cette tâche, on vous demande de compléter le code des trois fonctions ci-dessous sachant que la fonction `est_nbre_palindrome` s'appuiera sur la fonction `est_palindrome` qui elle-même s'appuiera sur la fonction `inverse_chaine`.

La fonction `inverse_chaine` inverse l'ordre des caractères d'une chaîne de caractères `chaine` et renvoie la chaîne inversée.

La fonction `est_palindrome` teste si une chaine de caractères chaine est un palindrome. Elle renvoie `True` si c'est le cas et `False` sinon. Cette fonction s'appuie sur la fonction précédente.

La fonction `est_nbre_palindrome` teste si un nombre `nbre` est un palindrome. Elle renvoie `True` si c'est le cas et `False` sinon. Cette fonction s'appuie sur la fonction précédente.

Compléter le code des trois fonctions ci-dessous.

```
def inverse_chaine(chaine):
        result = ...
        for caractere in chaine:
                result = ...
        return result

def est_palindrome(chaine):
        inverse = inverse_chaine(chaine)
        return ...

def est_nbre_palindrome(nbre):
        chaine = ...
        return est_palindrome(chaine)
```

Tests :

```
>>> inverse_chaine('bac')
'cab'
>>> est_palindrome('NSI')
False
>>> est_palindrome('ISN-NSI')
True
>>> est_nbre_palindrome(214312)
False
>>> est_nbre_palindrome(213312)
True
```

Exercice 11 - Recherche de séquence ADN.

La fonction `recherche` prend en paramètres deux chaines de caractères `gene` et `seq_adn`, et renvoie `True` si on retrouve `gene` dans `seq_adn` et `False` sinon.
Compléter le code Python ci-dessous pour qu'il implémente la fonction `recherche`.

```
def recherche(gene, seq_adn):
        n = len(seq_adn)
        g = len(gene)
        i = ...
        trouve = False
        while i < ... and trouve == ... :
                j = 0
                while j < g and gene[j] == seq_adn[i+j]:
                        ...
                if j == g :
                        trouve = True
                ...
        return trouve
```

Tests :

- `>>> recherche("AATC", "GTACAAATCTTGCC")` renvoie `True`.
- `>>> recherche("AGTC", "GTACAAATCTTGCC")` renvoie `False`

Exercice 12 - Plus grande série dans une liste.

Programmer une fonction en Python qui prend comme paramètre une liste et renverra le nombre de répétitions de la plus longue série repérée dans la liste.

```
def  plus_grande_serie(liste):
    """
    fonction qui renvoie la longueur
    de la grande série dans une liste
    """

    return longueur
```

Tests :

- `>>> plus_grande_serie( [1, 0, 0, 0, 1, 0, 1, 0, 0, 1] )` renvoie 3 car la plus longue série est composée de 3 zéros.
- `>>> plus_grande_serie( [0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 0] )` renvoie 7 car la plus longue série est composée de 7 uns.