Consigne : vous devez rendre une copie double avec vos réponses aux questions et tous les codes demandés écrits à la main en respectant scrupuleusement l'indentation.

Méthode *Listes en Python*

☞ En Python, il existe un type de données structurées nommé `list` qui permet de stocker des collections ordonnées d'éléments. Il s'agit de l'ordre d'apparition dans la liste pas de l'ordre par comparaison d'éléments. Une liste est délimitée par les crochets `[` et `]` et ses éléments sont séparés par une virgule. Le nombre d'éléments que contient une liste est sa longueur et s'obtient avec la fonction `len`.

On donne ci-dessous l'exemple de l'affectation d'une liste de notes à la variable `notes`.

```
>>> notes = [10, 8, 6, 11, 7]
>>> type(notes)
<class 'list'>
>>> len(notes)
5
```

☞ On accède à un élément d'une liste par son index, les éléments étant indexés de gauche à droite à partir de 0 pour le premier élément. Les éléments de la liste peuvent ainsi être lus ou modifiés.

```
>>> notes[0], notes[len(notes)-1]
10, 7
>>> notes[0] = 9
>>> notes
[9, 8, 6, 11, 7]
```

☞ On peut ajouter un élément à la fin d'une liste avec la fonction `append`. On peut ainsi peupler une liste vide notée `[]`. La réciproque de la fonction `append` est la fonction `pop`, utilisée sans argument elle retourne le dernier élément de la liste. `append` et `pop` s'utilisent avec la notation pointée car il s'agit de fonctions spécifique aux objets de type `list`.

```
>>> notes = []
>>> notes.append(8)
>>> notes
[8]
>>> notes.append(14)
>>> notes
[8, 14]
>>> notes.pop()
14
>>> notes
[8]
```

☞ On peut créer une liste de taille fixée remplie de 0 :

```
>>> t = [0] * 6
[0,0,0,0,0,0]
```

☞ On peut inverser une liste de plusieurs façons :

```
In [14]: t = [1,2,3]

In [15]: t[::-1]          #nouvelle liste distincte de t
Out[15]: [3, 2, 1]

In [16]: t.reverse()      #t est inversée sur place et modifiée

In [17]: t
Out[17]: [3, 2, 1]
```

☞ Un parcours de liste peut s'effectuer de deux façons avec une boucle `for` : en parcourant les index ou directement les valeurs des éléments.

Parcours sur les index

```
notes = [10, 8, 6, 11, 7]
for k in range(len(notes)):
    #affichage des notes
    v = notes[k]
    print(v)
```

Parcours sur les valeurs

```
notes = [10, 8, 6, 11, 7]
for v in notes:
    #affichage des notes
    print(v)
```

Exercice 1 *Parcourir une liste*

1. Parmi les fonctions ci-dessous déterminer celle(s) qui retourne(nt) la somme des éléments de la liste d'entiers `t` passée en paramètre.

```
def somme1(t):
    s = 0
    for k in t:
        s = s + k
    return s
```

```
def somme2(t):
    s = 0
    for k in range(len(t)):
        s = s + k
    return s
```

```
def somme3(t):
    s = 0
    for k in range(len(t)):
        s = s + t[k]
    return s
```

```
def somme4(t):
    s = 0
    for k in range(1,len(t)+1):
        s = s + t[k]
    return s
```

2. Écrire une fonction `produit(t)` qui retourne le produit des éléments d'une liste d'entiers `t`.

3. Écrire une fonction `memeTaille(t1, t2)` qui retourne `True` si les deux listes `t1` et `t2` sont de même longueur et `False` sinon.

4. **a.** Recopier et compléter la fonction `somme2listes(t1, t2)` ci-dessous pour qu'elle retourne une liste `t3` dont chaque élément est la somme des éléments de `t1` et `t2` de même index :

```
def somme2listes(t1, t2):
    if memeTaille(t1, t2):
        t3 = [0] * len(t1)
        for k in range(len(t1)):
```

```
            ....................
        return t3
    return None
```

```
In [12]: somme2listes([1,2,3], [4,5,6])
Out[12]: [5,7,9]
```

b. Quelle est la valeur retournée par l'appel `somme2listes([4,5,6,7], [1,2,3])` ?
À quoi sert le test au début du bloc d'instructions de la fonction ?

Exercice 2 *Recherche séquentielle*

1. Compléter la fonction `rechercheMax(t)` ci-dessous pour qu'elle retourne le maximum d'une liste d'entiers `t` :

```
def rechercheMax(t):
    if len(t) == 0:
        return None
    maxi = t[0]
    for k in range(1, len(t)):
        if .........:
            ..............
    return maxi
```

2. Écrire une fonction `rechercheMiniMaxi(t)` qui retourne le couple `(mini, maxi)` constitué du maximum et du minimum d'une liste d'entiers `t`.

3. Parmi les fonctions ci-dessous déterminer celles qui retournent `True` si la liste contient au moins un entier strictement négatif et `False` sinon :

```
def negatif1(t):
    for e in t:
        if e < 0:
            return True
        else:
            return False
```

```
def negatif2(t):
    for e in t:
        if e < 0:
            return True
    return False
```

```
def negatif3(t):
    for k in range(len(t)):
        if t[k] < 0:
            return True
    return False
```

```
def negatif4(t):
    rep = False
    for k in range(len(t)):
        if t[k] < 0:
            rep = True
    return rep
```

4. Écrire une fonction `listesEgales(t1, t2)` qui retourne `True` si les deux listes `t1` et `t2` sont de même taille et contiennent les mêmes éléments dans l'ordre ou `False` sinon.

⚠ Interdiction d'utiliser l'opérateur de comparaison == entre les deux listes !

Exercice 3 *Générer une liste*

On rappelle qu'en `Python` les opérateurs `//` et `%` retournent respectivement le quotient et le reste de la division euclidienne de deux entiers.

```
In [12]: 143 % 10
Out[12]: 3

In [13]: 143 // 10
Out[13]: 14
```

On considère la fonction `mystere` ci-dessous.

```
def mystere(n):
    t = []
    while n >= 10:
        t.append(n % 10)
        n = n // 10
    t.append(n % 10)
    return t
```

1. Recopier et compléter le tableau d'évolution des variables `t` et `n` lors de l'exécution de `mystere(842)`

	État de la mémoire		
Instruction	Test `n >= 10`	`n`	`t`
`t = []`	×	842	[]
`Test de boucle 1`	Vrai	...	...
`Test de boucle 2`	Vrai	...	...
`Test de boucle 3 ...`	...	...	...
`t.append(n % 10)`	×	...	...

2. Que représente la liste retournée par `mystere(n)` ?

3. **Question bonus**

Un nombre entier palindrome est un nombre qui peut se lire dans les deux sens comme par exemple 353.

Un nombre de Lychrel est un nombre entier naturel qui ne peut pas former de nombre palindrome lorsqu'il est soumis au processus itératif qui consiste à l'additionner au nombre formé de l'inversion de ses chiffres en base 10. À ce jour, il n'existe pas d'entier dont on a prouvé qu'il est de Lychrel, mais on conjecture que certains entiers comme 196 le sont parce que le processus itératif appliqué à ces nombres ne semble pas aboutir sur un nombre palindrome.

Voici quelques exemples de nombres qui ne sont pas de Lychrel :

- 124 nécessite une itération : 124 + 421 = 545
- 59 nécessite 3 itérations :
  59 → 59 + 95 = 154 ; 154 → 154 + 451 = 605 ; 605 → 605 + 506 = 1111

On considère qu'un nombre inférieur à 10 000 est supposé de Lychrel si à partir de ce nombre le processus itératif n'aboutit pas à un palindrome en moins de 50 itérations.

Déterminer la liste des nombres supposés de Lychrel qui sont inférieurs à 10 000.