**Université d'Artois**
Faculté des Sciences Jean Perrin – Département Informatique

---

**Algorithmique et Programmation 3**

**TP7 : Listes chaînées récursives**

*Semestre 3 – Novembre 2025*

---

# Objectif du TP

L'objectif de ce travail est de manipuler les **listes chaînées en version récursive**. Toutes les méthodes doivent être écrites en version récursive, sans utiliser de boucles.

1. **Reprendre les classes *MaillonRec* et *ListeChaineeRec*.**
   Toutes les méthodes implémentées devront être récursives.

2. **Méthode `construire`**
   Réécrire la fonction `construire` vue en TD, qui crée une liste chaînée à partir d'une liste Python passée en argument.

```
>> lc = ListeChainee()
>> lc.construire([1, 4, 7, 0, 12])
>> print(lc)
1 -> 4 -> 7 -> 0 -> 12
```

3. **Méthode `somme`**
   On suppose que la liste contient uniquement des nombres. Écrire la méthode `somme` qui retourne la somme des éléments d'une liste chaînée. La méthode renverra `None` si la liste est vide.

```
>> lc = ListeChainee()
>> lc.construire([1, 4, 7, 0, 12])
>> lc.somme()
24
```

4. **Méthode `appartient`**
   Écrire la méthode `appartient(val)` qui teste si une valeur donnée est présente dans la liste chaînée.

```
>> lc.appartient(0)
True
>> lc.appartient(3)
False
```

5. **Méthode `nb_occ`**
   Écrire la méthode `nb_occ(val)` qui calcule le nombre d'occurrences d'une valeur donnée.

```
>> lc.construire([1, 4, 7, 0, 12, 3, 4, 7, 4, 1, 8])
>> lc.nb_occ(1)
2
>> lc.nb_occ(4)
3
```

6. **Méthode `minimum`**
   Écrire la méthode `minimum()` qui retourne la plus petite valeur de la liste chaînée.

```
>> lc.construire([2, -4, 7, 0])
>> lc.minimum()
-4
```

7. **Méthode `min_max`**
   Écrire la méthode `min_max()` qui retourne le couple (minimum, maximum) de la liste.

```
>> lc.construire([2, -4, 7, 0, 12, 4, -1])
>> lc.min_max()
(-4, 12)
```

8. **Méthode `supprimer_occurrences`**
   Écrire la méthode `supprimer_occurrences(val)` qui supprime toutes les occurrences d'une valeur donnée.

```
>> lc.construire([1, -4, 7, 0, 3, 2, 1, 3, 2])
>> lc.supprimer_occurrences(3)
>> print(lc)
1 -> -4 -> 7 -> 0 -> 2 -> 1 -> 2
```