# Listes chaînées et Tableaux

## Structures linéaires

Une structure linéaire sur un ensemble est une suite d'éléments de cet ensemble (par exemple une suite de nombres entiers). Chaque élément a une place précise. On distingue deux types de structures linéaires : les **tableaux** et les **listes chaînées**.
Dans un **tableau** chaque élément possède un indice qui permet d'accéder directement à l'élément.
Dans une **liste chaînée** chaque élément possède un prédécesseur ou(et) un successeur.
L'avantage du tableau est la rapidité d'accès à un élément, l'avantage de la liste chaînée est la souplesse de la structure.

## Principe des listes chaînées

Un élément d'une liste chaînée est stocké en mémoire avec l'adresse mémoire de l'élément suivant. Donc pour atteindre un élément, il faut passer par tous les éléments précédent. On peut ajouter des éléments à tout endroit de la liste. Le premier élément de la liste est appelé la **tête (head)**, le reste de la liste est appelé la **queue (tail)**.

En Python, il n'existe pas de type correspondant aux listes chaînées. On peut implémenter cette structure avec le type **tuple** ou avec une **classe** en POO.

Création d'une classe `Maillon` et une classe `Liste_Chaine`.

```
class Maillon :
    def __init__(self,valeur=None,suivant=None):
        self.val = valeur
        self.suiv = suivant #valeur du maillon suivant

    def __str__(self): #méthode pour l'affichage
        if self.val is not None:
            return str(self.val) + " - " + str(self.suiv)
        else:
            return str(self.val)

class Liste_Chaine:
    def __init__(self,premier=None):
        self.tete = Maillon(premier)

    def ajoute(self, valeur): #ajout d'un élément en fin de liste
        m = Maillon(valeur) #création d'un maillon
        if self.tete.val is None: #si la liste est vide
            self.tete = m
        else:
            p = self.tete
            while p.suiv is not None: #recherche du dernier élément
                p = p.suiv
            p.suiv = m #m est le dernier élément

    def __str__(self) :
        return str(self.tete)
```

Test des classes **`Maillon`** et **`Liste_Chaine`** :

```
>>> maliste = Liste_Chaine(
>>> maliste.ajoute("N")
>>> maliste.ajoute("S")
>>> maliste.ajoute("I")
>>> print(maliste)
N - S - I - None
```

A faire : compléter le code de la classe **`Liste_Chaine`** avec une fonction **`head`** qui renvoie la tête de la liste et une fonction **`tail`** qui renvoie la queue.

## Principe d'un tableau

Dans un **tableau** (array en anglais) tous les éléments sont du même type et ils occupent tous la même taille en mémoire. La place en mémoire est réservée lors de la création du tableau, on ne peut pas remplacer un élément par un élément d'un autre type et on ne peut pas agrandir la taille du tableau.

En Python les objets de type **`« list »`** sont implémentés de manière à profiter des avantages des tableaux et de ceux des listes chaînées.

En Python, le type « array » n'existe pas, cependant le type « list » peut être utilisé pour simuler un tableau. Certaines bibliothèques, comme **NumPy** permettent de travailler avec des tableaux.