

# LISTES-PILES-FILES : SYNTHÈSE DE COURS

## 1) Type Abstrait de Données : Interface versus Implémentation

Avant de plonger dans le **concret** du code (**implémentation**), il est bon de prendre de la hauteur sur ce dont on a besoin. Dans ce processus d'**abstraction**, il convient notamment de préciser comment organiser les données sur lesquelles on doit travailler.

On réfléchit alors aux **Types Abstrait de Données** (TAD) qui seront utiles et l'on cherche à préciser leur **interface**, c'est à dire les opérations élémentaires dont on aura besoin.

## 2) Les structures de données linéaires et leurs interfaces

### a) Les listes

Une liste est une suite **ordonnée** d'éléments. Ces éléments sont repérés par leur rang (index).

Interface habituelle :

- créer\_liste\_vide()
- insérer(liste, valeur, index)
- supprimer(liste, index)
- lire(liste, index)
- est\_vide(liste) *ou* taille(liste)

**Remarque 1 :** Les listes sont souvent implémentées :

- soit comme listes chaînées (ajouts et suppressions rapides d'un élément)
- soit comme tableaux dynamiques (accès rapide à un élément).

**Remarque 2 :** Le type *list* de Python est une implémentation particulière et très améliorée du type abstrait liste.

**Remarque 3 :** La notion de liste est parfois définie récursivement :

Une liste est alors :

- Soit une liste vide
- Soit une paire contenant une donnée et une liste

### b) Les piles

Une pile est aussi une suite **ordonnée** d'éléments qui donne accès en priorité aux dernières données ajoutées.

Interface habituelle :

- créer\_pile\_vide()
- empiler(pile, valeur)
- dépiler(pile)
- est\_vide(pile)

**Remarque :** En Python, on utilisera souvent le type *list* avec les méthodes `append(valeur)` et `pop()`

### c) Les files

Une file est aussi une suite **ordonnée** d'éléments qui donne accès en priorité aux premières données ajoutées.

Interface habituelle :

- créer\_file\_vide()
- emfiler(file, valeur)
- défiler(file)
- est\_vide(file)

**Remarque :** En Python, on utilisera souvent le type *list* avec les méthodes `append(valeur)` et `pop(0)`