Interface et implémentation

Nous allons dans ce chapitre nous intéresser aux **structures de données** comme les **listes**, les **piles**, les **files** et les **dictionnaires**.

Les dictionnaires ont déjà été rencontrés en première. Plus précisément, c'est l'**implémentation** du **type abstrait** "dictionnaire" en Python qui a été utilisée.

Toutes ces structures de données sont des **types abstraits** qui doivent être définis dans un langage de programmation pour pouvoir être utilisés.

> Définition
>
> **Implémenter** un algorithme, c'est le traduire dans un langage de programmation.

Pour un type abstrait donné, disons les dictionnaires, plusieurs implémentations sont possibles. Elles peuvent se différencier par leur rapidité d'exécution ou leur capacité à travailler avec des données de grande taille par exemple.

Une fois implémenté, le type abstrait doit pouvoir être **utilisé** par un programmeur (utilisateur). Cet utilisateur n'a pas besoin de connaître comment le type "dictionnaire" a été concrètement implémenté dans le langage de programmation qu'il utilise. Par contre, il faut qu'il connaisse précisément les actions qu'i peut réaliser sur ce type de données.

Par exemple, un dictionnaire associe à un ensemble de clés un ensemble de valeurs (ce sont les données) et permet notamment les opérations :

- d'ajout d'un couple clé-valeur ;
- de suppression d'une clé, et donc de la valeur correspondante ;
- de modification de la valeur associée à une clé et ;
- de recherche de la valeur correspondant à une clé.

L'ensemble des fonctions (méthodes) associées à un type abstrait constitue son **interface**. Ces fonctions, et leurs spécifications, permettent à l'utilisateur d'utiliser le type abstrait dans son programme.

Quand on utilise une bibliothèque contenant l'implémentation de structures de données, l'ensemble de ces spécifications est nommée API (Application Programming Interface, Interface de Programmation en français).

Définition (d'après Wikipedia)

En informatique, une **interface de programmation** (souvent désignée par le terme API pour Application Programming Interface) est un ensemble normalisé de classes, de méthodes, de fonctions et de constantes qui sert de **façade** par laquelle un logiciel offre des services à d'autres logiciels. Elle est offerte par une bibliothèque logicielle ou un service web, le plus souvent accompagnée d'une description qui **spécifie** comment des **programmes consommateurs** peuvent se servir des fonctionnalités du **programme fournisseur**.

L'usage des bibliothèques permet à chaque programmeur d'ajouter des structures réalisant des types abstraits de données, cette implémentation n'étant pas nécessairement connue de l'utilisateur de la structure. Cette méthode de conception logicielle, utilisant l'**encapsulation**, permet à la fois :

- le développement séparé de l'application et de l'implémentation de la structure ;
- la modification de l'implémentation sans modification de ses utilisations (on préserve l'interface) ;
- l'utilisation facile de l'implémentation de la structure dans des programmes à venir ;
- la limitation des erreurs ;
- l'ajout :
  - de vérifications sous forme d'assertions ;
  - d'outils de correction des problèmes de programmation ;
- une meilleure lisibilité du code de l'application.

Dans la suite de ce chapitre, nous allons étudier successivement les listes, les piles, les files et les dictionnaire du point de vue du concepteur (implémentation) et de l'utilisateur (interface).