Les prépas ingénieur ESEO INNOVATION MAKES SENSE avec COURS LUMIÈRE

Cours d'informatique

Supposons que tu veux apprendre une recette.

L'enseignant de la recette doit imaginer le chemin de la recette et cela suppose qu'il connaît la structure de chaque ingrédient(donc de chaque donnée)à qui sera utilisé.

Le récipient des données dépend de la structure de ces données.

A quel moment les TAD(Types de Données Abstraites) interviennent-ils ? Dès la base de la conception des logiciels.

1) L'Algorithme
2) Stockage des données
3) L'échange de données

A) Rappels sur les types de données de base

- Entiers(Integers) sont représentés en Binaire(Les Bits "0" et "1" sont utilisés pour représentés en base "2")
- Chaînes de caractères (String) souvent représentés à l'aide du code ASCII. Chaque caractère est associé à un nombre entier qui le représente.
- Nombres décimaux (Float) qui sont représentés par trois parties : le signe, l'exposant et la mantisse.
- True or False(Boolean)
- Matrices ou Tableaux Multi dimensionnels(Array)
- Dictionnaires(Dict)

- Listes(List) - Tuples(Tuple)

a) Les opérations fondamentales associées

Nous avons l'addition(+), la soustraction(-), la multiplication(*), la division(/), le modulo(%), la puissance(**), len, max(), min() . . .
Les opérations fondamentales sont essentielles pour manipuler et traiter les TDB dans la programmation. Elles fournissent les éléments nécessaires pour effectuer des calculs, gérer des chaînes de caractères, accéder et modifier des éléments dans les tableaux; effectuer d'autres opérations de base sur les données.

b) Limitation des types de données de base

Les types de données de base sont fondamentaux en programmation mais ils présentent certaines limitations en termes de flexibilité et d'efficacité.

- Précision limitées des nombres à virgule flottant(essaie de print 0.1 + 0.2 et tu comprendras)
- Limitation de la taille des entiers(au delà d'une certaine limite, Python fait des erreurs)
- Immutabilité(contraire de immuabilité) des chaînes de caractères.
- Le manque de structures complexes(il n'y a pas de listes de tuples par exemple)
- Le manque de représentations abstraites(logiques)

B) Introduction aux TDA

a) Les opérations fondamentales associées

Les types de données abstraites (TDA) sont des modèles mathématiques qui définissent des opérations sur des données sans spécifier comment ces opérations sont implémentées. Pour concevoir un algorithme complexe, on adopte une démarche descendante : l'analyse procède par affinements successifs. Au départ on reste loin de toute implémentation, en particulier; la représentation concrète des données n'est pas fixée. On parle alors du type abstrait de données.

b) Différences entre types de données concrets et types de données abstraits

- Les listes : séquences ordonnées d'éléments. Opérations pouvant être implémentées à une liste
- C'est quoi Pile, File ? C'est quoi les avantages de l'abstraction ?
  Pile : Une pile est une structure de données linéaire qui suit le principe du dernier entré, premier sorti (Last In, First Out - LIFO).

Les opérations de base sur une pile sont push (ajout d'un élément au sommet de la pile) et pop (retrait de l'élément du sommet de la pile).
Une analogie courante est une pile d'assiettes, où vous ajoutez une assiette au sommet de la pile et retirez la dernière assiette ajoutée en premier

File : Une file est une structure de données linéaire qui suit le principe du premier entré, premier sorti (First In, First Out - FIFO).
Les opérations de base sur une file sont “enqueue” (ajout d'un élément à l'arrière de la file) et “dequeue” (retrait de l'élément du devant de la file).
Une analogie courante est une file d'attente, où la première personne à entrer est la première à sortir.

Ensemble : Un ensemble est une structure de données qui représente une collection d'éléments distincts, sans ordre spécifique. La caractéristique fondamentale d'un ensemble est qu'il ne permet pas la duplication d'éléments, et chaque élément est unique dans l'ensemble.

Avantages de l’abstraction :
Simplification : L'abstraction permet de simplifier la complexité en se concentrant sur les aspects essentiels d'un problème sans se perdre dans les détails de mise en œuvre. Cela rend la conception et la compréhension du système plus accessibles.

**Modularité : En utilisant des TDAs, les programmeurs peuvent concevoir des modules indépendants qui interagissent à travers des interfaces bien définies.
Cela favorise la modularité du code, facilitant la maintenance et les mises à jour.**

Réutilisation du Code : En utilisant des abstractions, le code peut être conçu de manière à être réutilisable dans différentes parties d'une application ou même dans des applications différentes.

Encapsulation : Les détails d'implémentation sont cachés derrière l'interface abstraite, ce qui permet d'encapsuler la complexité et de protéger le reste du programme des changements internes.

Généricité : Les TDAs peuvent être conçus de manière générique, ce qui signifie qu'ils peuvent être utilisés avec différents types de données sans modification significative du code, améliorant ainsi la flexibilité du système.

Facilitation de la Maintenance : Lorsque des changements doivent être apportés, la maintenance est facilitée car les modifications peuvent être apportées à l'implémentation interne d'un TDA sans affecter les autres parties du programme qui utilisent cette abstraction.