

Gestion de Stock par Arbre Binaire de Recherche (ABR)

Objectif

Concevoir et implémenter une application en Python pour gérer un catalogue de produits stocké sous forme d'Arbre Binaire de Recherche (ABR). L'application devra permettre l'insertion, la recherche rapide et le calcul de la valeur économique du stock.

Contexte

Dans la gestion de stock, il est crucial de pouvoir insérer de nouveaux produits, les retrouver rapidement et les lister dans un ordre précis. L'**Arbre Binaire de Recherche (ABR)** est une structure de données idéale pour ces tâches.

Propriété de l'ABR

Pour chaque nœud, toutes les clés du sous-arbre gauche sont inférieures à la clé du nœud, et toutes les clés du sous-arbre droit sont supérieures.

Les classes du module module arbre

Pour gérer notre inventaire, nous utilisons deux classes complémentaires qui séparent les informations du produit de l'organisation du stock.

1. La classe **Produit** (La Donnée)

C'est un objet qui contient uniquement les caractéristiques réelles d'un article en magasin :

- **Son identité** : un identifiant unique (`id_produit`) et un nom.
- **Sa valeur** : le prix unitaire et la quantité disponible.

À noter : Cette classe ne sait pas qu'elle fait partie d'un arbre. Elle se contente de stocker les informations.

2. La classe **Stock** (La Structure)

C'est le "nœud" de notre Arbre Binaire de Recherche (ABR). Elle sert de contenant et d'organisateur :

- **L'attribut valeur** : il n'est pas un simple nombre, mais reçoit un **objet** de la classe **Produit**.
- **Les attributs gauche et droit** : ils permettent de lier ce nœud à d'autres objets **Stock**, créant ainsi les ramifications de l'arbre.
- **La méthode visualiser** : elle permet d'afficher l'arbre graphiquement dans la console pour vérifier la bonne organisation des produits (triés par ID).

```
INSTANCE DE LA CLASSE 'STOCK' (Le Nœud)
+-----+
| valeur : -----| |
| gauche : [Stock ou None] | |
| droit  : [Stock ou None] | |
+-----+-----+
|
| v
|
+-----+
| INSTANCE DE LA CLASSE 'PRODUIT' |
+-----+
| id_produit : 50 |
| nom       : "Clavier" |
| stock     : 10 |
| prix_unitaire: 85.0 |
+-----+
```

Travail à réaliser

Vous utiliserez le fichier `gestion_stock_eleves.py` pour répondre et réaliser vos essais.

1. Insérer un produit dans l'arbre

Vous disposez d'une fonction `insérer_produit_dans_stock`.

- 1.1. Compléter la fonction
- 1.2. Visualisez l'arbre
- 1.3. Compléter les tests et les valider

2. Calcul du Stock Total Valorisée

On dispose d'une fonction `calculer_stock_total` qui renvoie le nombre d'objets présents dans le stock. On veut la modifier pour qu'elle renvoie la valeur totale du stock.

- 2.1. Modifier la fonction `calculer_stock_total` et modifier sa spécification.
- 2.2. Valider le test

3. Recherche d'un Produit

Vous disposez d'une fonction `rechercher_produit`. Mais cette fonction est incomplète, il va falloir la compléter :

- 3.1 Compléter la fonction `rechercher_produit`
- 3.2. Valider le test

4. Parcours de l'arbre

On veut effectuer un parcours de l'arbre pour obtenir les produits dans l'ordre croissant des identifiants des produits.

- 4.1. Quel parcours permet d'avoir les nœuds d'un arbre binaire dans l'ordre croissant ?
- 4.2 Écrire la fonction récursive de la fonction `parcours`.
- 4.3. Valider les tests
- 4.4. En déduire le code permettant d'afficher un inventaire du stock :

Liste des produits dans l'ordre de leur identifiant:

5 : Adaptateur USB-A vers C, 2500, 3.9

10 : Tapis de souris XXL, 800, 12.5

15 : Clé USB 64Go, 1200, 9.99

...

120 : Scanner de documents, 40, 155.5

130 : Tablette Graphique, 70, 89.0