Projet de Gestion d'une Boutique en ligne

Groupe 3 : Nessa Crebessac, Imane Benyettou

Mars 2025

Formation : L3 MIAGE
Matière : Programmation Orientée Objet
Année universitaire : 2024-2025
Professeur responsable : Madame Benois-Pineau
Date de production : Mars 2025

Table des matières

1 Introduction

Ce projet consiste en la conception et l'implémentation d'un système de gestion de magasin en Python, un langage de programmation interpreté multi-paradigme et multiplateforme crée en 1991 par Guido Van Rossum, doté d'un fort typage dynamique, d'une gestion automatique de la mémoire, et d'une syntaxe intuitive, il représente le langage idéal pour programmer des applicatifs de manière simplifiée

Ici, le système permet de gérer l'inventaire des produits, traiter les commandes des clients, suivre les ventes quotidiennes et générer des statistiques. Nous séparerons le projet en 3 classes principales qui assurerons chacune un role précis.

L'objectif principal est de créer une application robuste mais simple, nous permettant d'implémenter les principes fondamentaux de la programmation orientée objet, tout en utilisant des structures de données efficaces comme les dictionnaires et des bibliothèques externes comme numpy et matplotlib pour l'analyse et la visualisation de données.

Le système de gestion de magasin automatise plusieurs tâches essentielles, notamment :

- La gestion des produits (ajout, affichage des détails)
- Le traitement des commandes (vérification du stock, mise à jour de l'inventaire)
- L'annulation de commandes (restauration du stock)
- L'analyse des ventes quotidiennes (statistiques, visualisations)

2 Architecture du programme

Le système est structuré autour de trois classes principales qui interagissent entre elles :

2.1 Classe 'Product'

La classe `Product` représente un produit disponible dans le magasin. Elle possède les propriétés suivantes

- `product_id` : Identifiant unique du produit
- `name` : Le nom du produit
- `price` : Le prix du produit
- `stock` : La quantié disponible en stock

La classe implémente également la méthode `afficher_details_produit()` qui permet d'afficher les informations détaillées du produit de manière formatée.

2.2 Classe 'Order'

La classe `Order` représente une commande passée par un client. Elle compote les attributs suivants :

- `order_id` : Identifiant unique de la commande
- `customer_name` : Nom du client
- `order_date` : Date de la commande (au format DD/MM/YYYY)
- `produits` : Dictionnaire contenant les produits commandés et leurs quantités

La classe implémente la méthode `afficher_details_commande()` qui présente les détails de la commande.

2.3 Classe 'StoreManagement'

La classe `StoreManagement` constitue le cœur du système et gère l'interaction entre les produits et les commandes. Elle contient :

- `produits` : Liste des produits disponibles
- `commandes` : Liste des commandes passées
- `ventes_quotidiennes` : Tableau numpy représentant les ventes pour chaque jour du mois

Elle implémente plusieurs méthodes essentielles :

- `ajouter_produit()` : Ajoute un nouveau produit à l'inventaire
- `passer_commande()` : Prend en compte une nouvelle commande, vérifie la disponibilité et met à jour le stock
- `annuler_commande()` : Annule une commande existante et restaure le stock
- `afficher_produits()` : Affiche la liste des produits disponibles
- `afficher_commandes()` : Affiche la liste des commandes passées
- `afficher_ventes_quotidiennes()` : Génère un histogramme des ventes journalières
- `calculer_statistiques_ventes()` : Calcule des statistiques sur les ventes (ici la moyenne, l'écart-type, les ventes min et max)

3 Principes de la programmation objet utilisés

3.1 Encapsulation

L'encapsulation est mise en œuvre en regroupant les données (attributs) et les méthodes qui opèrent sur ces données au sein de classes. Chaque classe encapsule l'état et le comportement d'une entité :

- La classe `Product` encapsule les informations sur les produits
- La classe `Order` encapsule les informations sur les commandes
- La classe `StoreManagement` encapsule la logique de gestion du magasin

3.2 Abstraction

L'abstraction est réalisée en exposant uniquement les fonctionnalités nécessaires tout en masquant les détails d'implémentation complexes. Par exemple :

— La méthode `passer_commande()` gère toute la logique complexe de vérification du stock, mise à jour de l'inventaire et enregistrement des ventes, tout en présentant une interface simple à l'utilisateur

— La méthode `afficher_ventes_quotidiennes()` cache les détails techniques de la génération d'un histogramme avec Matplotlib, elle est très utile en terme d'accès restreint des données à certains potentiels utilisateurs de ce système

3.3 Composition

La composition est utilisée pour créer des relations entre les objets. La classe `StoreManagement` contient des listes d'objets `Product` et `Order`, illustrant une relation de composition où le magasin est composé de produits et de commandes.

3.4 Responsabilité unique de chaque classe

Chaque classe a une responsabilité unique et bien définie :
— `Product` gère uniquement les informations relatives aux produits
— `Order` traite uniquement les informations relatives aux commandes
— `StoreManagement` coordonne les interactions entre produits et commandes

4 Utilisation des dictionnaires

Les dictionnaires jouent un rôle central dans ce projet, notamment pour gérer les relations entre les produits et les quantités commandées.

1. **Dans la classe Order** :
   — Le dictionnaire `produits` associe chaque ID de produit à sa quantité commandée
   — Cette structure permet de parcourir facilement les produits commandés lors de l'affichage des détails
2. **Dans la méthode passer_commande()** :
   — Le dictionnaire `produits_commande` permet de vérifier efficacement si chaque produit est disponible en stock
   — La même structure est utilisée pour mettre à jour les quantités en stock
3. **Dans la méthode annuler_commande()** :
   — Le dictionnaire des produits de la commande annulée est parcouru pour restaurer le stock

4.1 Avantages des dictionnaires

1. **Accès rapide aux données** : Les dictionnaires permettent un accès en O(1) aux données, ce qui est crucial pour les opérations fréquentes de vérification et de mise à jour du stock.

2. **Représentation intuitive des associations** : Dans notre système, un dictionnaire est utilisé pour représenter la relation entre les ID des produits et les quantités commandées, ce qui est une représentation naturelle et efficace.

3. **Flexibilité** : Les dictionnaires permettent de stocker des informations supplémentaires ou de modifier facilement la structure de données sans changer l'interface.

5 Utilisation de bibliothèques externes

5.1 numpy

nous utilisons numpy pour :
— Stocker et manipuler efficacement les données de ventes quotidiennes avec un tableau (`self.ventes_quotidiennes`)
— Calculer des statistiques descriptives (moyenne, écart-type, min, max) sur les ventes

5.2 matplotlib

nous utilisons matplotlib pour :
— Visualiser les ventes quotidiennes sous forme d'histogramme
— Présenter les tendances de ventes de manière graphique et intuitive

6 Execution du programme

6.1 Le fichier main.py

le fichier main importe les classes que nous avons programmé au préalable, afin qu'il puisse accèder et appeller leur méthodes.

L'execution du programme se fait avec la commande suivante

```
python3 main.py
```

6.2 Focus sur le code du main.py

On commence par créer une instance du magasin pour gérer l'ensemble du système avec la ligne suivante

```
magasin = StoreManagement()
```

On crée ensuite plusieurs produits que l'on ajoute à l'inventaire du magasin en appellant la méthode `ajouter_produit` de la classe `StoreManagement`, que l'on pourra ensuite afficher en appelant `afficher_produits` de la même classe

A l'execution, on à l'affichage suivant, avec les données de test manuellement ajoutées dans le main

```
Liste des produits disponibles:
=== LISTE DES PRODUITS ===
Product ID: 1
Name: Smartphone
Price: 699.99
Stock: 50
------------------------------
Product ID: 2
Name: Laptop
Price: 1299.99
Stock: 30
------------------------------
Product ID: 3
Name: Airpods
Price: 129.99
Stock: 100
------------------------------
Product ID: 4
Name: Tablette
Price: 799.99
Stock: 40
------------------------------
Product ID: 5
Name: Apple Watch
Price: 349.99
Stock: 60
------------------------------
```

Nous pouvons de la même manière créer des commandes, à l'aide de la méthode `passer_commande` de la classe `StoreManagement`, qui prend en argument le nom de client, ainsi que les produits (id) et la quantité respectives. On peut également afficher la totalité des commandes passées

```
Liste des commandes:
=== LISTE DES COMMANDES ===
Order ID: 1
Customer Name: Martin Matin
Order Date: 05/03/2025
Products:
Product ID: 1, Quantity: 2
Product ID: 3, Quantity: 1
--------------------------------
Order ID: 2
Customer Name: Nessa
Order Date: 10/03/2025
Products:
Product ID: 2, Quantity: 1
Product ID: 5, Quantity: 2
--------------------------------
Order ID: 3
Customer Name: Imane
Order Date: 15/03/2025
Products:
Product ID: 4, Quantity: 3
Product ID: 3, Quantity: 2
--------------------------------
Order ID: 4
Customer Name: Angelo la debrouille
Order Date: 15/03/2025
Products:
Product ID: 1, Quantity: 1
Product ID: 2, Quantity: 1
Product ID: 5, Quantity: 1
--------------------------------
Order ID: 5
Customer Name: Lucas
Order Date: 20/03/2025
Products:
Product ID: 3, Quantity: 5
Product ID: 5, Quantity: 1
--------------------------------
Order ID: 6
Customer Name: Emma
Order Date: 25/03/2025
Products:
Product ID: 1, Quantity: 3
Product ID: 4, Quantity: 2
--------------------------------
```

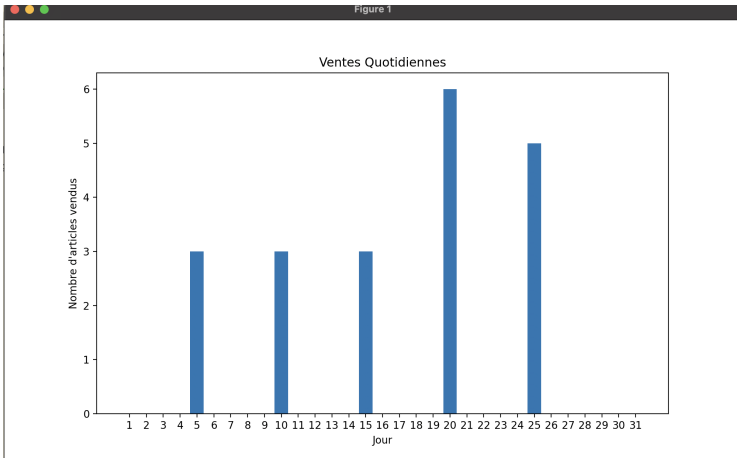
Chaque commande à un identifiant unique, qui peut être utilisé pour éventuellement annuler la commande avec la méthode `annuler_commande(id_commande)`, qui remettra en stock les produits commandés dans leur quantités respectives, et supprimera celle-ci du système. A chaque annulation, on affiche la liste des commandes après l'opération

Concernant la visualisation des ventes et statistiques, on appelle la méthode `calculer_statistiques_ventes` de la classe `StoreManagement`. Ces statistiques sont calculées grâce à numpy, et incluent la moyenne des ventes quotidiennes,

l'ecart type des ventes, et le minimum et maximum de ventes quotidiennes. En sortie, pour nos données test, on a :

```
Statistiques des ventes quotidiennes: {'moyenne': 0.6451612903225806, 'ecart_type': 1.55643072023417, 'minimum': 0, 'maximum': 6}
- Moyenne: 0.65 articles par jour
- Écart-type: 1.56
- Minimum: 0 articles
- Maximum: 6 articles
```

Par ailleurs, matplotlib nous permet de génerer un histogramme nous permettant de visualiser les ventes quotidiennes, afin d'avoir une vue globale sur la performance du magasin, intéressant d'un point de vue statistique. Nous pouvons afficher l'histogramme avec la méthode `afficher_ventes_quotidiennes`. En sortie, on à le graphique suivant avec les 31 derniers jours en abcisse, et le nombre de ventes en ordonnée.



7 Conclusion

Ce projet implémentant un système de gestion de magasin illustre l'application pratique des principes de la programmation orientée objet dans un contexte concret. La structure modulaire et l'utilisation de concepts comme l'encapsulation, l'abstraction et la composition ont permis de créer un système robuste et facile à maintenir.

L'utilisation judicieuse des dictionnaires comme structure de données principale pour associer les produits et leurs quantités offre une solution efficace et intuitive pour la gestion des commandes. De plus, l'intégration de bibliothèques comme numpy et matplot enrichit le système avec des capacités d'analyse et de visualisation des données.

8 Annexe et Bibliographie

8.1 Diagramme de classe

8.2 Bibliographie

1. Real Python. (s.d.). Python 3 Object-Oriented Programming. Real Python. https ://realpython.com/python3-object-oriented-programming/
2. W3Schools. (s.d.). Python Tutorial. W3Schools.
3. Pr. Benois-Pineau, J. (s.d.). Programmation Orientée Objet – Python, Chapitre 3 : Classes en Python [Cours de L3 MIAGE].
4. Pr. Benois-Pineau, J. (s.d.). Programmation Orientée Objet – Python, Chapitre 4 : NumPy [Cours de L3 MIAGE].