**Cours programmation orientéé objet**

**Différences et types entre les deux projets**

Tu travailles sur **deux projets différents** qui ont chacun **des objectifs et une approche distincte**. Voici les différences détaillées :

**🔷 Projet 2 : Scraper Books to Scrape (projet\_scraper\_poo)**

📌 **Type** : **Scraper Web**  
📌 **Objectif** : Extraire des informations sur les livres depuis un site web

**🔹 Caractéristiques du projet 2**

✅ **Utilisation de BeautifulSoup** pour récupérer les données du site  
✅ **Stockage des données en CSV et JSON** pour exploitation future  
✅ **Téléchargement des images des livres**  
✅ **Gestion des erreurs pour éviter les crashs**  
✅ **Architecture MVC** pour organiser le code proprement

**🔹 Type de projet**

* **Projet de scraping** (extraction de données depuis un site web)
* **Automatisation** (le programme fonctionne sans intervention humaine)
* **Traitement des données** (formatage, stockage et analyse)

👨‍💻 **Exemple d’application réelle** :  
Un e-commerce pourrait utiliser un scraper pour **surveiller les prix** de la concurrence.

**🔷 Projet 4 : Gestion d’un Tournoi d’Échecs (tournoi)**

📌 **Type** : **Application de gestion locale**  
📌 **Objectif** : Organiser et suivre un tournoi d’échecs en **hors ligne**

**🔹 Caractéristiques du projet 4**

✅ **Programmation Orientée Objet (POO)** pour structurer les données (tournois, joueurs, matchs)  
✅ **Utilisation de JSON pour stocker les informations des tournois**  
✅ **Architecture MVC pour séparer les responsabilités**  
✅ **Génération automatique des paires de joueurs**  
✅ **Classement des joueurs selon les points gagnés**

**🔹 Type de projet**

* **Application locale** (contrairement au scraper qui dépend d’un site web)
* **Gestion de données structurées** (tournois, joueurs, scores)
* **Interactivité avec l’utilisateur** (contrairement au scraper qui fonctionne en automatique)

👨‍💻 **Exemple d’application réelle** :  
Une fédération d’échecs pourrait utiliser ce logiciel pour **gérer ses tournois hors ligne** sans avoir besoin d’une base de données en ligne.

**📌 Différences clés entre les deux projets**

| **Critère** | **Projet 2 : Scraper** | **Projet 4 : Tournoi** |
| --- | --- | --- |
| **Type de programme** | Automatisation (Scraping) | Application locale |
| **Utilisation** | Extraire des données d’un site web | Gérer des données structurées |
| **Dépendance Internet** | Oui (besoin du site cible) | Non (fonctionne hors ligne) |
| **Stockage** | JSON & CSV (données web) | JSON (tournois, joueurs) |
| **Interaction** | Aucune (script automatisé) | Interface CLI (interaction utilisateur) |
| **Objectif final** | Analyser des livres | Organiser un tournoi |

**📌 Pourquoi ces deux projets sont utiles pour toi ?**

1. **Projet Scraper (P2) → Apprentissage du Web Scraping** 🕵️‍♂️  
   → Apprendre **BeautifulSoup**, **requests**, le **traitement des données**
2. **Projet Tournoi (P4) → Apprentissage de la POO et gestion de projet** ♟  
   → Travailler sur **les classes, la persistance des données, la gestion d’événements**

💡 **En conclusion** :  
Ces deux projets sont **complémentaires**.  
✅ **Le scraper t’apprend à extraire et manipuler des données**  
✅ **Le tournoi t’apprend à gérer une application et ses fonctionnalités**

Si ton mentor te demande la différence entre les deux, tu peux dire :  
👉 **"Le premier est un outil d’automatisation, tandis que le second est une application interactive pour gérer un événement."** 🚀

**📌 Différence entre un programme procédural (Projet 2) et un programme orienté objet (Projet 4)**

**🔷 Projet 2 : Approche procédurale / impérative**

📌 **Définition** :  
Un programme **procédural** est écrit sous forme d'une **suite d'instructions** exécutées **de haut en bas**, organisées en **fonctions indépendantes**.

📌 **Caractéristiques du projet 2 (Scraper)** :  
✅ **Chaque fonction fait une tâche spécifique** (ex: récupérer les catégories, scraper un livre, sauvegarder les données).  
✅ **Pas d’objets** : on manipule **des dictionnaires, des listes et des variables globales**.  
✅ **Flux linéaire** : on exécute une **série d’étapes** du début à la fin sans relation entre elles.  
✅ **Données séparées du traitement** : les livres sont juste des **dictionnaires**, pas des objets.

**👨‍💻 Exemple :**  
C’est comme **une recette de cuisine** où tu suis **chaque étape** dans l’ordre.

**🔷 Projet 4 : Approche orientée objet (POO)**

📌 **Définition** :  
La **Programmation Orientée Objet (POO)** repose sur la **création d’objets** qui contiennent **des données et des comportements**.

📌 **Caractéristiques du projet 4 (Tournoi d’échecs)** :  
✅ **Utilisation de classes et d’objets** pour représenter **les joueurs, les tournois, les matchs**.  
✅ **Les données et les actions sont regroupées** : par exemple, un **objet "Joueur" contient son nom + son score + ses méthodes**.  
✅ **Modularité et réutilisation** : les objets permettent de **modifier et faire évoluer le projet facilement**.  
✅ **Interactivité** : contrairement au scraper, ici, l’utilisateur peut **ajouter des joueurs, modifier un tournoi, etc.**

**👨‍💻 Exemple :**  
C’est comme **un jeu de société** où chaque joueur a sa propre **fiche avec ses caractéristiques et actions possibles**.

**📌 Comparaison directe**

| **Critère** | **Projet 2 : Scraper (procédural)** | **Projet 4 : Tournoi (POO)** |
| --- | --- | --- |
| **Approche** | Fonctionnelle (fonctions indépendantes) | Objet (classes et méthodes) |
| **Organisation** | Suite d’instructions exécutées en séquence | Objets qui interagissent entre eux |
| **Données** | Stockées dans des dictionnaires et listes | Stockées dans des objets avec attributs |
| **Modularité** | Difficile à modifier ou étendre | Facile à réutiliser et modifier |
| **Interactivité** | Automatique, sans interaction utilisateur | L’utilisateur peut manipuler les données |
| **Exemple concret** | Une recette de cuisine (étapes fixes) | Un jeu avec des personnages et des règles |

**📌 Comment expliquer ça à ton mentor ?**

Si ton mentor te demande **"Pourquoi as-tu utilisé la POO pour le projet 4 et pas le projet 2 ?"**, tu peux répondre :

👉 **"Le projet 2 est un simple script de scraping, il suit une logique linéaire, donc une approche procédurale suffisait. Mais le projet 4 est une application interactive avec plusieurs entités liées entre elles (joueurs, matchs, tournois), donc la POO permet une meilleure organisation et évolutivité."** 🚀

**🎤 1️⃣ Présentation du projet et de son architecture**

**❓ Question : Peux-tu expliquer ton projet ?**

💬 **Réponse :**  
Ce projet est un **scraper** qui extrait des informations sur les livres du site [Books to Scrape](https://books.toscrape.com/).

**📌 Fonctionnalités principales**

1. **Scraper les livres** → Récupération du titre, prix, stock, notation, catégorie, image
2. **Enregistrement des données** → Stockage en **JSON** et **CSV**
3. **Téléchargement des images** → Sauvegarde des couvertures des livres
4. **Gestion des erreurs** → Vérification des requêtes et des erreurs réseau
5. **Exclusion de la catégorie "Books"** → Éviter le scraping des livres en double

**❓ Pourquoi as-tu choisi une architecture MVC (Modèle-Vue-Contrôleur) ?**

💬 **Réponse :**  
J’ai utilisé **MVC** pour **séparer les responsabilités** et rendre le code **plus modulaire et maintenable** :

| **Composant** | **Rôle** | **Fichiers** |
| --- | --- | --- |
| **Modèle (M)** | Gère les données | models/book.py, models/data\_manager.py |
| **Vue (V)** | Affichage des données | views/display.py |
| **Contrôleur (C)** | Gère le flux du programme | controllers/scraper\_controller.py |

Cela permet **d'ajouter facilement des fonctionnalités** sans tout casser.

**❓ Peux-tu expliquer comment ton projet est structuré ?**

💬 **Réponse :**  
Voici la structure du projet :

📂 projet\_scraper\_poo

├── 📂 controllers/ # Gère la logique du programme

│ ├── scraper\_controller.py # Contrôleur principal

│

├── 📂 models/ # Gère les données

│ ├── book.py # Modèle de livre

│ ├── data\_manager.py # Gestion du stockage en JSON et CSV

│

├── 📂 utils/ # Outils pour le scraping

│ ├── scraper.py # Extraction des données du site

│ ├── image\_download.py # Téléchargement des images

│

├── 📂 views/ # Gestion de l'affichage

│ ├── display.py # Affichage des livres

│

├── 📂 data/ # Stocke les fichiers JSON, CSV et images

│

├── main.py # Point d'entrée du programme

├── README.md # Documentation

├── requirements.txt # Dépendances Python

**🎤 2️⃣ Fonctionnement du Scraper et Gestion des Données**

**❓ Comment fonctionne ton scraper ?**

💬 **Réponse :**  
Le scraper fonctionne en **plusieurs étapes** :

1. **Récupération des catégories** (get\_category\_links)
2. **Récupération des liens des livres par catégorie** (get\_books\_links)
3. **Extraction des données de chaque livre** (extract\_book\_data)
4. **Enregistrement des résultats en JSON et CSV** (save\_books\_to\_json, save\_books\_to\_csv)
5. **Téléchargement des images** (save\_image)

Tout est **automatisé** pour éviter le scraping en double et accélérer le processus.

**❓ Comment récupères-tu les données des livres et les enregistres-tu ?**

💬 **Réponse :**

* **Les données sont extraites avec BeautifulSoup** (scraper.py)
* **Les données sont nettoyées** (clean\_text(), clean\_price())
* **Les livres sont enregistrés en deux formats** :
  + **JSON** → Facile à recharger pour éviter de rescraper
  + **CSV** → Utile pour Excel ou un autre traitement externe

**❓ Pourquoi as-tu choisi d’enregistrer en JSON et CSV ?**

💬 **Réponse :**

* **JSON** 📂 → Chargement rapide en Python sans refaire le scraping
* **CSV** 📊 → Facile à utiliser avec Excel ou d'autres outils

Cela permet d'avoir **un historique** et de **gagner du temps**.

**❓ Comment gères-tu les erreurs lors du scraping ?**

💬 **Réponse :**  
J’ai ajouté **plusieurs sécurités** :

* Vérification du **statut HTTP** (requests.get(book\_url).status\_code)
* **Try/Except** pour éviter que le programme ne plante
* **Gestion des erreurs réseau** (requests.exceptions.RequestException)
* **Logs détaillés** pour voir les problèmes ([ERREUR] Impossible de scraper…)

**❓ Comment fais-tu pour éviter de scraper la catégorie "Books" qui contient tous les livres ?**

💬 **Réponse :**  
Dans get\_category\_links(), j’**exclus la catégorie "Books"** :

python

CopierModifier

if name.lower() != "books": # ⚠️ Exclure la catégorie "Books"

category\_links[name] = link

Cela évite de scraper **les livres en double** et **réduit le temps d’exécution**.

**🎤 3️⃣ Justification des choix techniques**

**❓ Pourquoi utilises-tu @classmethod dans Book pour scraper les données ?**

💬 **Réponse :**  
J’ai mis @classmethod pour pouvoir **créer une instance de Book directement depuis une URL** :

python

CopierModifier

@classmethod

def scrape\_from\_url(cls, url):

book\_data = extract\_book\_data(url)

return cls(\*\*book\_data)

Cela **simplifie** le code et permet d’extraire un livre **avec une seule ligne**.

**❓ Peux-tu expliquer l’intérêt de \_\_repr\_\_() et pourquoi tu l’as (ou pas) utilisé ?**

💬 **Réponse :**  
\_\_repr\_\_() est utile pour **afficher proprement un objet Book** :

python

CopierModifier

def \_\_repr\_\_(self):

return f"<Book {self.title} - {self.price}€>"

Je l’ai ajouté car **c’est pratique pour le debug** et ça rend les affichages plus clairs.

**❓ Comment as-tu nettoyé les données pour éviter les problèmes de caractères spéciaux ?**

💬 **Réponse :**  
J’ai ajouté **deux fonctions** :

python

CopierModifier

def clean\_text(text):

return re.sub(r"[^\w\s\-\']", "", text) # Supprime les caractères non désirés

def clean\_price(price):

return float(re.sub(r"[^\d.]", "", price)) # Supprime tout sauf les chiffres et le point

Cela permet **d’éviter les erreurs de parsing** et d’avoir des prix en format **float**.

**❓ Comment gères-tu le téléchargement des images et où les stockes-tu ?**

💬 **Réponse :**  
Les images sont **stockées dans data/images/** et **triées par catégories**.  
J’utilise cette fonction pour éviter les noms illisibles :

python

CopierModifier

def save\_image(image\_url, title, category, image\_folder="data/images"):

safe\_title = clean\_text(title)[:50] # Limite à 50 caractères pour éviter les erreurs

image\_path = os.path.join(image\_folder, category, f"{safe\_title}.jpg")

Cela évite **les caractères spéciaux** et **les noms trop longs**.

**🎤 4️⃣ Tests et robustesse du code**

**❓ Comment as-tu testé ton projet pour vérifier qu’il fonctionne bien ?**

💬 **Réponse :**  
J’ai fait **plusieurs tests** :  
✔ Vérification manuelle avec **JSON et CSV**  
✔ Test du **scraping d’un seul livre**  
✔ Test avec **différentes catégories**

**❓ Quelles améliorations tu pourrais encore apporter à ton projet ?**

💬 **Réponse :**

1. **Ajouter des tests automatisés (pytest)**
2. **Améliorer la gestion des erreurs (timeout, proxies, anti-bot)**
3. **Optimiser le téléchargement des images (multithreading)**

**🎤 5️⃣ Git et gestion du projet**

**❓ Comment as-tu organisé ton dépôt Git ?**

💬 **Réponse :**  
J’ai utilisé **des commits clairs** et **un README bien structuré**.  
Le dépôt est sur **GitHub** : [lien vers le repo](https://github.com/r3n3gat/Projet4_Tournoi).