Documentation technique du script d'importation des données

Nom du fichier principal : main_import.py

Dépendances :

Pour la bonne exécution de ce script et les autres fonctionnalités de ce projet, il est nécessaire d’exécuter les dépendances nécessaires pour ce projet.

Ainsi de créer un environnement virtuel, comme conda avec la commande :

<conda create --name <nom_de_l’env> python=3.10.12

Puis de l’activer :

<conda activate <nom_de_l’env>>

Et d’installer les dépendances à partir de requirements.txt :

<pip install -r requirements.txt>

Cette commande installera toutes les bibliothèques spécifiées dans le fichier requirements.txt dans l'environnement Conda que vous avez activé

Contexte du projet :

Le projet vise à extraire, transformer et consolider plusieurs jeux de données provenant d'API et de fichiers CSV pour générer une base de donnée utilisée dans l’analyse du projet. Les données incluent des informations sur :

- Les objets trouvés dans les gares parisiennes de l'API SNCF de 2021 à 2023.
- La fréquentation des gares parisiennes de 2021 à 2023.
- Les données météorologiques de Paris sur les années 2021 à 2023.

Objectif du script :

Le script **main_import.py** est conçu pour automatiser l'exécution de plusieurs scripts Python qui extraient et traitent les données de différentes sources, puis consolident ces données en fichiers CSV.

Il gère l'exécution de trois scripts :

1. **frequentation_api.py** : Extraction des données de fréquentation des gares parisiennes via l'API SNCF.
2. **objets_trouves_api.py** : Extraction des données relatives aux objets trouvés dans les gares parisiennes via l'API SNCF.
3. **concat.py** : Concaténation des fichiers CSV météorologiques téléchargés pour les 36 mois de 2021, 2022, et 2023.

Fonctionnalités clés :

1. **Initialisation des chemins** :
    - Le script utilise le module **`os`** pour interagir avec le système de fichiers, et définit le chemin vers les scripts à exécuter en fonction de l'endroit où le script est lancé. Le fichier est à lancer à la racine du projet avec la commande :

<python main_import.py>

2.**Exécution de sous-processus** :

- Le script utilise le module **subprocess** pour lancer les autres scripts Python comme sous-processus. Cela permet de capturer la sortie de ces scripts (succès ou erreurs) et d'afficher les résultats dans la console.

- La commande **subprocess.run** exécute chaque script en tant que processus séparé et capture sa sortie (normale et erreurs). Si l'exécution échoue, une erreur est levée et gérée de manière appropriée.
- **Gestion des erreurs** :

  Chaque tentative d'exécution d'un script est entourée d'un bloc **try-except**, qui permet de capturer les erreurs si un des scripts ne s'exécute pas correctement. Un message d'erreur détaillé est affiché pour faciliter le débogage.
- **Affichage des résultats** :

  Lorsqu'un script s'exécute avec succès, un message de confirmation est affiché. Si une erreur se produit, un message d'erreur détaillé est affiché pour identifier la source du problème.

Structure du Script

1. Imports :

2. Fonction `run_script` :

Cette fonction est responsable de l'exécution d'un script Python externe et de la gestion des erreurs associées. Elle prend deux arguments :

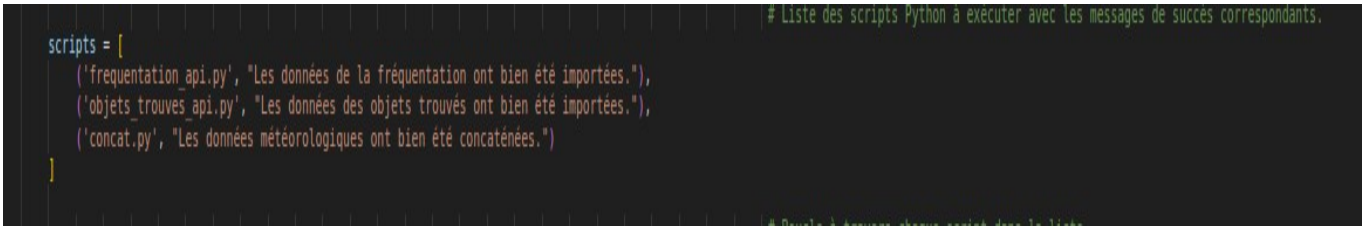
- **script_name** : Le nom du script à exécuter.
- **success_message** : Un message à afficher si l'exécution du script réussit.

3. Exécution conditionnelle des scripts :

Dans la fonction principale, une boucle parcourt une liste de tuples contenant les noms des scripts à exécuter et les messages de succès correspondants. Ces scripts sont :

- **frequentation_api.py** : Extraction des données de fréquentation des gares via l'API SNCF.
- **objets_trouves_api.py** : Extraction des données d'objets trouvés via l'API SNCF.
- **concat.py** : Concaténation des fichiers CSV de données météorologiques.

Exemple de la liste des scripts :



4. **Fonction principale** :

Le script vérifie si **main_import.py** est exécuté directement (et non importé comme module). Si c'est le cas, il lance l'exécution des scripts dans la liste.

Exemple de la boucle qui exécute chaque script :

Scripts externes exécutés et extraction dans le dossier <csv_modélisé>

1. **frequentation_api.py** : Ce script extrait les données de l'API SNCF concernant la fréquentation des gares et génère un fichier CSV appelé **frequentation.csv**.
2. **objets_trouves_api.py** : Ce script interroge l'API SNCF pour récupérer les données des objets trouvés dans les gares parisiennes et génère un fichier **objets_trouves.csv**.
3. **concat.py** : Ce script concatène les 36 fichiers CSV contenant les données météorologiques journalières de Paris sur trois ans (2021, 2022, 2023) en un seul fichier **all_meteo.csv**.

Extraction des données en fichiers CSV :

L’exécution des 3 fichiers va générer 3 créations de csv différents qui seront présents dans le dossier <csv_modélisé>, ainsi le fichier :

- <all_meteo.csv> pour les données météorologiques

- <frequentation.csv> pour les données de fréquentation
- <objets_trouves.csv> pour les données relatives aux objets trouvés

Conclusion :

Le script **`main_import.py`** est un point d'entrée simple et efficace pour automatiser l'exécution d'autres scripts d'extraction et de transformation de données. Grâce à une gestion claire des chemins, des sous-processus et des erreurs, il permet une gestion centralisée et fiable des tâches d'importation de données pour le projet.