**Interface AST - Transformations Modulaires**

Interface graphique pour appliquer des transformations automatisées sur du code Python en utilisant des fichiers JSON comme instructions.

**🎯 Concept Principal**

Cette application utilise un **système de plugins modulaires** où :

* Les **fichiers Python** (.py) sont les **cibles à modifier**
* Les **fichiers JSON** sont les **fichiers d'ordres** contenant les instructions
* Les **moteurs de transformation** appliquent les JSON sur les fichiers Python

**🚀 Processus d'Utilisation (4 Étapes)**

**1. Sélectionner les fichiers Python cibles**

* Cliquez sur **"Sélectionner Fichiers"** pour choisir des fichiers .py spécifiques
* OU cliquez sur **"Sélectionner Dossier"** pour traiter tous les .py d'un dossier
* Les fichiers sélectionnés apparaissent dans la liste

**2. Choisir le moteur de transformation**

* Dans la liste déroulante **"Transformations"**
* Sélectionnez le plugin **"Transformations JSON-AI"**
* ⚠️ **Important** : Ceci indique à l'application que les instructions viendront d'un fichier JSON

**3. Charger le fichier d'instructions JSON**

* Cliquez sur le bouton **"JSON"**
* Sélectionnez le fichier .json contenant les instructions de transformation
* Le JSON est validé et chargé comme "fichier d'ordres"

**4. Appliquer les transformations**

* Cliquez sur **"Appliquer la Transformation"**
* L'application applique les instructions JSON sur chaque fichier Python sélectionné
* Les résultats apparaissent dans la console

**📁 Structure des Fichiers**

projet/

├── interface\_gui\_principale.py # Interface graphique principale

├── core/

│ └── transformations/ # Dossier des plugins de transformation

│ └── json-ai-plugin.json # Plugin "Transformations JSON-AI"

├── instructions/ # Dossier pour vos fichiers JSON d'ordres

│ ├── refactor\_functions.json # Exemple : refactoriser les fonctions

│ ├── add\_docstrings.json # Exemple : ajouter des docstrings

│ └── optimize\_imports.json # Exemple : optimiser les imports

└── targets/ # Dossier contenant vos fichiers Python à modifier

├── module1.py

├── module2.py

└── utils.py

**🔧 Fonctionnalités de l'Interface**

**Section Fichiers**

* **Sélectionner Fichiers** : Choix de fichiers .py individuels
* **Sélectionner Dossier** : Traitement en lot d'un dossier
* **Effacer** : Vider la sélection
* **JSON** : Charger un fichier d'instructions JSON

**Section Transformations**

* **Menu déroulant** : Liste des moteurs/plugins disponibles
* **Sélection requise** : Doit choisir "Transformations JSON-AI" pour utiliser les JSON

**Console de Logs**

* **Messages en temps réel** : Progression des transformations
* **Erreurs et succès** : Feedback détaillé pour chaque fichier
* **Historique complet** : Traçabilité des opérations

**💡 Exemple de Flux de Travail**

1. **Préparation** :
2. targets/my\_script.py # Fichier Python à modifier

instructions/add\_logs.json # Instructions JSON

1. **Dans l'interface** :
   * Sélectionner targets/my\_script.py
   * Choisir "Transformations JSON-AI" dans la liste
   * Charger instructions/add\_logs.json
   * Cliquer "Appliquer la Transformation"
2. **Résultat** :

targets/my\_script\_transformed.py # Fichier modifié selon les instructions JSON

**📋 Format des Fichiers JSON d'Instructions**

json

{

"name": "add\_logging",

"display\_name": "Ajouter Logging",

"description": "Ajoute des logs automatiquement dans les fonctions",

"version": "1.0.0",

"instructions": {

"add\_imports": ["import logging"],

"add\_to\_functions": {

"start": "logging.info(f'Début de {function\_name}')",

"end": "logging.info(f'Fin de {function\_name}')"

}

}

}

**⚡ Avantages du Système**

* **Séparation claire** : Code Python (cibles) ≠ Instructions (JSON)
* **Réutilisabilité** : Un JSON peut s'appliquer à plusieurs fichiers Python
* **Flexibilité** : Changez les instructions sans modifier l'application
* **Sécurité** : Prévisualisation avant application des modifications
* **Extensibilité** : Ajout facile de nouveaux plugins de transformation

**🛠️ Installation et Lancement**

bash

*# Cloner le projet*

git clone <url-du-repo>

cd interface-ast-transformations

*# Lancer l'interface*

python interface\_gui\_principale.py

**🔍 Dépannage**

* **"0 transformation(s) trouvée(s)"** : Vérifiez le dossier core/transformations/
* **"JSON invalide"** : Vérifiez la structure de votre fichier JSON d'instructions
* **"Aucun fichier sélectionné"** : Sélectionnez des fichiers Python avant d'appliquer

**🎓 Cas d'Usage Typiques**

* **Refactoring automatisé** : Renommer variables, fonctions selon des règles
* **Ajout de documentation** : Insérer docstrings, commentaires automatiquement
* **Optimisation de code** : Améliorer imports, structure selon bonnes pratiques
* **Migration de versions** : Adapter code pour nouvelles versions Python/libraries
* **Standards d'équipe** : Appliquer conventions de codage uniformes

**Interface AST** - Transformez votre code Python avec précision et flexibilité ! 🐍✨