TP 3 : Suite de tests unitaires

Objectif

- Accélérer la création de tests unitaires grâce aux suggestions de Copilot.
- Mettre en place des tests auto-générés pour un projet existant.
- Vérifier, ajuster et valider les tests générés.
- Assurer une couverture minimale des fonctions clés.
- Détecter erreurs/anomalies et réitérer pour améliorer les tests.

Étapes

1. **Préparer le projet :**
    - Reprendre le dossier tp2_task_manager.
    - Créer un fichier test_app.py.
2. **Générer des tests unitaires :**

Cibler : la classe TaskService (dans app.py).

Couvrir : add_task, list_tasks, toggle_task, delete_task, et la gestion d'erreurs (id introuvable).

- **Prompt :**

```
# Générer des tests unitaires avec pytest pour les méthodes de TaskService
# Méthodes : add_task, toggle_task, delete_task
# Utiliser des assertions claires et des noms de tests explicites
# Ajouter les cas d'erreurs : id introuvable -> exception
```

- **Action :** relire les tests proposés, ajuster noms et assertions pour clarté.

3. **Ajouter des mocks :**

L'objectif est d'**isoler la logique métier** des I/O disque. On empêche toute écriture réelle dans tasks.json en simulant load_tasks et save_tasks.

- **Prompt :**

```
# Utiliser des mocks pour simuler le stockage JSON
# Empêcher l'écriture réelle dans le fichier tasks.json
```

4. **Exécuter et analyser :**
    - Commande : **py -m pytest -q**
    - Observer les résultats : erreurs, assertions échouées, comportements inattendus.
5. **Boucle de feedback (itération explicite)**
    - **Étape 1 :** Identifier les erreurs ou anomalies.
    - **Étape 2 :** Reformuler le prompt pour guider Copilot vers un meilleur test.

      Exemple :

```
# Générer un test unitaire pour toggle_task avec un id invalide
# Attendu : lever une exception ValueError
```

    - **Étape 3 :** Regénérer le test avec Copilot.
    - **Étape 4 :** Réexécuter pytest -q.
    - Répéter jusqu'à ce que tous les tests passent.
6. **Bonus : test d'intégration**

Le test d'intégration vérifie l'enchaînement "ajout → toggle → delete" en simulant le stockage, mais en passant par les méthodes publiques.

- **Prompt :**

```
# Écrire un test d'intégration pour vérifier le cycle complet :
# ajout -> toggle -> delete
```

Validation finale

- **Fonctionnel:** Tous les tests unitaires et le test d'intégration passent.
- **Isolation:** Aucune écriture réelle dans tasks.json (mocks/monkeypatch actifs).
- **Clarté:** Noms de tests explicites, assertions lisibles, fixture documentée.
- **Robustesse:** Gestion d'erreurs cohérente (id introuvable → exception), appels à save_tasks systématiques après modifications.