Plan de mise en oeuvre de tests

PINEL Antony

Préparation du plan de test :

- **Objectif** : Valider la fonctionnalité du script `process_user_data` en testant différentes situations et en s'assurant que les sorties sont correctes.
- **Outils** : Python, Pandas, pytest, fichiers CSV pour les données d'entrée et de sortie.

Environnement de test :

- **Configuration requise** : Python 3.x, Pandas, tqdm, pytest.
- **Fichiers d'entrée** : Plusieurs fichiers CSV avec diverses configurations de données.
- **Fichiers de sortie** : Fichiers CSV générés par le script.

Stratégie de test :

Notre stratégie de test s'appuie sur plusieurs principes fondamentaux pour garantir la qualité et la fiabilité du script.

1. **Approche Boîte Noire et Boîte Blanche** :
    - **Boîte Noire** : Tester les fonctionnalités sans connaissance de l'implémentation interne. Cela garantit que toutes les fonctionnalités sont testées de manière exhaustive.
    - **Boîte Blanche** : Comprendre l'implémentation pour concevoir des tests qui couvrent tous les chemins possibles dans le code.
2. **Minimisation des Risques de Faux-Positifs et Faux-Négatifs** :
    - **Faux-Positifs** : Résultat du test indiqué comme correct alors qu'il y a un bug. Cela pourrait donner une fausse assurance de la fiabilité du script.
    - **Faux-Négatifs** : Résultat du test indiqué comme incorrect alors qu'il n'y a pas de bug. Cela pourrait entraîner des corrections inutiles.
3. **Pour minimiser ces risques, nous avons mis en place :**
    - Des tests unitaires rigoureux pour chaque fonction.
    - Des validations croisées avec des jeux de données variés.
    - Une documentation claire pour chaque test, précisant les conditions initiales, les étapes et les résultats attendus.
4. **Automatisation des Tests** :
    - Utilisation de `pytest` pour automatiser les tests et réduire les erreurs humaines.

Cas de dest :

1. **Données de Base**
    - **Description** : Vérifier le calcul de base des séries et des vies.
    - **Entrée** : `basic_data.csv`
    - **Attendu** : Calcul correct des séries et des vies pour des enregistrements simples.
2. **Pratique Consécutive**
    - **Description** : Vérifier l'incrémentation des séries pour des pratiques consécutives.
    - **Entrée** : `consecutive_practice.csv`
    - **Attendu** : Séries correctement incrémentées pour des jours consécutifs.
3. **Jours Non Consécutifs**
    - **Description** : Vérifier la réinitialisation des séries pour des pratiques non consécutives.
    - **Entrée** : `non_consecutive_practice.csv`
    - **Attendu** : Séries réinitialisées pour les jours non consécutifs.
4. **Vies Maximales**
    - **Description** : Vérifier que les vies n'excèdent pas le maximum de 2.
    - **Entrée** : `max_lives.csv`
    - **Attendu** : Vies ne dépassant pas 2 après chaque 5ème série.
5. **Perte de Vies**
    - **Description** : Vérifier la diminution des vies en cas de non-pratique et réinitialisation des séries.
    - **Entrée** : `lose_lives.csv`
    - **Attendu** : Diminution correcte des vies et réinitialisation des séries lorsque les vies sont à zéro.
6. **Format de la Date**
    - **Description** : Vérifier la conversion correcte des dates et la gestion du fuseau horaire GMT+2.
    - **Entrée** : `date_format.csv`
    - **Attendu** : Dates correctement converties et triées.
7. **Gestion des Colonnes**
    - **Description**: Vérifier la présence et la gestion correcte des colonnes dans le fichier de sortie.
    - **Entrée**: `column_handling.csv`
    - **Attendu**: Colonnes présentes et bien formatées.

8. **Mélange de Pratiques**
    - **Description** : Vérifier le calcul des séries et des vies avec un mélange de pratiques valides et invalides.
    - **Entrée** : `mixed_practice.csv`
    - **Attendu** : Calcul correct des séries et des vies en tenant compte des pratiques valides et invalides.
9. **Pratique Incomplète**
    - **Description** : Vérifier le traitement des jours avec des pratiques partiellement complètes.
    - **Entrée** : `partial_practice.csv`
    - **Attendu** : Calcul correct des séries et des vies pour des jours avec des pratiques partiellement complètes.
10. **Plusieurs Utilisateurs**
    - **Description** : Vérifier le calcul correct des séries et des vies pour plusieurs utilisateurs.
    - **Entrée** : `multiple_users.csv`
    - **Attendu** : Calcul correct des séries et des vies pour chaque utilisateur.
11. **Grandes Données**
    - **Description** : Vérifier la performance du script avec un grand ensemble de données.
    - **Entrée** : `large_data.csv`
    - **Attendu** : Calcul correct des séries et des vies dans un délai raisonnable.
12. **Dates Identiques**
    - **Description** : Vérifier le traitement des enregistrements avec des dates identiques.
    - **Entrée** : `same_dates.csv`
    - **Attendu** : Calcul correct des séries et des vies pour des enregistrements avec des dates identiques.
13. **Données Manquantes**
    - **Description** : Vérifier le traitement des enregistrements avec des données manquantes.
    - **Entrée** : `missing_data.csv`
    - **Attendu** : Gestion correcte des enregistrements avec des données manquantes.

14. **Divers Fuseaux Horaires**
    - **Description** : Vérifier la conversion correcte des dates pour divers fuseaux horaires.
    - **Entrée** : `various_timezones.csv`
    - **Attendu** : Dates correctement converties et triées pour différents fuseaux horaires.
15. **Formats de date variés**
    - **Description**: Vérifier la gestion des formats de date variés dans le fichier d'entrée.
    - **Entrée** : `varied_date_formats.csv`
    - **Attendu** : Conversion correcte des dates à partir de formats variés.

Exécution des tests :

- Utiliser `pytest` pour automatiser les tests et valider les résultats.
- Écrire des tests unitaires pour chaque fonction utilisée dans le script.

Rapport de test :

- **Résultats attendus** : Chaque test doit passer sans erreurs.
- **Documentation** : Documenter les résultats de chaque cas de test, y compris les fichiers de sortie comparés aux résultats attendus.