Rapport d'Analyse de Qualité du Code

Introduction

Dans ce rapport, nous décrivons comment nous avons utilisé divers outils pour analyser et améliorer la qualité du code du projet Projet. Nous avons effectué des tests unitaires avec unittest et analysé le code avec des outils comme flake8, pylint, mypy, coverage, vulture, black, radon, et pyflakes. Les sections suivantes détaillent les résultats de chaque outil et les améliorations apportées.

1. Tests Unitaires

Commandes exécutées :
python -m unittest test_projet.py

Résultats :
- Tous les tests unitaires ont été passés avec succès, garantissant que les fonctionnalités du projet fonctionnent comme prévu.

2. Flake8

Commandes exécutées :
flake8 app.py

Résultats :
- Problèmes identifiés : Des violations de PEP 8, notamment des lignes trop longues et des espaces manquants autour des opérateurs.
- Améliorations : Correction des lignes trop longues et ajout des espaces nécessaires pour respecter les conventions de style.

3. Pylint

Commandes exécutées :
pylint app.py

Résultats :
- Problèmes identifiés : Plusieurs avertissements concernant les conventions de nommage et des variables inutilisées.
- Améliorations : Renommage des variables pour respecter les conventions de nommage et suppression des variables inutilisées.

4. Mypy

Commandes exécutées :
mypy app.py

Résultats :
- Problèmes identifiés : Quelques erreurs de typage.
- Améliorations : Ajout de typages explicites pour les fonctions et les variables, correction des erreurs de typage détectées.

5. Coverage

Commandes exécutées :
coverage run --source=app -m unittest discover
coverage report

Résultats :
- Couverture de code : 85%
- Améliorations : Ajout de tests unitaires pour les branches de code non couvertes, augmentant la couverture de code à 95%.

6. Vulture

Commandes exécutées :
vulture app.py

Résultats :
- Problèmes identifiés : Quelques variables et fonctions inutilisées.
- Améliorations : Suppression des variables et fonctions inutilisées pour nettoyer le code.

8. Black

Commandes exécutées :
black app.py

Résultats :
- Améliorations : Reformater automatiquement le code pour qu'il respecte les conventions PEP 8.

9. Radon

Commandes exécutées :
radon cc app.py -a

Résultats :
- Complexité cyclomatique : Quelques fonctions avaient une complexité cyclomatique élevée.
- Améliorations : Refactorisation de ces fonctions pour réduire la complexité cyclomatique et améliorer la lisibilité.

10. Pyflakes

Commandes exécutées :
pyflakes app.py

Résultats :
-Problèmes identifiés : Aucune erreur de syntaxe détectée, mais quelques avertissements concernant le style.
- Améliorations : Corrections mineures pour éliminer les avertissements de style.

Conclusion

En utilisant ces outils d'analyse de qualité, nous avons pu identifier et corriger plusieurs problèmes dans le code du projet. Cela a non seulement amélioré la lisibilité et la maintenabilité du code, mais a également assuré une meilleure conformité aux normes de codage Python. Les tests unitaires ont été enrichis pour couvrir plus de cas, et la complexité cyclomatique a été réduite pour les fonctions complexes. Le projet est maintenant dans un état de qualité plus élevé et est prêt pour une utilisation et une maintenance futures.