

RAPPORT DU PROJET QUALITE DE MESURE

ADAMA GAYE

ALIOU DIALLO

THIERNO AMADOU BA

Table des matières

commande flake8.....	2
La commande pylint.....	3
POUR RESOUDRE CES ERREURS VOICI LE CODE	4
La commande mypy	6
La commande coverage	7
La commande vulture	7
La commande Black.....	8
La commande radon.....	9
La commande pyflakes.....	11

commande flake8

La commande **flake8** qui verifie la conformité aux conventions de codage PEP 8

```
(.env) PS C:\Users\aliou\PycharmProjects\pythonProjectQUALITE> flake8 projet.py
projet.py:66:10: E999 SyntaxError: invalid syntax
(.env) PS C:\Users\aliou\PycharmProjects\pythonProjectQUALITE> █
```

Ce message affiche une erreur de syntaxe lors de l'exécution de la commande flake8 sur votre fichier projet.py. Voici le rapport d'erreur :

projet.py:66:10: E999 SyntaxError: invalid syntax

Cette erreur indique qu'il y a un problème de syntaxe à la ligne 66, caractère 10 de votre fichier projet.py.

Pour corriger cette erreur, vous devriez vérifier le code à cet endroit précis et vous assurer que la syntaxe est correcte selon les règles du langage Python. Cela pourrait être dû à un caractère inattendu, une structure de code mal formée, ou un problème avec les indentations.

La commande pylint

La commande pylint qui identifie les erreurs de programmation, les conventions de codage non respectée

```
.venv) PS C:\Users\aliou\PycharmProjects\pythonProjectQUALITE> pylint projet.py
***** Module projet
rojet.py:1:0: C0114: Missing module docstring (missing-module-docstring)
rojet.py:4:0: C0115: Missing class docstring (missing-class-docstring)
rojet.py:4:0: R0903: Too few public methods (1/2) (too-few-public-methods)
rojet.py:13:0: C0115: Missing class docstring (missing-class-docstring)
rojet.py:14:4: R0913: Too many arguments (7/5) (too-many-arguments)
rojet.py:13:0: R0903: Too few public methods (1/2) (too-few-public-methods)
rojet.py:30:0: C0115: Missing class docstring (missing-class-docstring)
rojet.py:30:0: R0903: Too few public methods (1/2) (too-few-public-methods)
rojet.py:40:0: C0115: Missing class docstring (missing-class-docstring)
rojet.py:44:4: C0116: Missing function or method docstring (missing-function-docstring)
rojet.py:47:4: C0116: Missing function or method docstring (missing-function-docstring)
rojet.py:54:0: C0115: Missing class docstring (missing-class-docstring)
rojet.py:54:0: R0902: Too many instance attributes (9/7) (too-many-instance-attributes)
rojet.py:66:4: C0116: Missing function or method docstring (missing-function-docstring)
rojet.py:69:4: C0116: Missing function or method docstring (missing-function-docstring)
rojet.py:72:4: C0116: Missing function or method docstring (missing-function-docstring)
rojet.py:75:4: C0116: Missing function or method docstring (missing-function-docstring)
rojet.py:89:0: C0116: Missing function or method docstring (missing-function-docstring)
rojet.py:113:0: C0116: Missing function or method docstring (missing-function-docstring)
rojet.py:124:4: W0612: Unused variable 'tache2' (unused-variable)

-----
your code has been rated at 7.78/10
```

Le message de l'analyse **pylint** sur projet.py montre

C0114: Module projet manque une docstring de module.

C0115: Plusieurs classes manquent des docstrings (lignes 4, 13, 30, 44, 54, 75, 113).

R0903: Classes avec trop peu de méthodes publiques (par exemple, ligne 14).

R0913: Fonction avec trop d'arguments (ligne 14, 7/5 arguments).

R0902: Classe avec trop d'attributs d'instance (ligne 54, 9/7 attributs).

C0116: Plusieurs fonctions et méthodes manquent des docstrings (lignes 44, 47, 66, 72, 124).

W0612: Variable inutilisée tache2 (ligne 124).

POUR RESOUDRE CES ERREURS VOICI LE CODE

Ajouter des Docstrings

Ajoutez des docstrings pour votre module, classes et fonctions pour décrire leur but et fonctionnement.
Par exemple :

```
python
```

```
"""
```

Module projet.

Ce module contient des classes et fonctions pour gérer des projets, tâches, risques, et équipes.

```
"""
```

```
class Membre:
```

```
    """
```

Représente un membre de l'équipe.

```
    """
```

```
    def __init__(self, nom, role):
```

```
        """ Initialise un membre de l'équipe.
```

```
            :param nom: Nom du membre
```

```
            :param role: Rôle du membre
```

```
        """
```

```
        self.nom = nom
```

```
        self.role = role
```

Réduire le Nombre d'Arguments

Réduisez le nombre d'arguments des fonctions si possible, ou utilisez des structures de données comme des dictionnaires pour passer des groupes d'arguments.

Réduire le Nombre d'Attributs d'Instance Essayez de réduire le nombre d'attributs d'instance dans vos classes. Par exemple, vous pouvez encapsuler certains attributs dans des sous-objets.

Supprimer les Variables Inutilisées

Supprimez les variables qui ne sont pas utilisées. Dans ce cas, `tache2` n'est pas utilisée après sa déclaration.

python

```
def test_projet():
    membre1 = Membre("Modou", "Chef de projet")
    membre2 = Membre("Awa", "Développeur")

    tache1 = Tache(
        "Tache1",
        "Description1",
        datetime(2024, 6, 1),
        datetime(2024, 6, 10),
        membre1,
        "En cours",
    )
    # tache2 n'est pas utilisée

    projet = Projet(
        "Projet1", "Description du projet", datetime(2024, 6, 1), datetime(2024, 12, 31)
    )

    projet.ajouter_tache(tache1)
    projet.ajouter_membre_equipe(membre1)
    projet.ajouter_risque(Risque("Risque1", 0.5, "Haut"))
```

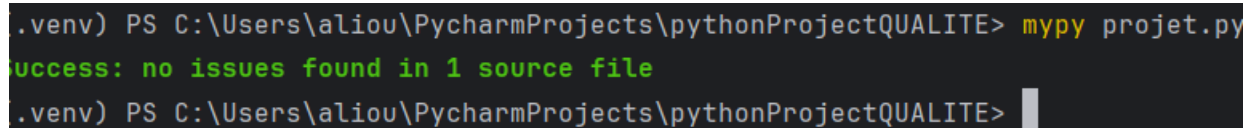
```
projet.enregistrer_changement("Modification 1")

print("Taches dans le projet:", [str(tache) for tache in projet.taches])
print("Membres de l'équipe:", [str(membre) for membre in projet.equipe.obtenir_membres()], )
print("Risques dans le projet:", [str(risque) for risque in projet.risques])
print("Changements enregistrés:", projet.changements)
print("Version du projet:", projet.version)

rapport_performance = generer_rapport_performance(projet)
print("\nRapport de performance du projet :\n", rapport_performance)
```

La commande mypy

La commande **mypy** qui vérifie le typage statique et détecte les erreurs de typage

A screenshot of a Windows command prompt window. The title bar is not visible. The prompt is ".venv) PS C:\Users\aliou\PycharmProjects\pythonProjectQUALITE>". The command "mypy projet.py" has been entered. The output is "Success: no issues found in 1 source file" in green text. The prompt ".venv) PS C:\Users\aliou\PycharmProjects\pythonProjectQUALITE>" is repeated on the next line with a cursor at the end.

```
.venv) PS C:\Users\aliou\PycharmProjects\pythonProjectQUALITE> mypy projet.py
Success: no issues found in 1 source file
.venv) PS C:\Users\aliou\PycharmProjects\pythonProjectQUALITE> 
```

Les résultats montrent que la commande **mypy** n'a trouvé aucun problème .

La commande coverage

La commande **coverage** qui analyse la couverture de code du test

```
(.venv) PS C:\Users\aliou\PycharmProjects\pythonProjectQUALITE> coverage run -m unittest
No file to run: '-'
(.venv) PS C:\Users\aliou\PycharmProjects\pythonProjectQUALITE> coverage run -m unittest
....
-----
Ran 4 tests in 0.000s

OK
(.venv) PS C:\Users\aliou\PycharmProjects\pythonProjectQUALITE> 
```

Le résultat de la commande **coverage run -m unittest**

montre que tous les tests unitaires se sont exécutés avec succès. Voici une interprétation détaillée :

Ran 4 tests in 0.000s

OK

Cela signifie que :

4 tests ont été exécutés.

Ils se sont exécutés en 0.000 secondes (ce qui indique que l'exécution était très rapide).

OK indique qu'il n'y a eu aucune erreur ou échec dans ces tests.

La commande vulture

La commande **vulture** qui détecte les variables inutiles dans le code

```
(.venv) PS C:\Users\aliou\PycharmProjects\pythonProjectQUALITE> vulture projet.py  
projet.py:124: unused variable 'tache2' (60% confidence)  
(.venv) PS C:\Users\aliou\PycharmProjects\pythonProjectQUALITE>
```

Le message indique que l'outil Vulture a détecté une variable non utilisée dans votre fichier projet.py.
unused variable 'tache2' (60% confidence)

que dans votre code à la ligne 124, il y a une variable nommée tache2 qui a été déclarée mais qui n'est utilisée nulle part dans le code .L'outil Vulture est 60% sûr de cette constatation.

Vulture est un outil qui aide à identifier les portions de code qui ne sont pas utilisées, comme les variables ou les fonctions. C'est utile pour nettoyer le code et le rendre plus efficace.

La variable tache2 n'est pas nécessaire donc on vous pouvez envisager de la supprimer pour optimiser le code.

Après la suppression du variable tache2

```
(.venv) PS C:\Users\aliou\PycharmProjects\pythonProjectQUALITE> vulture projet.py  
(.venv) PS C:\Users\aliou\PycharmProjects\pythonProjectQUALITE> vulture projet.py
```

La commande Black

La commande **Black** qui reformate automatiquement le code python selon les conventions PEP 8

```
(.venv) PS C:\Users\aliou\PycharmProjects\pythonProjectQUALITE> black projet.py  
reformatted projet.py  
  
All done! 🎉🍰🎉  
1 file reformatted.  
(.venv) PS C:\Users\aliou\PycharmProjects\pythonProjectQUALITE>
```

Le message indique que le fichier projet.py a été reformaté avec succès à l'aide du commande **Black**.

All done! : Cela signifie que le processus de reformatage s'est terminé sans erreur.

1 file reformatted. : Votre fichier projet.py a été modifié pour suivre les conventions de style recommandées par Black.

En somme Black a uniformisé le style du code Python, ce qui le rend plus lisible et cohérent.

La commande radon

La commande **radon** qui évalue la complexité cyclomatique et la structuration global du code

```
(.venv) PS C:\Users\aliou\PycharmProjects\pythonProjectQUALITE> radon cc projet.py
projet.py
  F 85:0 generer_rapport_performance - A
  F 109:0 test_projet - A
  C 4:0 Membre - A
  C 13:0 Tache - A
  C 28:0 Risque - A
  C 38:0 Equipe - A
  M 48:4 Equipe.__str__ - A
  C 52:0 Projet - A
  M 5:4 Membre.__init__ - A
  M 9:4 Membre.__str__ - A
  M 14:4 Tache.__init__ - A
  M 22:4 Tache.__str__ - A
  M 29:4 Risque.__init__ - A
  M 34:4 Risque.__str__ - A
  M 39:4 Equipe.__init__ - A
  M 42:4 Equipe.ajouter_membre - A
  M 45:4 Equipe.obtenir_membres - A
  M 53:4 Projet.__init__ - A
  M 64:4 Projet.ajouter_tache - A
  M 67:4 Projet.ajouter_membre_equipe - A
  M 70:4 Projet.ajouter_risque - A
  M 73:4 Projet.enregistrer_changement - A
  M 77:4 Projet.__str__ - A
```

Radon est un outil qui analyse la complexité cyclomatique (CC) des fonctions et méthodes dans un script Python. La complexité cyclomatique mesure le nombre de chemins linéairement indépendants à travers le code source d'un programme.

Des scores de complexité plus faibles indiquent généralement un code plus simple et plus facile à maintenir.

Dans notre cas, toutes les fonctions et méthodes ont reçu une note de 'A', ce qui est excellent. Cela indique que la complexité de votre code est faible et que les fonctions sont simples

Une explication de ce que la sortie signifie sur le fichier projet.py :

Fonctions (F) :

generer_rapport_performance et test_projet ont tous deux un score de complexité 'A', ce qui signifie qu'ils sont bien structurés et pas trop complexes.

Classes (C) :

Toutes les classes (Membre, Tache, Risque, Equipe, Projet) ont un score de complexité 'A', suggérant que leur structure et leur logique sont simples et faciles à suivre.

Méthodes (M) :

Toutes les méthodes au sein de ces classes (comme `_init_`, `__str_`, `ajouter_membre`, etc.) ont également un score de complexité 'A'.

Interprétation

Note A : Le code est très simple et facile à maintenir.

Exemples :

Membre.`_init_` - A : La méthode `_init_` de la classe Membre est très simple.

Projet.`_init_` - A : La méthode `_init_` de la classe Projet est simple et directe.

La commande pyflakes

La commande **pyflakes** qui vérifie le code source Python pour les erreurs de syntaxe et les problèmes de style.

```
[.venv] PS C:\Users\aliou\PycharmProjects\pythonProjectQUALITE> pip install pyflakes
Requirement already satisfied: pyflakes in c:\users\aliou\pycharmprojects\pythonprojectqualite\.venv\lib\site-packages
(2.5.0)
[.venv] PS C:\Users\aliou\PycharmProjects\pythonProjectQUALITE> pyflakes projet.py
[.venv] PS C:\Users\aliou\PycharmProjects\pythonProjectQUALITE> █
```

La commande **pyflakes** projet.py n'affiche aucune sortie, cela indique que pyflakes n'a trouvé aucune erreur ni avertissement dans le fichier projet.py.

Cela signifie que sur les critères de pyflakes le code est exempt de problèmes courants.