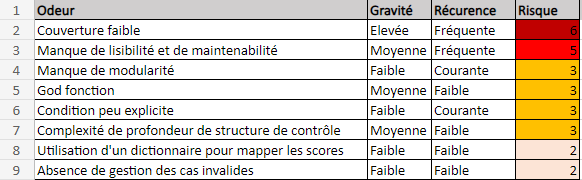
JOURNAL DE BORD

La première étape a été de se concerter en équipe pour décider quelle branche nous allons prendre. Nous avons fait le choix de partir sur le fichier python1.py car l’analyse de ce fichier était la plus poussée.



A travers cette analyse, nous nous sommes rendus compte que les deux problèmes principaux sont la couverture de tests qui est inexistante et l’illisibilité du code. Nous avons donc décidé de nous concentrer sur ces deux points en créant des golden master pour l’existant et des tests fonctionnels pour l’implémentation de la nouvelle fonctionnalité.

**Etapes du refactoring :**

Nous avons choisi d’effectuer le refactoring avant l’implémentation de la nouvelle fonctionnalité pour nous permettre de gagner en lisibilité en en compréhension. Dans ce code, certaines méthodes sont trop complexes tandis que les noms des variables ne sont pas claires, c’est donc pour cela que nous avons pris la décision de commencer par l’adoption d’une convention de nommage pour plus de clarté, et ce n’est qu’après que nous avons démarré la modification du code en lui-même.

Afin de nous assurer que nos modifications fonctionnent nous avons aussi mis en place des tests unitaires.

1. Améliorer les conventions de nommage :

Convention adoptée : Snake case.

Renommer ‘p1points’ et ‘p2points’ en ‘player1\_score’ et ‘player2\_score’

Les noms explicites améliorent la lisibilité du code. Cette modification est facile à faire grâce aux IDE et / ou éditeurs de texte.

1. Décomposition de la méthode score :

Découpage de la méthode ‘score’ afin de gagner en modularité.

Cette ‘God Method’ rend le code difficile à lire, à tester et à maintenir. En décomposant tout ça, on vise à respecter le principe de responsabilité unique. Cela va aussi faciliter le test unitaire de chaque comportement de calcul du score de manière isolée.

1. Simplification de la logique :

Ici on va chercher à supprimer certaines conditions afin d’améliorer la compréhension du code.

Nous avons défini les valeurs des scores en tant que constantes que nous avons stockées dans un tableau afin de gagner en lisibilité et de ne plus avoir plusieurs dictionnaires. Nous gardons donc une seule source de vérité pour ces informations.

1. Mise en place de différents tests:

Golden master :

Avant de démarrer le refactoring, nous avons mis en place des tests unitaires afin de nous assurer que nous ne « cassons » rien lorsque nous apportons des modifications au code.

Tests fonctionnels :

**Ajout de la fonctionnalité de Traduction :**

Afin de gérer la traduction français/anglais, nous avons créé 2 fichiers JSON (FR.json, EN.json) dans lesquels nous avons des paires clé valeur qui contiennent les mots de chaque langue.