Simulation d'itération

Journal de bord

Temps alloué : 4 heures
Objectifs :

- Implémenter une fonctionnalité supplémentaire.
- Assainir le code existant.

Étape 1 : Planification (30 minutes)

Analyse des priorités :

1) Typage des noms des joueurs : essentiel pour éviter des erreurs de type inattendues.
2) Simplification de la condition : améliore la lisibilité sans changer la logique.
3) Noms de variables descriptives : rend le code plus compréhensible.
4) Utilisation de constantes : évite les valeurs magiques et rend le code plus maintenable.
5) Usage cohérent de structures de contrôle : favorise la compréhension pour tous les développeurs.

Choix de la fonctionnalité supplémentaire :

Ajout d'une fonctionnalité pour suivre le nombre total de points gagnés par chaque joueur tout au long du jeu.

Étape 2 : Implémentation (3 heures)

1. Typage des noms des joueurs (30 minutes)

Ajout d'une vérification pour s'assurer que les noms des joueurs sont des chaînes de caractères non vides.

C'est un élément fondamental pour assurer la robustesse du code. Si les noms ne sont pas des chaînes de caractères, le reste de la logique peut échouer de manière imprévisible.

2. Simplification de la condition (20 minutes)

Remplacement de la condition playerName == "player1" par if playerName == self.player1Name.

Une condition simplifiée rend le code plus lisible et facile à comprendre, réduisant le risque d'erreurs lors de la maintenance.

3. Noms de variables descriptifs (35 minutes)

Renommage des variables endGameScore et regularScore en finalScore et currentScore.

Des noms de variables descriptifs facilitent la compréhension du code, surtout pour ceux qui le lisent pour la première fois.

4. Utilisation de constantes (35 minutes)

Création de constantes pour expliquer la signification des valeurs numériques magiques.

Les valeurs magiques dans le code peuvent être déroutantes. Les constantes rendent le code plus compréhensible et maintenable.

5. Usage cohérent de structures de contrôle (35 minutes)

Remplacement de match-case par des structures if-elif-else.

L'utilisation de structures de contrôle classiques améliore la lisibilité pour les développeurs qui ne sont pas familiers avec la syntaxe `match-case`.

6. Fonctionnalité supplémentaire : suivi des points totaux (1 heure)

- Ajout d'attributs pour suivre le nombre total de points gagnés par chaque joueur.
- Mise à jour de la méthode won_point pour incrémenter ces attributs.
- Ajout d'une méthode pour afficher les points totaux.

Cette fonctionnalité supplémentaire améliore la visibilité des performances des joueurs tout au long du jeu, ce qui peut être utile pour des analyses post-match.