Simulation d'itération

Journal de bord

Temps alloué : 4 heures
Objectifs :

- Valider les changements apportés.
- Tester la nouvelle fonctionnalité.
- Documenter les résultats et les validations.

Étape 1 : Validation des changements apportés

1.1 Typage des noms des joueurs

Description : Les noms des joueurs doivent être des chaînes de caractères non vides.

Action : Vérification de la validité des noms lors de l'initialisation.

Test : Initialisation de la classe avec des noms valides et invalides pour vérifier que la vérification fonctionne correctement.

Résultat : Les tests ont réussi. La validation des noms de joueurs fonctionne comme prévu.

1.2 Simplification de la condition

Description : La condition de mise à jour des points du joueur peut être simplifiée.

Action : Simplification de la condition pour utiliser `if playerName == self.player1Name.`

Test : Simuler des points gagnés par les joueurs pour s'assurer que les scores sont correctement mis à jour.

Résultat : Les tests ont réussi. La mise à jour des scores est correcte.

1.3 Noms de variables descriptifs

Description : Les noms de variables doivent être plus explicites.

Action : Renommer les variables `endGameScore` et `regularScore` en `finalScore` et `currentScore`.

Test : Vérifier que les nouvelles variables sont utilisées de manière cohérente dans le code.

Résultat : Les tests ont réussi. Les nouvelles variables sont utilisées de manière cohérente.

1.4 Utilisation de constantes

Description : Remplacer les valeurs magiques par des constantes.

Action : Créer des constantes pour les valeurs numériques (par exemple, 4).

Test : Confirmer que les constantes sont utilisées correctement pour remplacer les valeurs magiques.

Résultat : Les tests ont réussi. Les constantes sont utilisées correctement.

1.5 Usage cohérent de structures de contrôle

Description : Remplacer `match-case` par des structures `if-elif-else` pour une meilleure cohérence et lisibilité.

Action : Utiliser `if-elif-else` au lieu de `match-case`.

Test : Vérifier que les structures `if-elif-else` fonctionnent comme prévu.

Résultat : Les tests ont réussi. Les structures de contrôle sont cohérentes et lisibles.

1.6 Suivi des points totaux

Description : Ajouter une fonctionnalité pour suivre le nombre total de points gagnés par chaque joueur.

Action : Ajouter des attributs pour suivre les points totaux et mettre à jour la méthode `won_point`.

Test : Simuler plusieurs points gagnés par les joueurs et vérifier que le suivi des points totaux fonctionne correctement.

Résultat : Les tests ont réussi. Le suivi des points totaux fonctionne correctement.

Étape 2 : Tests unitaires

Objectif : Valider les changements apportés avec des tests unitaires.
Action : Créer un fichier de test `test_tennis_game6.py` avec les tests unitaires (voir repo git).

Étape 3 : Documentation des résultats et des validations

Résultats des tests unitaires

- **Résumé** : Tous les tests ont réussi, confirmant que les modifications et les nouvelles fonctionnalités fonctionnent comme prévu.

Conclusion des tests

- **Validations réussies** :
  - Typage des noms des joueurs.
  - Simplification de la condition.
  - Utilisation de noms de variables descriptifs.
  - Utilisation de constantes.
  - Structures de contrôle cohérentes.
  - Suivi des points totaux.

Problème	Description	Recommandation	Validation	Résultat des Tests
Typage des noms des joueurs	Les noms des joueurs peuvent être de tout type, ce qui peut causer des erreurs imprévues.	Vérifier que les noms des joueurs sont des chaînes de caractères non vides avant de les utiliser. Typé les noms des joueurs.	Tests de création de joueurs avec des noms valides et invalides réussis	OK
Simplification de la condition	La condition `playerName == "player1"` est redondante et peut être simplifiée.	Remplacer la condition par `if playerName == self.player1Name.`	Tests de mise à jour des scores des joueurs réussis.	OK
Noms de variables descriptifs	Les noms de variables `endGameScore` et `regularScore` ne sont pas assez descriptifs.	Utiliser des noms de variables plus explicites, tels que `finalScore` et `currentScore`.	Code révisé et tests de scores réguliers et finaux réussis.	OK
Utilisation de constantes	Pourquoi le nombre 4 ?	Créer des constantes pour expliquer la signification des valeurs numériques magiques.	Tests de validation des scores finaux avec l'utilisation de constantes réussis.	OK
Usage incohérent de `match-case`	L'usage de `match-case` n'est pas cohérent avec le reste du code Python et peut rendre le code moins compréhensible pour ceux qui ne sont pas familiers avec cette syntaxe.	Utiliser des structures de contrôle plus classiques comme des `if-elif-else` pour une meilleure cohérence et lisibilité.	Tests de validation des scores réguliers réussis.	OK
Fonctionnalité Supplémentaire	Ajout d'une fonctionnalité pour suivre le nombre total de points gagnés par chaque joueur.	Ajout d'attributs pour suivre les points totaux et mise à jour de la méthode `won_point`.	Tests de validation des points totaux réussis.	OK