Préparation de la soutenance orale du projet Tic Tac Toe

Le temps de parole doit être équitablement réparti entre tous les membres du groupe.
La durée de la soutenance est comprise entre 5 et 10 minutes.

Présentation

Présentez les membres du groupe ainsi que votre projet avec si possible une courte démo.
Expliquez le fonctionnement général de votre projet : Comment est-il structuré ? Quelles méthodes ont été implémentées ? Dans quel but ?
Expliquez brièvement le fonctionnement de chaque classes et méthodes implémentées.

Questions :

Une fois la présentation générale effectuée, répondez à ces questions :

- Quels ont été les points difficiles du projet pour vous ? Pourquoi ?
- Combien de temps avez vous pris pour réaliser ce projet ?
- Comment vous êtes-vous réparti le travail au sein du groupe ?
- Avez vous aimé réaliser ce projet et pourquoi ?

Structure du projet et contraintes

Obligatoire :

> Pour réaliser ce jeu du Morpion vous allez devoir définir au minimum 4 classes : Joueur, Case, Grille et Jeu.

Voici des indications sur les classes à implémenter.
Si vous ne voulez pas suivre ses indications à la lettre car elles ne vous conviennent pas et vous voyez une manière différente de résoudre un problème, vous êtes libre de faire comme bon vous semble. Le tout est de pouvoir expliquer vos choix.

La classe **Joueur** :

- Un Joueur possède 2 attributs : un nom et un symbole ("X" ou "O").

- On peut instancier un Joueur grâce à un constructeur et il sera possible d'afficher un Joueur en le "convertissant" en chaine de caractère grâce à la méthode __str__

La classe **Case** :

- Une Case possède 2 attributs : une position et une valeur. La position correspond au numéro de la case (0 à 8) et la valeur correspond au symbole présent dans la case ("X", "O" ou None si la case est vide).

- On peut instancier une Case grâce à un constructeur et il sera possible d'afficher une Case en le "convertissant" en chaine de caractère grâce à la méthode __str__

La classe **Grille** :

- Une Grille possède 1 attribut : un tableau de 9 Cases.

- Il est possible d'instancier une Grille grâce à un constructeur

- Une méthode qui vérifie si une Case est vide ou non.

- Une méthode qui permet de changer la valeur d'une Case de la grille en "X" ou "O"

- Une méthode qui vérifie s'il y a un gagnant (horizontalement, verticalement ou diagonalement)

- Une méthode __str__ qui permettra d'afficher la Grille sous la forme :

```
|X|O|X|
|O|X|O|
|O|X|O|
```

(Ceci est un exemple)

La classe **Jeu** :

- Un Jeu possède 4 attributs : Une liste de 2 Joueurs, une Grille, un indice du joueur actuel (pour pouvoir récupérer le Joueur dans la liste des Joueurs) et un compteur de coups joués.

- Il est possible d'instancier un Jeu grâce à un constructeur

- Une méthode qui permet de renvoyer le Joueur correspondant à l'indice du joueur actuel.
- Une méthode qui permet de changer l'indice du joueur actuel (Si 0 passe à 1 et inversement).

- Une méthode qui gère un tour de jeu :

    1. Affiche la Grille de jeu
    2. Demande au joueur une Case où jouer
    3. Boucle tant que la case choisie n'est pas vide
    4. Joue le symbole du joueur dans la case choisie
    5. Augmente le compteur de coups

- Une méthode qui gère le jeu en entier :

    1. Boucle tant qu'il n'y a pas de gagnant ou égalité
    2. Effectue un tour de jeu
    3. Vérifie s'il y a un gagnant ou égalité et affiche le message correspondant
    4. Si non, change de joueur

L'ordre des classes est présenté de la sorte à aller du plus simple au plus difficile. Si vous êtes en difficulté pour structurer le projet, un code d'aide sera fourni.

Prolongements

Pour les groupes ayant fini le projet en avance, voici les options que vous pouvez rajouter :

- Avoir la possibilité à la fin d'une partie de rejouer une autre partie.

- Créer une nouvelle classe qui va gérer tous les affichages et demandes (input()). Il est important qu'un objet soit responsable d'effectuer des tâches qui le concerne uniquement. L'affichage et les demandes d'informations se feront par l'intermédiaire d'une autre classe qu'il sera inutile d'instancier. Les méthodes de cette classe seront alors statiques (rechercher sur le web ce que cela signifie).

- Avoir la possibilité de revenir en arrière. A chaque coup joué, le joueur doit avoir la possibilité de revenir sur son coup s'il s'est trompé. Le jeu doit reprendre son état précédant le coup du joueur.

- Faire une interface graphique avec l'outil de votre choix. Vous pouvez utiliser les bibliothèques pygame ou tkinter.

- Avoir la possibilité de jouer contre l'ordinateur. A vous de choisir les stratégies de jeu de l'ordinateur.

- Gérer les joueurs dans une base de données (comme vu en TP) et pouvoir noter les scores et tenir à jour un tableau de joueurs ayant les meilleurs résultats au jeu.

- Pouvoir changer la taille de la grille et les conditions de victoires (lignes plus longues ou schéma en particulier)

Annexes

Critères de notation :

- Les contraintes sont respectées.
- Chaque classe et méthodes sont bien définies.
- Il n'y a pas de bugs au lancement du jeu.
- Le jeu est jouable et fonctionne correctement.
- Des points bonus seront attribués en fonction des prolongements mis en place.

Attention ! Il vaut mieux un jeu qui fonctionne correctement sans prolongements plutôt qu'une mauvaise implémentation d'un prolongement provoquant des problèmes à l'exécution.

Grille évaluation oral :

	Qualité orale de l'épreuve	Qualité de la prise de parole en continu	Qualité des connaissances	Qualité de l'interaction	Qualité et construction de l'argumentation
Très insuffisant	Difficilement audible sur l'ensemble de la prestation. Le candidat ne parvient pas à capter l'attention.	Énoncés courts, ponctués de pauses et de faux démarrages ou énoncés longs à la syntaxe mal maîtrisée.	Connaissances imprécises, incapacité à répondre aux questions, même avec une aide et des relances.	Réponses courtes ou rares. La communication repose principalement sur l'évaluateur.	Pas de compréhension du sujet, discours non argumenté et décousu.
Insuffisant	La voix devient plus audible et intelligible au fil de l'épreuve mais demeure monocorde. Vocabulaire limité ou approximatif	Discours assez clair mais vocabulaire limité et énoncés schématiques.	Connaissances réelles, mais difficulté à les mobiliser en situation à l'occasion des questions du jury.	L'entretien permet une amorce d'échange. L'interaction reste limitée.	Début de démonstration mais raisonnement lacunaire. Discours insuffisamment structuré.
Satisfaisant	Quelques variations dans l'utilisation de la voix ; prise de parole affirmée. Il utilise un lexique adapté. Le candidat parvient à susciter l'intérêt.	Discours articulé et pertinent, énoncés bien construits.	Connaissances précises, une capacité à les mobiliser en réponses aux questions du jury avec éventuellement quelques relances	Répond, contribue, réagit. Se reprend, reformule en s'aidant des propositions du jury.	Démonstration construite et appuyée sur des arguments précis et pertinents.
Très satisfaisant	La voix soutient efficacement le discours. Qualités prosodiques marquées (débit, fluidité, variations et nuances pertinentes, etc.). Le candidat est pleinement engagé dans sa parole. Il utilise un vocabulaire riche et précis.	Discours fluide, efficace, tirant pleinement profit du temps et développant ses propositions.	Connaissances maîtrisées, les réponses aux questions du jury témoignent d'une capacité à mobiliser ces connaissances à bon escient et à les exposer clairement.	S'engage dans sa parole, réagit de façon pertinente. Prend l'initiative dans l'échange. Exploite judicieusement les éléments fournis par la situation d'interaction.	Maîtrise des enjeux du sujet, capacité à conduire et exprimer une argumentation personnelle, bien construite et raisonnée.