420-SN1-RE

Programmation en sciences

Travail Pratique #2

Programme / Département	200.B1 Sciences de la nature
Nom	Création d'un jeu "Tic-Tac-Toe"
Session	Hiver 2024
Professeur	Tommy Gagnon Joyal
Bureau / téléphone	C-1207
Courriel	Utiliser MIO sur Omnivox / Sur Discord
Disponibilité	Lundi de 10h40 à 15h20 et Mardi de 10h40 à 12h30 (sur rendez-vous seulement)

Description du Travail

Objectif

L’objectif du deuxième TP est de créer le jeu Tic-Tac-Toe en utilisant les concepts suivants :

- Variables et CONSTANTES
- Opérateurs booléens combinés
- Énoncés conditionnels
- Boucles (For et/ou While)
- Fonctions (pour la modularité du code)

Mise en place du jeu

Vous devez créer un programme Python qui permettra à deux joueurs de s'affronter dans une partie de Tic-Tac-Toe. Le jeu se déroulera sur une grille de 3x3 cases.

Fonctionnalités du jeu

Le premier joueur jouera avec "X" et le deuxième joueur avec "O".

Les joueurs alterneront pour placer leur symbole sur la grille jusqu'à ce qu'il y ait un gagnant ou que la grille soit pleine.

Le programme doit vérifier s'il y a un gagnant après chaque coup et annoncer le gagnant le cas échéant.

Le programme doit également vérifier s'il y a égalité lorsque la grille est pleine et annoncer le match nul.

Versions du jeu

Vous devrez programmer deux versions du jeu :

1. **Joueur contre Joueur :** Deux joueurs se succèdent pour placer leurs symboles sur la grille.
2. **Joueur contre Ordinateur :** Le joueur affronte un ordinateur qui joue de manière automatisée. L'ordinateur doit être capable de prendre des décisions logiques pour jouer contre le joueur.

Affichage du Tic-Tac-Toe

Utilisez une série de dessins ASCII (en lettre, dans la console) pour représenter les différentes étapes du Tic-Tac-Toe, c'est-à-dire les "x" et les "x" au cases correspondantes. (*Voir figure 1*).

Affichez la progression du pendu à chaque mauvais coup du joueur.

Si le joueur utilise toutes ses tentatives sans deviner correctement le mot, il perd la partie.

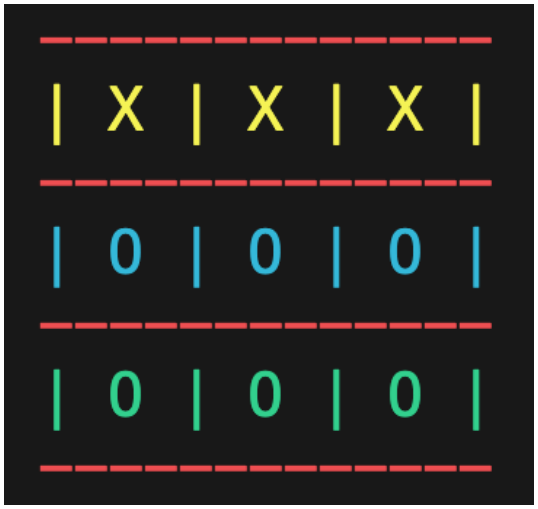

Figure 1. Exemple d'affichage du jeu en ASCII dans la console fait à partir de print().

Consignes supplémentaires

Le programme doit être bien commenté pour expliquer chaque partie du code **lorsque nécessaire** (ne pas mettre de commentaire pour les choses évidentes, voir *Figure 2*).

Utilisez des noms de variables significatifs pour rendre le code facile à comprendre.

Testez vos deux versions du programme avec différentes situations de jeu pour assurer leur bon fonctionnement.

```
# Le nombre de chance est 6
chances = 6

# Le mot secret est python
mot_secret = "python"
```

Figure 2. Exemple de commentaire inutile ou trop explicite.

Évaluation

Ce travail compte pour **20%** de la note finale et sera évalué sur **le Fonctionnement, la Qualité du code et le rapport.**

Le fonctionnement et la qualité du code sont aussi importants l'un que l'autre. Un mauvais fonctionnement pourrait pénaliser la note pour la qualité du code et une mauvaise qualité du code pourrait pénaliser la note pour le fonctionnement. Le rapport quant à lui, permet de voir si la matière a été bien assimilée. Les lignes du rapport sont à titre indicatif. Ne pas écrire de roman inutilement au péril de perdre des points. Les réponses doivent être concises, les réponses vagues ou trop longues seront pénalisées.

La remise du travail sera **Vendredi le 17 Mai avant 23h59,** et la remise se fera par Léa en remettant un fichier compressé en format zip de tous les fichiers sources (fichier python et rapport au format docx/pdf).

Grille de correction et Questionnaire

Partie Rapport (5% de la note finale)

1. **Quelles étaient les principales difficultés rencontrées lors de la programmation du jeu Tic-Tac-Toe ?**

2. **Quels étaient les choix de conception que vous avez faits lors de l'implémentation du jeu ? Pourquoi avez-vous choisi cette approche ?**

3. **Comment avez-vous géré la logique du jeu, notamment la vérification des conditions de victoire et de match nul ?**

4. **Quelles stratégies avez-vous utilisées pour programmer l'intelligence artificielle de l'ordinateur dans la version "Joueur contre Ordinateur" du jeu ?**

5. **Comment avez-vous testé votre programme pour vous assurer de son bon fonctionnement dans différentes situations de jeu ?**

6. **Comment avez-vous organisé votre code pour le rendre modulaire et facile à comprendre ?**

7. **Quelles leçons avez-vous apprises en programmant ce jeu Tic-Tac-Toe ? Quelles compétences avez-vous développées ou renforcées ?**

8. **Comment avez-vous abordé la gestion des erreurs et des exceptions dans votre programme, en particulier lors de la saisie des coordonnées par les joueurs ?**

9. **Quelles fonctions ou méthodes ont été les plus critiques dans votre programme, et comment les avez-vous testées pour assurer leur bon fonctionnement ?**

10. **Comment avez-vous utilisé les commentaires et la documentation pour faciliter la compréhension de votre code par d'autres programmeurs ou vous-même à l'avenir ?**

Partie Code (15% de la note finale)

Élément	Description	Note
Constantes et variables	Les variables et les constantes sont bien désignées, sont claires et toutes utilisées dans le programme.	5%
Opérateurs booléens combinés	Les opérateurs booléens combinés sont bien utilisés lors de la validation dans les énoncés “if”, les boucles “for” et “while” et dans les fonctions.	5%
Utilisations des boucles	L’utilisation de la boucle “for” pour itéré lors de la validation d’un mot et l’utilisation de la boucle while pour valider si le joueur est toujours dans une partie.	10%
Utilisation des fonctions	Le programme est séparé en fonction, permettant d’avoir un code beaucoup plus modulaire et lisible.	25%
Validation d'entrée	Le programme valide ce que l’utilisateur a entré permettant de jouer au jeu de façon efficace. (ex. un caractère à la fois, une position valide, etc.)	5%
Validation de fin de partie.	Le programme valide quand le tableau est rempli ou quand l’un des joueurs à gagné.	10%
Affichage du jeu	Le jeu s’affiche sous format ASCII (en lettre) afin d’avoir un retour visuel.	5%
Mode Joueur contre Joueur	Le programme doit permettre un mode Joueur contre Joueur. Le système doit être clair et permettre de comprendre s’il s’agit du tour au premier ou au second joueur de faire son choix.	10%
Mode Joueur contre Ordinateur	Le programme doit permettre un mode Joueur contre Ordinateur. Le système doit donc prendre des choix, et jouer contre vous. Si le système est trop idiot (ex. toujours faire un choix aléatoire, vous aller perdre des points dans cette section).	20%
Bonus : ASCII ++	Si vous ajoutez un module complémentaire pour mettre de la couleur ou autre ajout intéressant permettant d’avoir un jeu qui est visuellement plus attrayant qu’un simple cube avec des “X” et des “O”.	5%

*Bonus: La note maximum est fixée à 20% de la note finale, l'intérêt de faire les bonus est avant tout le plaisir, et de pallier à quelques oublis / faiblesses dans le rapport.