TP 1 – Chifoumi : Mode Tournoi et Statistiques

Objectif du TP

Concevoir une version **améliorée et modulaire** du jeu du **Chifoumi**, intégrant :

- un **mode tournoi** (plusieurs parties successives),
- une **gestion des scores cumulés**,
- et la **sauvegarde automatique** des résultats dans un fichier texte.

Ce TP a pour but de consolider les compétences sur :

- la **structuration d'un programme complet**,
- la **manipulation de fichiers**,
- et la **décomposition fonctionnelle**.

Remarque si vous souhaitez vérifier qu'un fichier existe, utilisez le code suivant :

```
import os

if os.path.exists("data/notes.txt"):
    print("Le fichier existe.")
else:
    print("Le fichier n'existe pas.")
```

Contexte

Après avoir conçu un jeu de Chifoumi simple (un seul match), l'objectif est maintenant d'ajouter des **fonctionnalités d'organisation de parties multiples**, permettant d'obtenir un **classement final** entre le joueur et l'ordinateur.

Chaque partie correspond à une série de manches (par exemple 3 ou 5), et plusieurs parties composent un **tournoi**.

Consignes

1. Le programme propose un **tournoi** composé de plusieurs parties (exemple : 3 parties de 5 manches chacune).
2. Chaque partie suit les règles du Chifoumi classique :
    - le joueur choisit `pierre`, `feuille` ou `ciseaux` ;
    - l'ordinateur choisit aléatoirement ;
    - le vainqueur de la manche est déterminé selon les règles standards.

3. Le programme affiche, à la fin de chaque partie :
    - le score du joueur et de l'ordinateur,
    - le vainqueur de la partie.
4. À la fin du tournoi, le programme affiche un **résumé global** :
    - nombre total de victoires du joueur,
    - nombre total de victoires de l'ordinateur,
    - nombre d'égalités,
    - vainqueur du tournoi.
5. Le programme enregistre automatiquement le **résultat du tournoi** dans un fichier texte `scores.txt`, sous la forme :

```
Date : 2025-10-22
Joueur : 2 victoires
Ordinateur : 1 victoire
Égalités : 0
Vainqueur du tournoi : Joueur
```

6. Le programme est organisé à l'aide de fonctions :
    - `play_match(rounds)` : gère une partie complète,
    - `play_tournament(nb_matchs, rounds)` : gère le tournoi,
    - `save_results(data)` : enregistre les résultats dans le fichier texte.

Éléments attendus

- Utilisation d'une **boucle imbriquée** (une pour le tournoi, une pour les manches).
- Manipulation de **fichiers texte** avec les fonctions `open()`, `write()`, `close()`.
- Gestion d'une **structure de données** pour stocker les scores (par exemple un dictionnaire).
- Affichage clair et lisible des résultats intermédiaires et finaux.
- Respect des **principes de modularité** et de **réutilisation du code**.

Exemple d'exécution attendue

```
=== Tournoi de Chifoumi ===
Nombre de parties : 3
Nombre de manches par partie : 5

--- Partie 1 ---
Vainqueur : Joueur
```

```
--- Partie 2 ---
Vainqueur : Ordinateur

--- Partie 3 ---
Vainqueur : Joueur

=== Résumé du tournoi ===
Joueur : 2 victoires
Ordinateur : 1 victoire
Égalités : 0
Vainqueur du tournoi : Joueur

Les résultats ont été enregistrés dans scores.txt
```

Améliorations possibles (bonus)

- Ajouter un **menu principal** :
  - 1 : Lancer un tournoi
  - 2 : Afficher les résultats précédents
  - 3 : Quitter
- Permettre le **paramétrage du tournoi** (nombre de manches et de parties).
- Ajouter un **niveau de difficulté** :
  - "Facile" : choix aléatoire simple
  - "Difficile" : choix influencé par les tours précédents
- Créer un **rapport des scores** en fin de partie au format CSV ou JSON.

Contraintes techniques

- L'utilisation du module `random` est obligatoire.
- Les données doivent être sauvegardées dans un fichier texte.
- Le code doit être commenté, indenté et organisé en fonctions.
- Aucune interaction graphique (console uniquement).