Python : TP2

Dans ce TP, nous allons revoir les bases du python vues jusquà présent en posant les bases d'un moteur de jeu de cartes, dans le style Hearthstone ou Magic.

Principe (simplifié) :
Chaque carte représente un personnage, aussi appelé serviteur.
Chaque serviteur a :

- un nom
- un nombre de points de vie
- un nombre de points d'attaque
- un coût en mana

Les parties se déroulent en duel. Chaque joueur à :

- un nombre de points de vie
- un nombre de points de mana

Le but du jeu est de faire tomber le nombre de points de vies de l'adversaire à 0 en premier.

L'espace de jeu comporte plusieurs parties :

- la pioche
- la main de chaque joueur, ou sont stockées les cartes que possède chaque joueur, et qui n'ont pas encore été utilisées
- le terrain, on sont posés les serviteurs qui peuvent attaquer et être attaqués.

Initialisation :
Une pioche commune contient toutes les cartes.
Chaque joueur commence avec :

- 1 point de mana (augmente de 1 à chaque tour)
- 30 points de vie
- 4 cartes

A tour de rôle, chaque joueur :

1. pioche une carte (pioche -> main)
2. attaque avec ses serviteurs déjà sur le terrain : chaque serviteur peut attaquer une fois, soit un serviteur adverse, soit le joueur ennemi
3. pose de nouveaux serviteurs sur le terrain (main -> terrain) en fonction de son mana

Attaques :
Quand un serviteur attaque un autre serviteur, chacun retire un nombre de points de vie à l'autre égal à son nombre de points d'attaque.
Par exemple, si A (3PVie, 2PAttaque) attaque B(3Pvie, 4Pattaque), après l'attaque, A aura (-1PVie, 2PAttaque) et sera donc mort, et B aura (1Pvie, 4Pattaque).

Quand un serviteur attaque un joueur, il lui retire un nombre de points de vie égal à son nombre de points d'attaque, mais ne prend pas de dégats.

Exercices :
A - on représentera une carte/serviteur par un dictionnaire, ou les différentes clés seront :

- name : le nom du serviteur
- health : le nombre de points de vie
- attack: le nombre de points de dégâts
- cost : le coût en mana

Écrire une fonction **loadCard** qui prend ces 4 paramètres en arguments et retourne le dictionnaire correspondant.

B - Écrire une fonction **printCard** qui prend en paramètre un serviteur et un champs displayMana (qui vaut True par défaut) et affiche sur une ligne :

- “name (attack/health) : mana” si displayMana vaut True
- “name (attack/health)” sinon

C - Écrire une fonction **fight** qui prend en paramètre deux serviteurs et les fait attaquer (diminuer leur points de vies en fonction de l’attaque de l’autre)

D - Écrire une fonction **loadCardSet** qui prend en paramètre un nom de fichier, et retourne la liste des serviteurs contenus dedans. Le fichier contient les descriptions des serviteurs en en décrivant un par ligne, au format ***“name attack health mana”*** **name** étant une chaîne de caractères contenant uniquement des lettres, et **attack**, **health** et **mana** étant des entiers.

E - On va maintenant représenter un joueur par un dictionnaire contenant :

- son nombre de points de vie (clé : **health**)
- son nombre de points de mana (clé : **mana**)
- la liste de ses cartes en main (clé : **hand**)
- la liste de ses serviteurs sur le terrain (clé : **field**)

Écrire une fonction **initPlayer** qui prend en paramètre une liste de serviteurs (deck), et qui retourne un joueur, dont les valeurs sont les suivantes :

- heath = 30
- mana = 1
- hand = liste contenant 4 copies aléatoires de cartes de deck
- field = liste vide

F - Écrire une fonction **playerTurn** qui prend en paramètres deux joueurs, **player** et **ennemy** et qui :

1. interagit avec l'utilisateur pour lui demander qui il veut attaquer avec ses serviteurs (utilisez la fonction printCard pour afficher les serviteurs pouvant attaquer et les serviteurs adverses), et effectuer les instructions au fur et a mesure.
2. interagit avec l'utilisateur pour lui demander quels serviteurs il veut placer sur le terrain, et effectuer les instructions au fur et a mesure.

G - Écrire une fonction principale qui charge les serviteurs, puis fait se dérouler une partie comme expliqué en introduction.