TP Classes - Un jeu Pokémon

Terminale NSI - Lycée Henri Wallon

Erwan DEMERVILLE

Sommaire

1	${f R}$ éa	lisation d'un jeu Pokémon	2
	1.1	Objectif du TP	2
	1.2	Pré-requis	2
	1.3	Modélisation	2
	1.4	Réalisation	4
		1.4.1 Déroulement du jeu	4
		1.4.2 Déroulement d'un combat	5

1 Réalisation d'un jeu Pokémon

1.1 Objectif du TP

L'objectif de cette séance est de réaliser, en **Python**, un jeu de type RPG. En particulier, on s'inspirera du jeu **Pokémon**.

Ce projet sera constitué des fichiers suivants :

- main.py Programme principal permettant l'exécution du jeu
- jeu.py Classe qui gère l'ensemble du jeu et réutilise les autres classes
- combat.py Classe permettant de représenter un combat du jeu
- pokemon.py Classe permettant de représenter les Pokémons

Voici une capture de ce que l'on souhaite obtenir :

Figure 1: Exécution du jeu

1.2 Pré-requis

Si nécessaire, vous pouvez relire le cours sur les classes en cliquant ici.

Les notions d'objets, d'attributs, de méthodes, de constructeur notamment doivent vous être familières.

1.3 Modélisation

Voici une modélisation sous forme de diagramme des différentes classes constituant ce projet :

(list) de Pokemon pokemons nb victoires int nb defaites int nb_combats_joues int init_(): Constructeur de la classe getNbVictoires(): (int) Retourne nombre de victoires getNbDefaites(): (int) Retourne nombre de défaites getNbCombatsJoues : (int) Retourne nombre de combats joués ajouterVictoire(): / Ajoute 1 victoire ajouterDefaite(): / Ajoute 1 défaite ajouterCombatJoue(): / Ajoute 1 combat joué pokemonAleatoire(pokemon_choisi) : (Pokemon) Retourne un objet Pokemon aléatoire de la liste _pokemon, qui n'est pas pokemon_choisi. choixPokemon(): Affiche une liste des pokémons vivants, invite l'utilisateur à choisir un pokémon, et le retourne. deoulementTour(): / Gère le déroulement d'un tour.

Combat pokemon joueur (Pokemon) Le pokémon du joueur pokemon ennemi: (Pokemon) Le pokémon de l'adversaire int - Numéro du tour ntour <u>init__(pokemon_joueur, pokemon_ennemi)</u> : Constructeur de la classe getNumeroTour(): (int) Retourne le numéro du tour actuel augmenterTour(): Augmente de 1 le nombre de tours afficherEtat(): Affiche les informations sur les pokémons (points de vie, état, etc.) activationEffets(): Exécute les effets liés à l'état des pokémon (poison, paralysie. iouer() : (bool) Exécute le combat au tour par tout attaquer(attaque, attaquant, cible) : / Lance une attaque sur un autre joueur messageVictoire(): Affiche un message de victoire messageDefaite(): Affiche un message de défaite.

Pokemon (str) Nom du pokémon types (list of str) Types du pokémon pts vies (int) Points de vie du pokémon (int) Attaque du pokémon attaque : defense (int) Défense du pokémon liste_attaques : (list of dicts) Liste des attaques etat (dict) État du pokémon init__(nom, types, pots_vies, attaque, defense, liste_attaques) : Constructeur de la classe getNom, getVies, getAttaque, getDefense, getEtat : Récupérer les valeurs des attributs associés setEtat(): Modifier la valeur de _etat baisserVies, baisserAttaque, baisserDefense : Baisser d'une certaine valeur les augmenterVies, augmenterAttaque, augmenterDefense : Idem mais pour augmenter. estMort(): (bool) Retourne True si pokémon est mort, False sinon choixAttaqueAleatoire(): Retourne les infos d'une attaque choisie aléatoirement choixAttague() : Affiche la liste des attagues puis invite le joueur à en choisir une Un tableau contenant ses informations est retourné.

Figure 2: Modélisation des classes du jeu

Notes:

• Les données sur les pokémons sont stockées dans un fichier pokemons.txt.

L'inscription d'un pokémon se fait de la manière suivante :

```
Bulbizarre Plante, Poison 45 49 49
Charge, 50,_,_ Rugissement,_, attaque-e, 5 Vampigraine,_, drainage, 3
```

Chaque donnée est séparée par une tabulation.

La première ligne renseigne les données principales sur le Pokémon : Son **nom**, ses **types**, son **nombre de vies**, son **attaque** et enfin sa **défense**.

La deuxième ligne contient les attaques du pokémon.

S'il s'agit d'une attaque qui inflige simplement des dégâts, on l'écrit sous la forme : Nom_de_lattaque, puissance (donc seulement deux éléments)

S'il s'agit d'une attaque qui n'inflige pas de dégâts mais effectue une autre action, ou inflige une altération d'état, on écrira sous la forme suivante (3 éléments) :

Nom_attaque,poison,3: Empoisonnement pendant 3 tours (par exemple)

Nom_attaque, attaque+, 10 : Augmenter de 10 points l'attaque du pokémon allié

Nom_attaque, defense-e, 10: Diminuer de 10 points la défense du pokémon ennemi.

Pour l'instant, il existe :

- poison: Empoisonner l'ennemi
- paralysie : Paralyser l'ennemi
- drainage : Drainer la vie de l'ennemi (même effet que poison)
- attaque+, attaque-, defense+, defense-: Augmenter/Diminuer l'attaque ou la défense
- attaque-e, defense-e : Diminuer l'attaque ou la défense ennemie.
- Rien ne vous empêche de rajouter de nouveaux effets, si vous vous sentez capable de les rajouter dans le code.

Les attaques de chaque Pokemon seront enregistrées dans l'attribut _liste_attaques sous forme d'une liste de dictionnaires.

Par exemple, s'il y a 2 attaques : [{"nom_attaque": "Charge", "degats_attaque": 35}, {"nom_attaque": "Rugissement", "effet_attaque": "attaque-e", "valeur_attaque": 5}]

- La première attaque "Charge" a une **puissance de 35**.
- La seconde attaque diminue l'attaque de l'ennemi de 5 points.
- L'état du pokémon sera stockée sous la forme d'un dictionnaire.
 - Initialement, l'attribut _etat sera égal à {"nom_etat": "normal"} => Cela indique que le pokémon est dans son état normal.
 - S'il est empoisonné par exemple, _etat sera égal à {"nom_etat": "poison", "duree_etat": 3} => Empoisonnement **pendant encore 3 tours**. (Même principe pour la **paralysie**, ou le **drainage**).
- Vous pouvez ajouter de nouveaux pokémons dans le fichier pokemons.txt, en respectant bien le format. Vous pouvez créer vos propres pokémons ou vous inspirer du pokédex :

https://www.pokebip.com/pokedex/pokedex_5G_liste_des_pokemon.html

1.4 Réalisation

Ouvrez le dossier Projet_Pokemon.

A l'intérieur, vous trouverez des fichiers pré-remplis, que vous devrez compléter.

Les zones à complétez sont marquées par des pointillés

- Pensez à écrire la docstring de vos fonctions.
- Certaines fonctions sont **déjà** entièrement **écrites**, et il vous sera dans ce cas demandé d'**écrire la docstring** pour vérifier votre compréhension.
- D'autres fonctions **contiennent uniquement la docstring**, qui vous aidera à écrire le corps de la fonction.

Lisez bien les docstrings et les commentaires pour vous assurer de bien comprendre le fonctionnement du projet!

1.4.1 Déroulement du jeu

Le jeu, défini dans la classe Jeu, se déroule indéfiniment et ne s'arrête que si :

- Tous les pokémons du jeu sont morts
- Le nombre de victoires correspond à la valeur de la variable globale NB_VICTOIRES
- Le nombre de défaites correspond à la valeur de la variable globale NB DEFAITES

Si aucune de ces conditions n'est remplie :

- On demande au joueur de choisir un pokémon parmi les pokémons encore vivants du jeu
- L'adversaire choisit un pokémon de manière aléatoire

- On lance un combat.
- A la fin du combat, en cas de victoire, on incrémente la valeur du nombre de victoires, en cas de défaite, on incrémente la valeur du nombre de défaites.
- On recommence le même déroulement tant qu'aucune des 3 conditions définies précédemment n'est remplie.

1.4.2 Déroulement d'un combat

Pour vous aider à comprendre, voici une explication du déroulement d'un combat (méthode jouer de la classe Combat) :

- On vérifie si le pokémon du joueur est mort. Si oui, fin du combat, avec un message de défaite.
- Sinon, on vérifie si le pokémon de l'ennemi est mort. Si oui, fin du combat, avec message de victoire.
- Si aucun pokémon n'est mort :
 - On affiche les infos sur les pokémon (points de vies, état, attaque, défense...)
 - On active les effets relatifs aux altérations d'état.
 - Si c'est au tour du joueur, le joueur choisit une attaque.
 - Si c'est au tour du joueur adversaire, l'attaque est choisie aléatoirement.
 - A la fin, on incrémente le nombre de tours