SIMULACIÓN COMBATE POKEMON

Aldana Smyna Medina Lostaunau (aldana.medina@udc.es)

Claudia Vidal Otero (claudia.votero@udc.es)

Manual de Usuario: Simulador de Batalla Pokémon

Al iniciar este programa, podrás simular batallas entre entrenadores y sus Pokémon. A continuación, tiene una guía paso a paso para comenzar a utilizar el simulador:

1. Preparación del Archivo de Entrada

Antes de comenzar, asegúrate de tener un archivo de texto que contenga la información sobre los entrenadores y sus Pokémon. El archivo debe seguir el siguiente formato:

Entrenador 1

Pokemon 1 (Tipo: Fire, Nivel: 50, Atk: 76, Def: 38, HP: 192, Agi: 41, Temperature: 0.2)

Pokemon 2 (Tipo: Water, Nivel: 30, Atk: 74, Def: 36, HP: 190, Agi: 39, Surge_mode: False)

...

Entrenador 2

Pokemon 1 (Tipo: Grass, Nivel: 40, Atk: 120, Def: 44, HP: 352, Agi: 50, Healing: 0.15)

Pokemon 2 (Tipo: Fire, Nivel: 35, ...)

Cada entrenador debe tener un nombre seguido de una lista de Pokémon, donde cada Pokémon está en una línea separada y contiene sus datos: (1) nombre, (2) tipo, (3) nivel, (4) fuerza, (5) defensa, (6) puntos de vida, (7) agilidad y (8) (temperatura/healing/surge_mode) **en ese orden**. En última posición existe un atributo específico para FirePokemon, GrassPokemon y WaterPokemon, respectivamente. Asegúrate que el nombre y tipo de Pokémon sean cadenas de texto; healing y temperatura sean float; surge_mode sea False; y que el resto de atributos sean números enteros.

2. Ejecución del Programa

Para ejecutar el programa, abre una terminal y navega hasta el directorio donde se encuentran los archivos del simulador (main.py, trainer.py, pokemon.py) y el archivo de entrada. Luego, ejecuta el programa de la siguiente manera:

python main.py nombre_archivo.txt

Reemplaza 'nombre_archivo' con el nombre de tu archivo de entrada.

3. Observación de la Batalla

Una vez que ejecutes el programa, comenzará la simulación de la batalla que consta de varios combates entre dos Pokémon de los entrenadores. Un combate consta de una o más rondas. Verás la información sobre cada ronda de la batalla, incluyendo los Pokémon que están luchando y las acciones que realizan: dos acciones de ataque, una por parte de cada Pokémon en donde el Pokémon con mayor agilidad es el primero de los dos en atacar (en caso de empate, se priorizará el del primer entrenador).

4. Revisión de Estadísticas

Después de que la batalla haya concluido, el programa generará estadísticas detalladas sobre el desempeño de los Pokémon durante la batalla. Estas estadísticas se presentarán en la terminal y también se guardarán en una lista.

Mecanismo de Ejecución del Programa

El programa comienza importando dos módulos de Python, **sys** y **pandas**, y otros que contienen definiciones de clases y funciones necesarias para la simulación: **pokemon** y **trainer**. Luego, se define la clase PokemonSimulator, la cual se encarga de organizar la simulación y proporciona métodos para la creación de entrenadores, la ejecución de rondas de batalla y el manejo de eventos clave durante la batalla.

Dentro de la clase PokemonSimulator, se encuentran funciones esenciales como create_trainer_and_pokemons, la cual interpreta el archivo de entrada proporcionado para crear un entrenador y sus Pokémon asociados. El método parse_file analiza el texto completo que representa a dos entrenadores y sus Pokémon, creando instancias de **Trainer** para cada uno.

El mecanismo principal de la simulación se lleva a cabo en la función **battle**. Aquí, se seleccionan los Pokémon iniciales para cada entrenador, se imprime un mensaje indicativo del inicio de la batalla con print_battle_start, y se inicia un bucle *while* que continúa hasta que uno de los Pokémon esté debilitado, es decir su HP sea igual a 0 y se queda sin vida.

En cada ronda, se determina el orden de ataque con determine_attack_order y se ejecuta la ronda con execute_round, el cual imprime información sobre la ronda, realiza ataques y maneja debilitaciones. El método attack desencadena los ataques específicos de cada tipo de Pokémon, y los resultados se registran en instancias de la clase **Estadistica** almacenadas en **lista_datos_pokemon** para futuros análisis.

Los ataques se ejecutan en el método attack, el cual determina el tipo de ataque en función de las condiciones actuales, como si la ronda es impar o par y el tipo de Pokémon. Se utilizan métodos específicos de cada tipo de Pokémon, como 'grass_attack', 'fire_attack' y 'water_attack', para realizar los ataques. Se imprimen mensajes detallados sobre los ataques y los resultados obtenidos. Cada tipo de Pokémon se ha definido como una subclase de la clase base Pokemon (creada en pokemon.py), esta jerarquía de clases aprovecha el mecanismo de herencia para organizar y compartir comportamientos comunes, mientras que cada clase hija puede proporcionar implementaciones específicas y personalizadas de métodos.

Cabe recalcar que el método heal de los Pokémon de planta devuelve los puntos de curación que realmente el Pokémon ha recibido. Es decir, en el caso de que la curación exceda el total de HP del Pokémon, el método calculará y devolverá únicamente la cantidad de puntos reales hasta ese límite. Esto asegura que el Pokémon no tenga más HP del permitido, y además proporciona la cantidad precisa de curación que se le ha aplicado.

Cuando un Pokémon está debilitado, se llama a handle_debilitated_pokemon, que verifica si el entrenador tiene más Pokémon disponibles para continuar la batalla o si ha ganado el enfrentamiento. Finalmente, al concluir la batalla, se imprime un mensaje indicando el fin del enfrentamiento y el nombre del entrenador ganador.

La función **main** es la que se encarga de la lectura de un archivo (especificado como argumento de línea de comandos) que contiene información sobre los entrenadores y sus Pokémon.

El resultado final es un programa orientado a objetos que simula batallas Pokémon. La POO nos permite modelar de manera eficiente las relaciones y comportamientos entre entrenadores y Pokémon.

Durante la batalla, se registran los siguientes datos de los Pokémons para las estadísticas: el tipo, nombre, daño infligido y healing realizado por el atacante, mientras que del defensor solo se muestra su tipo.

Una vez que la batalla ha terminado, se crea un DataFrame de pandas a partir de los datos recopilados en lista_datos_pokemon. Luego, llamando a la función estadisticas se realizan varios análisis de datos utilizando este DataFrame para calcular estadísticas sobre el desempeño de los Pokémon durante la batalla, en específico la media y desviación típica de estas 5:

(1) el daño promedio causado por cada Pokémon individualmente.

(2) el daño promedio causado por los Pokémon de cada tipo (agua, fuego, planta).

(3) el daño promedio que cada tipo de Pokémon inflige a cada uno de los otros tipos.

(4) la curación promedia realizada por cada Pokémon.

(5) la curación promedia realizada por los Pokémon de cada tipo.

Fases de Desarrollo

1. Diseño de Clases: Se diseñaron las clases 'Trainer' y 'Pokémon', definiendo sus atributos y métodos necesarios para simular las batallas Pokémon y recopilar estadísticas.
2. Implementación de Funcionalidades: Se implementaron métodos para crear entrenadores y sus Pokémon, imprimir por pantalla el número de ronda y los datos de los Pokémon al principio de cada combate, realizar ataques propios o básicos según el número de ronda, determinar el orden de ataque, simular la batalla entre los entrenadores y recopilar estadísticas de la batalla.
3. Pruebas Unitarias y Depuración: Se realizaron pruebas para garantizar que el programa funcionara correctamente en una variedad de situaciones. Se depuraron errores y se realizaron ajustes en la lógica del programa según fuera necesario.
4. Guardado de datos: Creación de la clase 'Estadistica' para guardar los datos de cada ronda, del DataFrame de Pandas y de los cálculos que imprimen las medias y desviaciones típicas.
5. Optimización y Documentación: Se recolocó el código para mejorar su coherencia y se documentaron todas las clases, métodos y funciones mediante docstrings para facilitar su comprensión y mantenimiento.