Software Dev. & Problem Solving I — GCIS-123

Methods & Special Methods — Assignment 11.2

Goals of the Assignment

The goal of this assignment is to get more practice with Python classes and methods, including special methods. You will implement a Pokémon battle simulation based on a combination of Pokémon, the War card game, and Rock, Paper, Scissors. You will be expected to demonstrate good software engineering practice including ***encapsulation, test-driven development*** (***TDD***), and good use of version control. Read this document ***in its entirety*** before seeking help from the course staff.

A Pokémon has 4 characteristics: name, type, health points, and damage points. At the start, each Pokémon has an initial amount of health points. During the battle, a Pokémon will lose health points if it loses a round. The damage points of the winning Pokémon determine the decrease in health points of the losing Pokémon during a round. If a Pokémon's health points reach 0 or below, it faints and is out of the battle.

Valid Pokémon types are: Fire, Water, and Grass. Each of the types can defeat exactly one other type.

- Water douses Fire
- Fire burns Grass
- Grass consumes Water

Activities

1. Create a new file named "`pokemon.py`". Define a Pokémon class as follows:
    a. In the `__slots__` static field, declare private fields for `name` (string), `type` (string), `health_points` (integer), and `damage_points` (integer).
    b. Add a constructor with parameters for `self` and the fields described in (a).
        - Use the constructor parameters to initialize the instance fields.
    c. Add an accessor (getter) for the `damage_points` field.
    d. Add a `lose_round` method that declares parameters for `self` and `damage_points`. The function should subtract the `damage_points` from the Pokémon's current `health_points`
    e. Add an `is_fainted` method that declares a parameter for `self`. The function returns `True` if a Pokémon's `health_points` are less than or equal to 0, `False` otherwise.
    f. Add a `__str__` method that declares a parameter for `self` and returns the `name` field

g. Add a `__repr__` method that declares a parameter for `self` and returns the following `string`: "`<name>:<type>:<health_points>`"
Note: As discussed, the `__str__` method should return a compact string and the `__repr__` method should return a verbose string. However, since we will be passing data structures containing `Pokemon` objects to the `print` function, we will be taking liberties with these methods to achieve nicely formatted output.

h. Relational operators will be used during battle to determine the winner, if any, during each round.
   - Two `Pokemon` are equal if their `type` and `health_points` are the same.
   - One `Pokemon` is less than, or conversely greater than, the other based on the `type` and `health_point` fields. Recall that
     - Water douses Fire
     - Fire burns Grass
     - Grass consumes Water

     If the `type` field is the same for both Pokemon, comparing the `health_point` fields will determine if one Pokemon is less than or greater than the other.

i. Add a `__hash__` method

2. You have been provided a CSV file of Pokémon characters – `data/pokemon.csv`. You will use this file to populate your Pokedex.

   - Each record in the CSV file has 4 fields: Name, Type, Health Points, and Damage Points
   - Feel free to add your own Pokémon to the file, but ensure it is of one of the three types

   Create a new Python file named "`pokedex.py`". Define a `Pokedex` class as follows:
   a. In the `__slots__` static field, declare a private field for `pokemon_list`
   b. Add a constructor with a parameter for `self`
      - Initialize the `pokemon_list` to an empty list
   c. Add a `load` method that declares parameters for `self` and `filename`. For each CSV record, create a new `Pokemon` object using the record fields and add it to `pokemon_list`.
   d. Add a `create_parties` method that declares a parameter for `self`.
      Create two `sets` - each representing the Pokémon `party` of each trainer
      Randomize `pokemon_list` (hint: random.shuffle())
      Add 6 `Pokemon` from `pokemon_list` to each `set`
      Ensure that all 12 `Pokemon` added to the `sets` are unique
      Return both `sets`

3. Create a new Python file named "`pokemon_battle.py`". Add a function called `battle` that declares parameters for two `sets`: `party1` and `party2`. When printing a `Pokemon` or either `party`, you should pass just the `Pokemon` or `set` to the print function and let the `__str__` and `__repr__` methods handle the details.

   While each trainer's `party` has at least one `Pokemon`
      Print the round number and each party
         "`Round: <round_number>`

Party 1: <party1>
Party 2: <party2>"
Remove a pokemon from each trainer's party (hint: use set.pop())
Compare both Pokemon using relational operators
If both Pokemon are equal
Print "<pokemon1> and <pokemon 2> battle to a draw"
Add both pokemon back into their respective party
Else, the greater Pokemon is the winner of the round
Print "<winner> has won the round over <loser>"
Place the winner back into their party
Call the lose_round method of the loser
If the loser has fainted
Print "<loser> has fainted and is out of the battle"
Else
Place the loser back into their party

Prompt the user to press enter to continue to the next round

When all of the pokemon from one party has fainted, print "Winning: Party <party>"

4. Add a `main` function to your `pokemon_battle.py` module.

Create a Pokedex object
Load the data from the CSV file
Create the parties
Call battle to run the simulation

Example Output

```
Round 1
Party 1: {Starmie:Water:46, Floatzel:Water:27, Sceptile:Grass:49, Magcargo:Fire:45,
Sawsbuck:Grass:46, Feraligatr:Water:34}
Party 2: {Blastoise:Water:36, Lapras:Water:43, Jellicent:Water:28, Magmortar:Fire:34,
Simisear:Fire:45, Carnivine:Grass:49}
Starmie has won the round over Blastoise
Press enter for next round...

. . .

Round 9
Party 1: {Starmie:Water:46, Floatzel:Water:15, Sawsbuck:Grass:46,
Feraligatr:Water:34, Sceptile:Grass:49, Magcargo:Fire:45}
Party 2: {Blastoise:Water:9, Jellicent:Water:28, Lapras:Water:13, Magmortar:Fire:25,
Simisear:Fire:45, Carnivine:Grass:49}
Floatzel has won the round over Blastoise
Blastoise has fainted and is out of the battle
Press enter for next round...

. . .
```

```
Round 36
Party 1: {Magcargo:Fire:12}
Party 2: {Simisear:Fire:45, Carnivine:Grass:49}
Simisear has won the round over Magcargo
Magcargo has fainted and is out of the battle
Press enter for next round...

Winning Party: {Simisear:Fire:45, Carnivine:Grass:49}
```