Pokémon Battle Version 2: Adding and testing Pokémon methods

In this version, you will add methods to the `Pokemon` class. Then, you will test your methods (we wrote the unit tests for you).

What to do

Use the `Pokemon` class that you implemented in version 1. Keep the `__init__` method as is in the class. Add the following methods in that class.

1. `__str__ method`: This method will return a string representation of the Pokémon. This string should contain the name of the Pokémon, its current health and its maximum health.
Consider the following examples for how the string should be formatted:

If the name of the Pokémon is Pikachu and its current health is the same as its maximum health, the string returned by the method should be:

```
Pikachu (health: 35/35)
```

If the name of the Pokémon is Bulbasaur and its current health is 35 and its maximum health is 45, the string returned by the method should be:

```
Bulbasaur (health: 35/45)
```

2. `lose_health method`: This method will take in the `amount` of health lost as a parameter and it will change the current health of the Pokémon as follows:

- If `amount` is less than the current health, the current health of the Pokemon should decrease by the `amount`
- If `amount` is greater or equal than the current health, the current health of the Pokemon should decrease to 0
- If `amount` is negative, nothing should happen

3. `is_alive method`: This method will check if the Pokémon is alive or not. A Pokémon is alive if its current_health is greater than 0.

4. `revive method`: This method will revive the Pokémon. A Pokémon is revived by setting its current heath to its maximum health.

When adding the method in the `Pokemon` class, you must use the following method definitions to complete the task:

```python
def __str__(self) -> str:
    """
    Return a string representation of the Pokemon.
    """
    # TODO: Implement this method.

def lose_health(self, amount: int) -> None:
    """
    Lose health from the Pokemon.
    """
    # TODO: Implement this method.

def is_alive(self) -> bool:
    """
    Return True if the Pokemon has health remaining.
    """
    # TODO: Implement this method.

def revive(self) -> None:
    """
    Revive the Pokemon.
    """
    # TODO: Implement this method.
    print(f"{self.name} has been revived!")
```

Your `main` function will change a little bit. Since you now have the `__str__` method, you can use it to print the Pokémon.

Here is your updated `main()` function:

```python
def main():
    """
    Battle of two Pokemon
    """
    pokemon1 = Pokemon("Pikachu", 55, 40, 35, 35)
    pokemon2 = Pokemon("Bulbasaur", 49, 49, 45, 45)
    print(f"Welcome, {pokemon1} and {pokemon2}!")
```

Hints

- Each method should do one thing, so the code of each method should be short. Some methods will only have one line, and others will likely have no more than 3-5 lines of code.

Program name

Save your program as `pokemon2.py`.

Demo

https://asciinema.org/a/XtVvAAjKdsQW1AGsdtytfcO8P

Testing

To make sure your program works correctly, you should test it.

- Run your program with `python pokemon2.py`. Your program should print:

```
Welcome, Pikachu (health: 35/35) and Bulbasaur (health: 45/45)!
```

Unit Tests

We wrote unit tests for you to test the methods: test_pokemon2.py.

You should run them with `python -m pytest test_pokemon2.py`.

Make sure that all tests pass.

Submitting

Submit `pokemon2.py` via eClass.

You may submit either all versions you complete, or only the final version.

Copyright

I. Akhmetov, J. Schaeffer, M. Morris and S. Ahmed, Department of Computing Science, Faculty of Science, University of Alberta (2023).