Dominic Lim

Project 1 Reflections – NOT POKEMON

Overview:

NOT POKEMON is a turn-based game in which the human player picks 1 of 3 POKEMON "types" to battle against an instantiated Computer Player and POKEMON. The entire sequences of game stages is managed in the NotPokemon class. The gameplay makes heavy use of ASCII art.

The best way to test the game is to simply play! By choosing the 3 different POKEMON types, each battle will exhibit the different characteristics of each POKEMON. Ultimately the game should be consistent in that either terminating condition will end the game:

- Computer's POKEMON or the Player's POKEMON's HP drops to 0.
- The Player decides to surrender.

The game is composed of 4 .py files:

- NotPokemon.py (the main game .py file, imports from below modules)
- pokemon_monster.py (holds Monster and LightMonster, FireMonster, IceMonster subclasses)
- pokemon_player.py (holds Player class)
- pokemon_ascii.py (holds all ASCII art)

To start, the player runs the game in Terminal:

```
limdo@DESKTOP-LJ3SJ1H MINGW64 ~/OneDrive/Desktop/github/mids-w200-fall21-Dominic-LimREPO/project_1 (main)
$ python NotPokemon.py
```

Note: **PLEASE MAXIMIZE YOUR TERMINAL WINDOW AND ZOOM OUT FOR BEST EXPERIENCE**

To Do:

One thing I wanted to build out is a HIGH SCORE Ranking. By building out a dictionary of Player names (key) and the Tuples of # of games won and total points of incurred damage (value).

I also would like to refactor the code to be able to better scale the ASCII art in any sized terminal window. In the future, I would like to use a GUI such as Tkinter.

Challenges:
One of the biggest challenges was to get ASCII art to fit on a terminal window screen, especially in the main battle sequence. This was my most unexpected challenge which also involved rescaling ASCII images manually. One solution that I did take advantage of was using `os.system('cls||clear')` to clear the terminal window.

Ultimately, these solutions are impermanent since users will have different terminal window settings, and zoom in/out settings.

Another challenge I encountered was testing the game mechanics. Since the random library is used heavily in calculating damage output, testing involved tinkering with `random.choices(weights param)` and a LOT of player testing.