Programming Project 11

Updated 11/26/2019

This assignment is worth 55 points (5.5% of the course grade) and must be **completed and turned in before 11:59 on Thursday, December 5, 2019.**

Assignment Overview

This assignment will give you more experience creating and using Python classes. We will create our own pokemon and move classes in the `pokemon.py` file. Then we will utilize those classes to have pokemon battles in the project.py file.

Assignment Background

Pokémon is a very successful franchise not only in Japan and the United States, but worldwide. https://en.wikipedia.org/wiki/Pokémon. Some of you may be familiar with the franchise so we hope to increase your interest in programming by coding something similar. And for those of you unfamiliar, we hope this may generate more interest in the types of things you want to create with code.

CSV files for the pokemon and moves are received from the following sources:

- https://github.com/veekun/pokedex/tree/master/pokedex/data/csv
  - from veekun/pokedex GitHub Repository

- https://gist.github.com/armgilles/194bcff35001e7eb53a2a8b441e8b2c6
  - from armgilles GitHub Gist

Project Specifications

pokemon.py

This file contains two classes: Move and Pokemon. It also contains three dictionaries that are used when calculating damage of moves. These dictionaries are already completed and should not be changed in any way. It is your job to fill out the methods in the Move and Pokemon classes. The initialization methods are provided.

Move class

Attributes:

- **name**: the name of the move (string)
- **element**: the move's type (string)
- **power**: the base damage that the move inflicts (int)
- **accuracy**: how likely the move is to hit an opponent (int)
- **attack_Type**: classification if this is a physical attack, special attack or status move (int)

Methods:

1. **__init__ (self, name, element, power, pp, accuracy, attack_type)**
   This method initializes attributes of the Move object. (already given)

2. **__eq__(self) --------→ boolean**
   True if attributes are equal; False otherwise. Provided by instructors

3. **__str__ (self) ------→ str**
   Returns just the name of the move (used for printing).
   # for printing. Takes 1 arg: self. Returns a string.

4. **__repr__ (self) ----→str**
   Returns just the name of the move (can utilize the __str__() method here).
   # for displaying in the shell. Takes 1 arg: self. Returns a string.

5. **get_name (self) ---→ str**
   Returns the name attribute.

6. **get_element (self) --→str**
   Returns the element attribute.

7. **get_power (self) ---→ int**
   Returns the power attribute.

8. **get_accuracy (self) --→ int**
   Returns the accuracy attribute.

9. **get_attack_type (self) --→ int**
   Returns the attack_type attribute.

Pokemon class

Attributes:

- **name**: the name of the pokemon (string)
- **element1**: the first element of the pokemon (string)
- **element2**: the second element of the pokemon (string) (None if the pokemon has one element)
- **hp**: the health points of the pokemon, when this reaches 0 the pokemon faints (int)
- **patt**: the physical strength of the pokemon (int)
- **pdef**: the physical defense of the pokemon (int)
- **satt**: the special strength of the pokemon (int)
- **sdef**: the special defense of the pokemon (int)
- **moves**: the list of moves that this pokemon has access to (list of Moves)

Methods:

1. **__init__ (self, name, element1, element2, moves, hp, patt, pdef, satt, sdef)**
   This method initializes attributes of the Pokemon object. (already given)

2. **__str__ (self) ---→ str**
   Returns a string containing the parts of the pokemon object divided into three lines. The first line will display in this order: name, hp, patt, pdef, satt and sdef, followed by a newline character directly after sdef with no space inbetween sdef and the newline character. The second line will display the element1 and element2 with a newline character after element2. The third line will display all the moves of that pokemon.

   Each attribute will take up exactly 15 spaces and be left adjusted no matter the type.

3. **__eq__(self) ---→Boolean**
   True if attributes are equal; False otherwise. Provided by instructors

4. **__repr__ (self) --→ str**
   Returns the same value as the __str__() method to.
   # for displaying in the shell. Takes 1 arg: self. Returns a string.

5. **get_name (self) --→ str**
   Returns the name attribute.

6. **get_element1 (self) --→ str**
   Returns the element1 attribute.

7. **get_element2 (self) --→ str or None**
   Returns the element2 attribute.

8. **get_hp (self) ---→ int**
   Returns the hp attribute.

9. **get_patt (self) ---→ int**
   Returns the patt attribute.

10. **get_pdef (self) --→ int**
    Returns the pdef attribute.

11. **get_satt (self) --→ int**
    Returns the satt attribute.

12. **get_sdef (self) --→ int**
    Returns the sdef attribute.

13. get_moves (self) --→ list
Returns the moves attribute.

14. get_number_moves (self) --→ int
Returns the number of moves.

15. choose (self, index) --→ Move class object or None
Takes an index and returns the corresponding move from the moves list. If there is an IndexError returns None.

16. show_move_elements(self)
Displays the elements of the pokemon's moves (each in a 15-space field, left justified). This function does not return anything.

17. show_move_power(self)
Displays the power of the pokemon's moves (each in a 15-space field, left justified). This function does not return anything.

18. show_move_accuracy(self)
Displays the accuracy of the pokemon's moves (each in a 15-space field, left justified). This function does not return anything.

19. add_move (self, move)
Adds the move parameter to the list of moves for this pokemon if this pokemon has three or less moves. This function does not return anything.

20. attack (self, move, opponent)
This method takes the move used by the attacker (self) and deals damage to the opponent (who should also be an instance of class Pokemon). It does not return anything.
The equation for calculating damage will be a slight variation of the actual calculator because Pokemon levels are not taken into account. It is as follows:

$$\text{Damage} = \left[\frac{\left[mp*\left(\frac{A}{D}\right)*20\right]}{50}+2\right]*modifier$$

mp: the power of the move
A: The patt or satt of the attacking Pokemon
D: The pdef or sdef of the defending Pokemon
modifier: takes into effect same-type attack bonus (STAB) if move is super effective or not effective.

First the power of the move is extracted using the get_power() method from the move object. Then A and D are determined by obtaining the attack_type from the move using the get_attack_type() method. If the attack_type is equal to 2 then A is the attacker's patt attribute and D is the defender's pdef attribute. If the attack_type is equal to 3 then A is the attacker's satt attribute and D is the defender's sdef. If the attack_type has some other value, print the message `"Invalid attack_type, turn skipped."` and return.

Next an accuracy value is created using randint from the random module. The accuracy value will be a random integer between 1 and 100 and will be compared to the accuracy attribute of the move object (using the get_accuracy() method). If this randomly created accuracy value is greater than the accuracy attribute of the move, then the move misses (`print("Move missed!")` and return) and no damage is dealt.

Next the modifier is calculated. The modifier is first initialized to 1.0. We begin by considering element1 of the opponent. The modifier doubles if the element of the opponent is weak with respect to the element of the attacking move (the opponent's element is in the `is_effective_dictionary` entry for the move's element). The modifier is halved if the element of the opponent is strong with respect to the element of the attacking move (the opponent's element is in the `not_effective_dictionary` entry for the move's element). Finally, no damage is done to the opponent if the element of the move has no effect on the element of opponent (the opponent's element is in the `no_effect_dictionary` entry for the move's element).

Note that you repeat that calculation for the opponent's second element. That is, you will adjust the modifier a second time, e.g. possibly doubling it again, or halving it, or doing nothing. For example, it is possible to double it for the first element and then halve it for the second element so the modifier ends up with a value of 1.0. Or you may double it twice so the modifier is 4, and so on.

If the move is super effective, i.e. modifier > 1, print `"It's super effective!!!!"`. If the modifier < 1, print `"Not very effective..."`.

Finally, there is one more possible adjustment to the modifier. The same-attack type bonus (STAB) gives a boost in damage if the element of the move is the same as one of the elements from the attacking Pokemon. If this is true, then the modifier is multiplied by 1.5.

The final part of this function is to adjust the damage to the opponent. Damage is subtracted from the opponent's health (hp). The damage should be converted to type int before being passed as a parameter to the opponent's subtract_hp() method.

21. **subtract_hp (self, damage)**
    This method takes the damage variable and subtracts it from the hp of the Pokemon object. The hp of the Pokemon object becomes the maximum of the (hp – damage) or 0. (Adventuresome students may want to try Python's ternary operator). This function does not return anything.

proj11.py
This file contains the main and other functions that are to be filled out.

1. **`read_file_moves(fp) --->list[Move objects]`**
    This function takes in the file pointer created from opening the moves.csv file and returns a list of move objects. Use csv.reader.
    In this function, you should first ignore the header line then iterate through each line of the file creating a move object with the contents of the line and adding that move object to the list of moves.
    The column labeled "identifier" is the "name" argument for the Move object.

    The column labeled "type_id" is an index into the "element_id_list" provided at the top of the file. The value found in the "element_id_list" at the "type_id" index is the "element" argument for the Move object.

    The column labeled "damage_class_id" is the "attack_type" argument for the Move object. The damage_class_id has three values, 1, 2 and 3, where 2 denotes a physical attack, 3 denotes a special attack and 1 denotes a stat changing or status effect move. If the damage_class_id is 1 then that move will not be added to the move list.

    We will **not** add moves to the move list if:
    - generation_id column does **not** equal 1.
    - damage_classification_id does equal 1.
    - there is no power value.
    - there is no accuracy value.

    Remember to change the power and accuracy attributes to integers before using them to create the move instance.

    Remember to create a Move object and append it to the list of moves.

2. **`read_file_pokemon(fp) --> list[Pokemon objects]`**
    This function takes in the file pointer created from opening the pokemon.csv file and returns a list of pokemon objects. In this function, you should first skip the header line, then iterate through each line of the file creating a pokemon object with the contents of

the line and adding that pokemon object to the list of pokemon. If the Generation column is **not** equal to '1'you should ignore the line,
Two lines may have the same ID (column 0). Keep the first one read and ignore subsequent ones. Hint: use a data structure to keep track of the IDs that you have read.

The column labeled "Type 1" is the "element1" argument for a Pokemon object; similarly the column labeled "Type 2" is the "element2" argument. For consistency make these strings lower case.

"HP" is the "hp" argument for a Pokemon object. Similarly "Attack" is the "patt" , "Defense" is "pdef", "Sp. Atk" is "satt", and "Sp. Def" is "sdef. All these numbers should be converted to ints.

There is no list of moves in this file so use None for that argument to the `Pokemon` object.

Strings such as "name" should be lower case.

Remember to create a Pokemon object instance and add it to the list

3. **`choose_pokemon (choice, pokemon_list)`--→ Pokemon object or None**
   This function takes in user input (called `choice`) as a string and the list of available pokemon. If the user input is an integer, the integer becomes the index for selecting a pokemon from the pokemon_list and **a deepcopy** of the pokemon object at that index is returned. The choice parameter starts at 1 so you need to subtract 1 before indexing. If the input is a string, then that string is compared to the name attribute of the pokemon in the list and if a name matches then **a deepcopy** of the pokemon object with that name is returned. If the index is out of range or the string is not found, then None is returned. Hint: I found `enumerate()` useful when `choice` is a string. Also, `type`() is useful when checking the type of a variable.

4. **`add_moves (pokemon, moves_list)` ---→ Boolean**
   This function first adds one random move to the `pokemon`'s move list, then adds three more moves that match one of the elements of this `pokemon`. Each move is to be added using this `pokemon`'s object's `add_move()` method. Find the first random move using a random integer to index the `moves_list` parameter. Then get three more random moves from `moves_list`, but add them to this `pokemon` only if the random move's element matches either element1 or element2 of this `pokemon` and the move is not already in this `pokemon`'s move list. Hint: remember to ensure that your random index is in the correct range. Also, do not change the moves_list in this function because it is passed by reference.
   What if four valid moves do not exist? Since we are randomly selecting moves it is difficult and time consuming to know if we have tried all possibilities so we will take the easy way and if we have not found four valid moves in 200 attempts, we will give up and

return `False`. Otherwise, if we were able to add all moves to the `pokemon`, this function return `True`.

5. **`turn (player_num, player_pokemon, opponent_pokemon) --->` Boolean**
   Player_num is an int for printing which player: 1 or 2
   Player_pokemon, the attacker, and opponent_pokemon are of type Pokemon
   Player Turn steps
   a. Print `"Player {}'s turn"` and print the player's pokemon.
   b. Prompt for index into attacker's list of moves or show request or quit.
      If the player requests show ('show ele', 'show pow', 'show acc') the appropriate value will be displayed followed by a re-prompt.
      If 'q', print `"Player {} quits, Player {} has won the pokemon battle!"` and end battle. return `False`.
   c. Print the selected move.
   d. Print the opponent's health.
   e. Attack (call the player's attack method).
   f. Print the opponent's health again.
   g. If opponent's health is less than or equal to zero, the player wins.
      print `"Player {}'s pokemon fainted, Player {} has won the pokemon battle!"` . return `False`.
   h. return `True`

6. **`main()`**
   The main function. This function simulates a 1-on-1 pokemon battle between Player 1 and Player 2.
   a- First the relevant moves and pokemon should be added to respective lists using the `read_file` functions. Note that we are opening hard-coded files `"moves.csv"` and `"pokemon.csv"`. The files `"moves.csv"` and `"pokemon.csv"` will be used in the function tests.
   b- The user is asked if he would like to have a pokemon battle and keep prompting until a valid input is entered. The only valid responses are 'Y', 'y', 'N', 'n', 'Q', or 'q'.
   c- If yes, each player picks a `pokemon`(use previously defined functions as needed). Keep asking the player for a valid pokemon name or index.
   d- If valid, Four moves are added to each `pokemon` using the `add_moves()` function.
   e- Finally, the opponents battle until one of the player's pokemon faints or one player quits with the `'Q' or 'q'` command.
   f- After a battle ends the user is prompted if they would like to have another battle. If the user responds with `'Y' or 'y'`, then a new battle ensues. If the user responds with `'N', 'n', 'Q', or 'q'` then the program finishes execution. All other inputs are invalid.

Assignment Deliverables

The deliverables for this assignment are the following two files:

`proj11.py` – the source code for your Python program
`pokemon.py` – the source code for the Pokemon and Move classes

Assignment Notes
Different from real Pokemon:

- For those of you knowledgeable about Pokemon you will notice a few things missing. The speed stat is not utilized.
- Moves that take more than one turn to perform, have multiple hits and have cooldowns are simplified to single hit, single turn moves.
- Status effect moves, moves that lower stats, moves that change terrain and recovery/protection moves are also not utilized.
- Finally, there are no critical hits or levels to take into consideration.

All of these factors are to contribute to the simplicity of the project as the focus here is on classes.

Avoid two instances of the same ID:
When working with the `read_file_pokemon()` method, be wary of pokemon with mega forms. Their ID will appear more than once in the file; however, we only want the first instance of the pokemon. For example, ID 3 contains the pokemon name `"Venusaur"`, another instance of ID 3 also contains the name `"VenusaurMega Venusaur"`. We only want the pokemon that appears with the first instance of the ID. One way to work with this is to utilize a `set` of IDs.

First generation only:
The element and pokemon lists are created from the first generation only for this project.

Remember to use `deepcopy`:
When selecting a pokemon from the pokemon list, use `deepcopy`. If you assign a variable equal to one of the pokemon, then that variable is a reference to the pokemon from the list and not a copy. Think of it like this. Let's say we both choose the pokemon Pikachu. If we do not use deepcopy, then my Pikachu and your Pikachu are the same Pikachu. If we do use deepcopy then my Pikachu is different from your Pikachu and their moves can be different.

Test Cases

Test 1

```
Would you like to have a pokemon battle? Nope

Invalid option! Please enter a valid choice: Y/y, N/n or Q/q: quit

Invalid option! Please enter a valid choice: Y/y, N/n or Q/q: q
Well that's a shame, goodbye
```

Test 2

```
 Would you like to have a pokemon battle? y
Player 1, choose a pokemon by name or index: Bulbasaur
pokemon1:
bulbasaur      45             49             49             65             65
grass          poison

Player 2, choose a pokemon by name or index: Mewtwo
pokemon2:
mewtwo         106            110            90             154            90
psychic

Player 1's turn
bulbasaur      45             49             49             65             65
grass          poison
slam           smog           petal-dance    razor-leaf
Show options: 'show ele', 'show pow', 'show acc'
Select an attack between 1 and 4 or show option or 'q': show pow
80             30             120            55
Show options: 'show ele', 'show pow', 'show acc'
Select an attack between 1 and 4 or show option or 'q': 3
selected move: petal-dance

mewtwo hp before:106
mewtwo hp after:51

Player 2's turn
mewtwo         51             110            90             154            90
psychic
clamp          dream-eater    confusion      psychic
Show options: 'show ele', 'show pow', 'show acc'
Select an attack between 1 and 4 or show option or 'q': show ele
water          psychic        psychic        psychic
Show options: 'show ele', 'show pow', 'show acc'
Select an attack between 1 and 4 or show option or 'q': 4
selected move: psychic

bulbasaur hp before:45
It's super effective!!!!
bulbasaur hp after:0

Player 1's pokemon fainted, Player 2 has won the pokemon battle!
Battle over, would you like to have another? n
Well that's a shame, goodbye
```

Test 3

```
Would you like to have a pokemon battle? Y
Player 1, choose a pokemon by name or index: charmander
pokemon1:
charmander     39             52             43             60             50
fire

Player 2, choose a pokemon by name or index: 4
pokemon2:
charmeleon     58             64             58             80             65
fire

Player 1's turn
charmander     39             52             43             60             50
fire
slam           fire-blast     fire-spin      ember
Show options: 'show ele', 'show pow', 'show acc'
Select an attack between 1 and 4 or show option or 'q': 4
selected move: ember

charmeleon hp before:58
Not very effective...
charmeleon hp after:46

Player 2's turn
charmeleon     46             64             58             80             65
fire
bite           fire-blast     flamethrower   fire-spin
Show options: 'show ele', 'show pow', 'show acc'
Select an attack between 1 and 4 or show option or 'q': 2
selected move: fire-blast

charmander hp before:39
Not very effective...
charmander hp after:0

Player 1's pokemon fainted, Player 2 has won the pokemon battle!
Battle over, would you like to have another? Y
Player 1, choose a pokemon by name or index: Machamp
pokemon1:
machamp        90             130            80             65             85
fighting

Player 2, choose a pokemon by name or index: Golem
pokemon2:
golem          80             120            130            55             65
rock           ground

Player 1's turn
machamp        90             130            80             65             85
fighting
egg-bomb       high-jump-kick jump-kick      rolling-kick
Show options: 'show ele', 'show pow', 'show acc'
Select an attack between 1 and 4 or show option or 'q': show ele
normal         fighting       fighting       fighting
Show options: 'show ele', 'show pow', 'show acc'
Select an attack between 1 and 4 or show option or 'q': show acc
75             90             95             85
Show options: 'show ele', 'show pow', 'show acc'
Select an attack between 1 and 4 or show option or 'q': show pow
100            130            100            60
Show options: 'show ele', 'show pow', 'show acc'
Select an attack between 1 and 4 or show option or 'q': 2
```

```
selected move: high-jump-kick

golem hp before:80
It's super effective!!!!
golem hp after:0

Player 2's pokemon fainted, Player 1 has won the pokemon battle!
Battle over, would you like to have another? N
Well that's a shame, goodbye
```

Test 4

```
Would you like to have a pokemon battle? y
Player 1, choose a pokemon by name or index: pikachu
pokemon1:
pikachu        35             55             40             50             50
electric

Player 2, choose a pokemon by name or index: 18
pokemon2:
pidgeot        83             80             75             70             70
normal         flying

Player 1's turn
pikachu        35             55             40             50             50
electric
slam           thunder-punch  thunderbolt    thunder
Show options: 'show ele', 'show pow', 'show acc'
Select an attack between 1 and 4 or show option or 'q': 3
selected move: thunderbolt

pidgeot hp before:83
It's super effective!!!!
pidgeot hp after:0

Player 2's pokemon fainted, Player 1 has won the pokemon battle!
Battle over, would you like to have another? y
Player 1, choose a pokemon by name or index: Rhydon
pokemon1:
rhydon         105            130            120            45             45
ground         rock

Player 2, choose a pokemon by name or index: 18
pokemon2:
pidgeot        83             80             75             70             70
normal         flying

Player 1's turn
rhydon         105            130            120            45             45
ground         rock
bubble         rock-throw     bone-club      dig
Show options: 'show ele', 'show pow', 'show acc'
Select an attack between 1 and 4 or show option or 'q': show ele
water          rock           ground         ground
Show options: 'show ele', 'show pow', 'show acc'
Select an attack between 1 and 4 or show option or 'q': 3
selected move: bone-club

pidgeot hp before:83
No effect!
pidgeot hp after:83

Player 2's turn
pidgeot        83             80             75             70             70
normal         flying
wrap           horn-attack    self-destruct  egg-bomb
Show options: 'show ele', 'show pow', 'show acc'
Select an attack between 1 and 4 or show option or 'q': show ele
normal         normal         normal         normal
Show options: 'show ele', 'show pow', 'show acc'
Select an attack between 1 and 4 or show option or 'q': 4
selected move: egg-bomb

rhydon hp before:105
```

```
Not very effective...
rhydon hp after:84

Player 1 hp after: 84
Player 2 hp after: 83

Player 1's turn
rhydon         84             130            120            45             45
ground         rock
bubble         rock-throw     bone-club      dig
Show options: 'show ele', 'show pow', 'show acc'
Select an attack between 1 and 4 or show option or 'q': 4
selected move: dig

pidgeot hp before:83
No effect!
pidgeot hp after:83

Player 2's turn
pidgeot        83             80             75             70             70
normal         flying
wrap           horn-attack    self-destruct  egg-bomb
Show options: 'show ele', 'show pow', 'show acc'
Select an attack between 1 and 4 or show option or 'q': 1
selected move: wrap

rhydon hp before:84
Not very effective...
rhydon hp after:80

Player 1 hp after: 80
Player 2 hp after: 83

Player 1's turn
rhydon         80             130            120            45             45
ground         rock
bubble         rock-throw     bone-club      dig
Show options: 'show ele', 'show pow', 'show acc'
Select an attack between 1 and 4 or show option or 'q': q
Player 1 quits, Player 2 has won the pokemon battle!
Battle over, would you like to have another? n
Well that's a shame, goodbye
```

Grading Rubric

Computer Project #_
Scoring Summary

General Requirements:
(5 pts) Coding Standard 1-9
(descriptive comments, mnemonic identifiers, format, etc…)

Implementations:
(4 pts) Move class
(8 pts) Pokemon class (not including attack method)
(8 pts) attack method (Pokemon class)
(2 pts) read_file_moves
(2 pts) read_file_pokemon
(3 pts) choose_pokemon function
(4 pts) add_moves function
(4 pts) Test 1
(5 pts) Test 2
(5 pts) Test 3
(5 pts) Test 4