CSE101

DUE DEC 07

23:59 KST

ASSIGNMENT V

PROFESSOR

FRANCOIS

RAMEAU

OOP

General instructions

- Please add your name and email in the dedicated location at the top of your code
- Do not use any external library, except when explicitly requested
- Try to follow the naming convention proposed by PEP-8 seen in class
- Use ONLY what we have already seen in class. If not, you will get ZERO points
- Use meaningful names for your variables and functions
- If you face a problem submitting your code via GitHub, please contact the professor and the TA by email
- Note that the received code will be tested on a classifier to detect potential usage of Large Language Models. We will also pay particular attention to plagiarism
- Leave comments in your code to explain your code and describe the difficulties you faced

INVITATION LINK

https://classroom.github.com/a/hLa9Vqps

In this assignment, we will create a turned-based game simulating a Pokémon fight. Our game will consist of two Pokémons with various types, of moves. Each Pokémon will have different attributes, including level and health, which will affect the outcome of the fight.

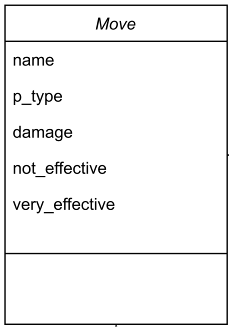
The fight will take place in an arena where the Pokémons will take turns attacking each other. The damage each Pokémon deals will be determined by their moves' damage, their level, and a bit of luck.

Your task is to implement the **Move**, **Pokemon**, and **Arena** classes to create a fight simulation. This assignment will give you a chance to apply object-oriented programming concepts and see how they can be used to create an interactive program. Note that we will not use any inheritance in this homework (it is for simplicity, but inheritance could be very interesting for such a task).

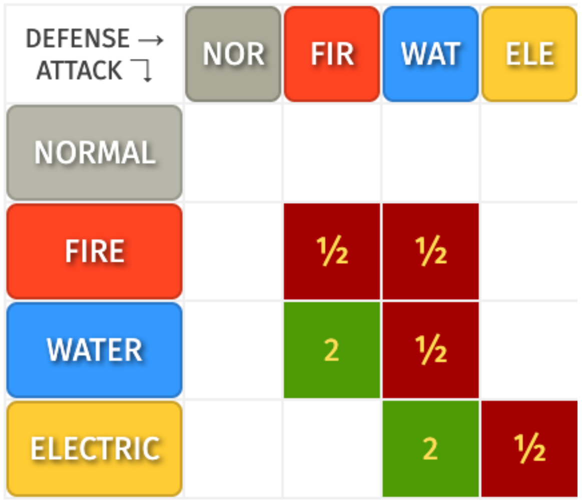
Part I: The Moves (2 points)

The very first thing your Pokemons will need is their "moves" to attack opponents. Therefore, we will start creating a class `Move`.

Each of these moves has a `name` and `damage` associated with it. But, as you probably know, in Pokemon, each move is associated with a `p_type` (like "Fire", "Water", or "Electric"), and its effectiveness can change depending on the opponent's type. For instance, a water move will deal much more damage if it is used against a Fire Pokémon (twice more damage!). However, a Fire attack will deal poor damage on a Water Pokémon (Half damage).
To encompass all these properties, our Move class will contain the following attributes:



Note that the attributes `not_effective` and `very_effective` depend on the `p_type`. For instance, if the `p_type` of the move is "Electric," then its `not_effective` list will contain every `p_type` against which it is poorly effective: `["electric"]`, and the `very_effective` list will contain all the Pokémon types against which it is very effective: `["Water"]`.
To simplify the implementation of this code, we will restrict ourselves to only 4 possible types of Pokémon: Normal, Fire, Water, and Electric. To create your lists of `not_effective` and `very_effective` depending on the type of your move, please refer to the following diagram:



DEFENSE → ATTACK ↴	NOR	FIR	WAT	ELE
NORMAL				
FIRE		½	½	
WATER		2	½	
ELECTRIC			2	½

Note that the class Move does not contain any methods; it just holds the properties of each Pokémon move. Now that you have implemented all the attribute of this class, you will prepare your moves in the main as follow:

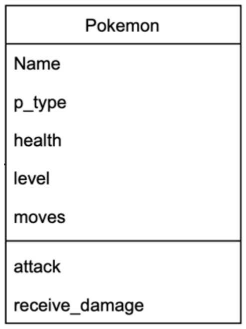
```
# Create moves for fire:
ember = Move("Ember", "Fire", 40)
fiery_dance = Move("Fiery Dance", "Fire", 80)
blaze_kick = Move("Blaze Kick", "Fire", 85)
# Create moves for water:
clamp = Move("Clamp", "Water", 35)
dive = Move("Dive", "Water", 80)
bubble_beam = Move("Bubble Beam", "Water", 65)
```

Be creative and prepare more moves for your Pokemons!

What should you implement for the Move class?

1. An __init__ method that initializes all the attributes	2 points
2. Declare multiple Moves in the main	0 point

Part II: The Pokémon (4 points)



Now that we have the moves ready, it's time to create the Pokémons themselves. A Pokémon in our game is created using our **Pokemon** class, which has certain **attributes** and **methods**.

1. Attributes of the Pokemons (2 points)

Each Pokémon has a name, a type, health points, a level, and a set of moves

- The **name** attribute is a string representing the Pokémon's name. The name is provided by the programmer when initializing the Pokémon
- The **health** attribute is an integer initialized with an integer value for each Pokémon, representing the Pokémon's current health points.
- The **level** attribute specifies the level of the Pokémon which will impact the quantity of damage the Pokémon can inflict

- The **moves** attribute is a list containing multiple instances of the **Move** class we have previously defined. The programmer will assign a list of moves to his Pokémon during instantiation.

Thus, the definition of your **__init__** method should look like this:

```
def __init__(self, name, p_type, health, level, moves):
```

After your Pokémon class is declared with the proper attributes in the **__init__** method, you will instantiate multiple Pokémons in the main part of the code (you can create your own Pokémons if you are a fan):

```
squirtle = Pokemon("Squirtle", "Water", 100, 5, [clamp, dive, bubble_beam])
charmander = Pokemon("Charmander", "Fire", 90, 4, [ember, fiery_dance, blaze_kick])
```

What should you implement for the Pokémon class attribute?

1.	An __init__ method that initializes all the attributes mentioned	2 point
2.	Declare multiple Pokémon in the main	0 point

2. Methods of the Pokemons (2 points)

Your Pokémon now have some moves and are in good health; they are ready for action! The behavior of your Pokémon is simple; they can either attack an opponent or receive the damage. These two behaviors will be defined in two different methods called respectively **attack()** and **receive_damage()**.

II.2.1 - The attack method

The **attack** method calculates the damage dealt by the Pokémon and applies it to the opponent Pokémon receiving the damage.

But before attacking, the trainer (you!) can select what move he wants to use. Therefore, every time the attack method is used on a Pokémon a menu containing the different moves available for this Pokémon should be displayed in the terminal as such (use the **input()** function):

```
Choice your attack :
0. Clamp
1. Dive
2. Bubble Beam
Select your move:
```

After the trainer selects what move to perform, the damage inflicted on the opponent Pokémon is computed as follows:

$$damage = move_{damage} * level * luck * damage_{multiplier},$$

Where:

- $move_{damage}$ is the damage value of the selected move.
- $level$ is the level of the attacking Pokémon.
- $luck$ is a random number between 0 and 1 to add variability to the attack.
- $damage_{multiplier}$ adjusts the damage based on type effectiveness:
    - 0.5 if the opponent's type is in the move's `not_effective` list.
    - 2 if the opponent's type is in the move's `very_effective` list.
    - 1 otherwise (neutral effectiveness).

After calculating this damage value, use the method `receive_damage()` on your opponent to apply this damage to it! This method is described just below.

II.2.1 - The receive_damage method

The damage computed for the Pokémon's attacker can be used to inflict damage on the other Pokémon. This process is implemented in the `receive_damage()` method. This method takes the incoming damage and subtracts it from the Pokémon's health: $health = health - damage$

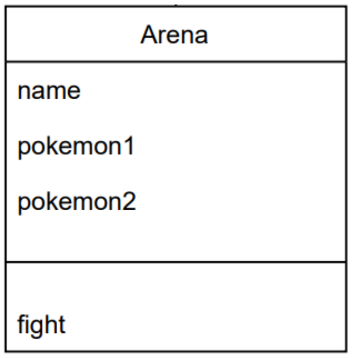
After applying the change to `self.health` , if its value is lower than zero, you can print "*pokemon.name* has fainted!"

What should you implement for the Pokemon class methods?

1.	Define an attack() method with the behavior described	1 point
2.	Define an received_damage() method with the behavior described	1 point

In the next section, we will place our Pokémons into an Arena and let the fight begin!

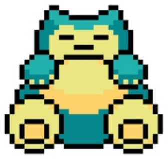
Part III: The Arena (4 points)



The final piece of our Pokemon fight simulation is the Arena. The Arena is where our Pokémons will engage in their battle. It's represented by the `Arena` class, which has its own attributes and methods.

1. Attributes of the arena (1 point)

An Arena instance has two main attributes: a `name` and `pokemon1` and `pokemon2`, representing the two Pokémons who will fight in this Arena.



What should you implement for the Arena class attribute?

1.	An __init__ method that initializes the three attributes mentioned	1 point
2.	Declare an arena in the main (already done)	0 point

2. Method of the arena (3 points)

The Arena class has one main method: `fight`().

The `fight` method initiates a turn-based fight between the two Pokémons. The Pokémons take turns attacking each other until one of them has no more health points (i.e., their health reaches 0 or below); you might need a while loop here. The method should print out a message for each attack, showing who attacked whom and how much damage was dealt (it can be integrated directly into the `attack` and `receive_damage` methods of the Pokémons for convenience). Once a Pokémon's health points reach 0 or less, the fight ends, and the method prints out the name of the winner.

What should you implement for the Arena class method?

1.	Implement the method fight() in the arena class	3 points
2.	Run a fight in the main (already done)	0 point

With this, we have all the pieces in place for our Pokémon Battle Simulation. Your final task is to create a few Pokémons, equip them with Moves, place them in an Arena, and start a fight.

Part IV: Extra Bonus!!! (0.5 points to compensate for some Quizzes/homeworks)

Complete the course evaluation survey and attach the screen capture of completion. The screenshot should clearly show your name, course (CSE101), and the message that states completion of the survey. CAUTION: Do NOT include any information that can be used to identify your responses.

→ Send the screenshot directly via email to: youngwon.choi@stonybrook.edu