Classes Challenge 38: Pykemon Simulation App

Description:
You will be responsible for writing a program the emulates playing the hit game Pokemon. Your program will generate Pykemon creatures randomly. Each Pykemon creature will be one of three different elemental types: fire, water, or grass. Each Pykemon type will have its own set of unique moves and each individual Pykemon will have it's one name, health stat, and speed stat.You will be given one Pykemon and then be tasked with fighting other pykemon until you run out of health. In the original Pokemon, the user is presented with three Pokemon to choose from at the start of the game; one of each elemental type. Pykemon is no different. You will choose your starting Pykemon and then be off on a journey to battle other wild Pykemon until your Pykemon faints.

Step By Step Guide:
Defining your classes

- Define a class Pykemon()
    - This will be a parent class in which the child classes, Fire, Water, and Grass will inherit from.
- Define a class Fire()
    - A child of the Pykemon class.
- Define a class Water()
    - A child of the Pykemon class.
- Define a class Grass()
    - A child of the Pykemon class.
- Define a class Game()

Defining your class methods
The Pykemon Class

- Define an __init__() method for the Pykemon class which takes five parameters, the required self parameter, a name, an element, a health, and a speed.
    - Initialize an attribute name and set it equal to the given name in title case.
    - Initialize an attribute element and set it equal to the given element.
    - Initialize an attribute current_health and set it equal to the given health.
    - Initialize an attribute max_health and set it equal to the given health.
    - Initialize an attribute speed and set it equal to the given speed.
    - Initialize an attribute is_alive and set it equal to True.

- Define a light_attack() method for the Pykemon class which takes two parameters, the required self parameter and an enemy pykemon.
    - The light attack will be guaranteed to do minimal damage to the enemy Pykmeon. This light attack will have a different name, depending on the elemental type of the pokemon. However, all light attacks will appear in a list called moves, which is an attribute of a specific Pykemon, at index 0.
    - Create a random integer from 15 to 25 and store it in a variable damage.

        - Type import random as the first line of code in your program.
    - Print a message stating that the Pykemon performed a specific attack.
        - Use the Pykemon's name and the name of the move.
        - self.name and self.moves[0]
    - Print a message stating how much damage was dealt.
    - Subtract that damage from the enemy Pykemon's current_health attribute.

- Define a heavy_attack() method for the Pykemon class which takes two parameters, the required self parameter and an enemy pykemon.
    - The heavy attack has the potential to deal massive damage to the enemy Pykemon but it could also deal no damage at all; it is risky. This heavy attack will have a different name, depending on the element type of the pokemon. However, all heavy attacks will appear in a list called moves, which is an attribute of a specific Pykemon at index 1.
    - Create a random integer from 0 to 50 and store it in a variable called damage.
    - Print a message stating that the Pykemon performed a specific attack.
        - Use the Pykemon's name and the name of the move.
        - self.name and self.moves[1]
    - If the damage was less than 10:
        - Print a message stating the attack missed.
        - Deal no damage.
    - Else:
        - Print a message stating how much damage was dealt.
        - Subtract that damage from the enemy Pykemon's current_health attribute.

- Define a restore() method for the Pykemon class which takes one parameter; the required self parameter.
    - The restore move doesn't deal any damage at all but instead restores a small portion of health to the Pykemon calling the move. This restore move will have a different name, depending on the element type of the pokemon. However, all restore moves will appear in a list called moves, which is an attribute of a specific Pykemon at index 2.
    - Create a random integer from 15 to 25 and store it in a variable called heal.
    - Print a message stating that the Pykemon performed a specific attack.
        - Use the Pykemon's name and the name of the move.
        - self.name and self.moves[2]
    - Print a message stating how many points the Pykemon is healed by.
    - Add that heal value to the Pykemon's current_health attribute.
    - If the value of the Pykemon's current_health attribute is greater than the Pykemon's max_health attribute:
        - Set the current_health attribute equal to the max_health attribute.

- Define a faint() method for the Pykemon class which takes one parameter, the required self parameter.

    - If the Pykemon's current_health attribute is less than or equal to zero:
        - Set the Pykemon's is_alive attribute to False.
        - Print a message stating that the Pykemon has fainted.
        - Prompt the user to press enter to continue.

- Define a show_stats() method for the Pykemon class which takes one parameter, the required self parameter.
    - Print the Pykemon's name.
    - Print the Pykemon's element type.
    - Print the Pykemon's current health and maximum health.
    - Print the Pykemon's speed.

The Fire Class

- Define an __init__() method for the Fire class which takes five parameters, the required self parameter, a name, and element, a health, and a speed.
    - Call the super function to inherit from the parent Pykemon class.
        - Pass the correct arguments required by the Pykemon class: name, element, health, and speed.
    - Initialize an attribute called moves and set it equal to a list containing the following four strings: 'Scratch', 'Ember', 'Light', 'Fire Blast'
        - These are the light attack, heavy attack, restore move, and special attack for all fire Pykemon.

- Define a special_attack() method for the Fire class which takes two parameters, the required self parameter and an enemy Pykemon.
    - The special attack deals massive damage to grass type Pykemon, normal damage to fire type Pykemon, and minimal damage to water type Pykemon. This special attack will have a different name, depending on the element type of the pokemon. However, all special attacks will appear in a list called moves, which is an attribute of a specific Pykemon at index 3.
    - Print a message stating that the Pykemon performed a specific attack.
        - Use the Pykemon's name and the name of the move.
        - self.name and self.moves[3]
    - If the enemy Pykemon's element attribute is GRASS:
        - Print a message stating that the move is super effective.
        - Create a random integer between 35 and 50 and store it in a variable damage.
    - Elif the enemy Pykemon's element attribute is WATER:
        - Print a message stating that the move is not very effective.
        - Create a random integer between 5 and 10 and store it in a variable damage.
    - Else:
        - Create a random integer between 10 and 30 and store it in a variable damage.
    - Print a message stating how much damage was dealt.

    - Subtract that damage from the enemy Pykemon's current_health attribute.

- Define a move_info() method for the Fire class which takes one parameter, the required self parameter.
    - Print a message header for the Pykemon's moves.
    - Print the name of each move and a brief description.
        - Moves will be stored in the moves lists.
        - See example output for formatting options.

The Water Class

- Define an __init__() method for the Water class which takes five parameters, the required self parameter, a name, and element, a health, and a speed.
    - Call the super function to inherit from the parent Pykemon class.
        - Pass the correct arguments required by the Pykemon class: name, element, health, and speed.
    - Initialize an attribute called moves and set it equal to a list containing the following four strings: 'Bite', 'Splash', 'Dive', 'Water Cannon'
        - These are the light attack, heavy attack, restore move, and special attack for all water Pykemon.

- Define a special_attack() method for the Water class which takes two parameters, the required self parameter and an enemy Pykemon.
    - The special attack deals massive damage to fire type Pykemon, normal damage to water type Pykemon, and minimal damage to grass type Pykemon. This special attack will have a different name, depending on the element type of the pokemon. However, all special attacks will appear in a list called moves, which is an attribute of a specific Pykemon at index 3.
    - Print a message stating that the Pykemon performed a specific attack.
        - Use the Pykemon's name and the name of the move.
        - self.name and self.moves[3]
    - If the enemy Pykemon's element attribute is FIRE:
        - Print a message stating that the move is super effective.
        - Create a random integer between 35 and 50 and store it in a variable damage.
    - Elif the enemy Pykemon's element attribute is GRASS:
        - Print a message stating that the move is not very effective.
        - Create a random integer between 5 and 10 and store it in a variable damage.
    - Else:
        - Create a random integer between 10 and 20 and store it in a variable damage.
    - Print a message stating how much damage was dealt.
    - Subtract that damage from the enemy Pykemon's current_health attribute.

- Define a move_info() method for the Water class which takes one parameter, the required self parameter.
    - Print a message header for the Pykemon's moves.
    - Print the name of each move and a brief description.
        - Moves will be stored in the moves lists.
        - See example output for formatting options.

The Grass Class

- Define an __init__() method for the Grass class which takes five parameters, the required self parameter, a name, and element, a health, and a speed.
    - Call the super function to inherit from the parent Pykemon class.
        - Pass the correct arguments required by the Pykemon class: name, element, health, and speed.
    - Initialize an attribute called moves and set it equal to a list containing the following four strings: 'Vine Whip', 'Wrap', 'Grow', 'Leaf Blade'
        - These are the light attack, heavy attack, restore move, and special attack for all grass Pykemon.

- Define a special_attack() method for the Grass class which takes two parameters, the required self parameter and an enemy Pykemon.
    - The special attack deals massive damage to water type Pykemon, normal damage to grass type Pykemon, and minimal damage to fire type Pykemon. This special attack will have a different name, depending on the element type of the pokemon. However, all special attacks will appear in a list called moves, which is an attribute of a specific Pykemon at index 3.
    - Print a message stating that the Pykemon performed a specific attack.
        - Use the Pykemon's name and the name of the move.
        - self.name and self.moves[3]
    - If the enemy Pykemon's element attribute is WATER:
        - Print a message stating that the move is super effective.
        - Create a random integer between 35 and 50 and store it in a variable damage.
    - Elif the enemy Pykemon's element attribute is FIRE:
        - Print a message stating that the move is not very effective.
        - Create a random integer between 5 and 10 and store it in a variable damage.
    - Else:
        - Create a random integer between 10 and 20 and store it in a variable damage.
    - Print a message stating how much damage was dealt.
    - Subtract that damage from the enemy Pykemon's current_health attribute.

- Define a move_info() method for the Grass class which takes one parameter, the required self parameter.
    - Print a message header for the Pykemon's moves.

    - Print the name of each move and a brief description.
        - Moves will be stored in the moves lists.
        - See example output for formatting options.

The Game Class

- Define an __init__() method for the Game class that takes one parameter, the required self parameter.
    - Initialize an attribute pykemon_elements and set it equal to a list containing the strings ‘FIRE’, ‘WATER’, and ‘GRASS’.
    - Initialize an attribute called pykemon_names and set it equal to a list containing strings that represent various Pykemon names.
        - For this challenge the following names were used: 'Chewdie', 'Spatol', 'Burnmander', 'Pykachu', 'Pyonx', 'Abbacab', 'Sweetil', 'Jampot', 'Hownstooth', 'Swagilybo', 'Muttle', 'Zantbat', 'Wiggly Poof', 'Rubblesaur'
    - Initialize an attribute called battles_won and set it equal to zero.
        - This will keep track of how many Pykemon you have defeated.

- Define a create_pykemon() method for the Game class which takes one parameter, the required self parameter.
    - Create a variable health and set it equal to a random integer from 70 to 100.
    - Create a variable speed and set it equal to a random integer from 1 to 10.
    - Create a variable element and set it equal to a random element from the Game classes’ pykemon_elements attribute.
    - Create a variable name and set it equal to a random name from the game classes’ pykemon_names attribute.
    - If element is equal to “FIRE”:
        - Create a Fire Pykemon passing the name, elmement, health, and speed.
    - Elif element is equal to “WATER:
        - Create a Water Pykemon passing the name, element, health, and speed.
    - Else:
        - Create a Grass pykemon passing the name, element, health, and speed.
    - Return the created pykemon.

- Define a choose_pykemon() method for the Game class which takes one parameter, the required self parameter.
    - Create a variable starters and set it equal to a blank list.
    - While the length of the list starters is less than 3:
        - Create a Pykemon by calling the Game classes’ create_pykemon method.
        - Create a variable valid_pykemon and set it equal to True.
        - Loop through the list starters. For each iteration:
            - Check if the given starter’s name is equal to the currently created Pykemon’s name or if the starter’s element is equal to the currently created Pykemon’s element. We do not want repeated Pykemon names or elements in our starter options.

            - If they are repeating, set valid_pykemon equal to False.
        - After the loop has run, if valid_pykemon is still True:
            - Append the currently created Pykemon to the list starters as it is a unique Pykemon.
    - Loop through the list starters. For each iteration:
        - Show the given Pykemon's stats by calling the Pykemon's show_stats() method.
        - Show the given Pykemon's moves by calling the Pykemon's move_info() method.
    - Print a message stating that the Professor presents three Pykemon.
    - Print a message that informs the user to press 1 for the first Pykemon, 2 for the second Pykemon, and 3 for the third Pykemon.
        - Use the Pykemon's name.
    - Get user input for their choice.
    - Create a variable pykemon and set it equal to the users choice.
        - Their choice 1, 2, or 3 corresponds to an index in the list starters 0, 1, or 2.
    - Return chosen pykemon.

- Define a get_attack() method for the Game class which takes two parameters, the required self parameter and a Pykemon.
    - Print a message asking the user what move they would like.
    - Print a message stating the user should press 1 for the light attack, 2 for the heavy attack, 3 for the restore move, and 4 for the special attack.
        - Use the Pykemon specific move names.
    - Ge user input for their choice.
    - Print a blank line
    - Print a line of dashes for formatting reasons "----------"
    - Return the users choice.

- Define a player_attack() method for the Game class which takes four parameters, the required self parameter, a move, a player, and a computer.
    - If the given move equals 1:
        - Call the player's light_attack() method.
    - Elif the given move equals 2:
        - Call the player's heavy_attack() method.
    - Elif the given move equals 3:
        - Call the player's restore() method.
    - Elif the given move equals 4:
        - Call the player's special_attack() method.
    - Check if the computer fainted by calling the computer's faint() method.

- Define a computer_attack() method for the Game class which takes three parameters, the required self parameter, a player, and a computer.
    - Create a variable move and set it equal to a random integer from 1 to 4.

        - If the given move equals 1:
            - Call the computer's light_attack() method.
        - Elif the given move equals 2:
            - Call the computers's heavy_attack() method.
        - Elif the given move equals 3:
            - Call the computer's restore() method.
        - Elif the given move equals 4:
            - Call the computer's special_attack() method.
        - Check if the player fainted by calling the player's faint() method.

- Define a battle() method for the Game class which takes three parameters, the required self parameter, a player, and a computer.
    - Get the players move by calling the Game classes' get_attack() method.
    - If the player's speed is greater than or equal to the computer's speed:
        - Let the player attack by calling the Game classes' player_attack() method.
        - If the computer is still alive after the attack:
            - Let the computer attack by calling the Game classes' computer_attack() method.
    - Else:
        - Let the computer attack by calling the Game classes' computer_attack() method.
        - If the player is still alive after the attack:
            - Let the player attack by calling the Game classes' player_attack() method.

The main code

- Print a welcome summary about Pykemon.
    - This summary should include a narrative about how a professor is going to give you a Pykemon.
- Prompt the user to press enter to continue.

- Create an active variable playing_main and set it equal to True.
- Use this variable to control a while loop.
    - Create a Game object called game.
    - Allow the player to choose their Pykemon by calling the game object's choose_pykemon() method.
    - Print a message summarizing the Pykemon chosen.
    - Prompt the user to press enter to continue.

    - While the player's pykemon is alive:
        - Create a Pykemon for the computer by calling the game object's create_pykemon() method.
        - Print a message stating the computer Pykemon's name.
        - Show the stats of the computer Pykemon by calling the show_stats() method.

        - While the computer Pykemon is alive and while the player Pykemon is alive:
            - Simulate battle by calling the game objects battle() method.
            - If the computer Pykemon is alive and if the player Pykemon is alive:
                - Show the player's stats by calling the show_stats() method.
                - Show the computer's stats by calling the show_stats() method.
                - Print a line of dashes for formatting reasons "----------"
        - If the player is alive:
            - Increment the game object's kills attribute by 1.

    - Print a message stating your Pykmeon has fainted.
        - Use the Pykemon's name.
    - Print a message stating how many Pykemon they defeated.
    - R
    - Get user input for whether they would like to play again.
    - If they do not want to play again:
        - Set playing_main equal to False.
        - Print a message thanking the user for playing.

Example Output:

Welcome to Pykemon!
Can you become the worlds greatest Pykemon Trainer???

Don't worry! Prof Eramo is here to help you on your quest.
He would like to gift you your first Pykemon!
Here are three potential Pykemon partners.
Press Enter to choose your Pykemon!

Name: Abbacab
Element Type: GRASS
Health: 92 / 92
Speed: 5

Abbacab Moves:
-- Vine Whip --
  An efficient attack...
  Guaranteed to do damage within the range of 15 to 25 damage points.

-- Wrap --

A risky attack...
Could deal up to 50 damage points or as little as 0 damage points.

-- Grow --
A restorative move...
Guaranteed to heal your Pykemon 15 to 25 health points.

-- Leaf Blade --
A powerful GRASS based attack...
Guaranteed to deal MASSIVE damage to WATER type Pykemon.

Name: Pyonx
Element Type: WATER
Health: 84 / 84
Speed: 3

Pyonx Moves:
-- Bite --
An efficient attack...
Guaranteed to do damage within the range of 15 to 25 damage points.

-- Splash --
A risky attack...
Could deal up to 50 damage points or as little as 0 damage points.

-- Dive --
A restorative move...
Guaranteed to heal your Pykemon 15 to 25 health points.

-- Water Cannon --
A powerful WATER based attack...
Guaranteed to deal MASSIVE damage to FIRE type Pykemon.

Name: Spatol
Element Type: FIRE
Health: 94 / 94
Speed: 5

Spatol Moves:
-- Scratch --
An efficient attack...
Guaranteed to do damage within the range of 15 to 25 damage points.

-- Ember --
A risky attack...

Could deal up to 50 damage points or as little as 0 damage points.

-- Light --
A restorative move...
Guaranteed to heal your Pykemon 15 to 25 health points.

-- Fire Blast --
A powerful FIRE based attack...
Guaranteed to deal MASSIVE damage to GRASS type Pykemon.

Professor Eramo presents you with three Pykemon:
(1) - Abbacab
(2) - Pyonx
(3) - Spatol
Which Pykemon would you like to choose: 3

Congratulations Trainer, you have chosen Spatol!

Your journey with Spatol beings now...Press Enter

OH NO! A wild Muttle has approached!

Name: Muttle
Element Type: GRASS
Health: 93 / 93
Speed: 2

What would you like to do....
(1) - Scratch
(2) - Ember
(3) - Light
(4) - Fire Blast
Please enter your move choice: 2

Pykemon Spatol used Ember.
It dealt 31 damage.
Pykemon Muttle used Vine Whip.
It dealt 15 damage.

Name: Spatol
Element Type: FIRE
Health: 79 / 94
Speed: 5

Name: Muttle
Element Type: GRASS
Health: 62 / 93
Speed: 2

What would you like to do....
(1) - Scratch
(2) - Ember
(3) - Light
(4) - Fire Blast
Please enter your move choice: 1

Pykemon Spatol used Scratch.
It dealt 25 damage.
Pykemon Muttle used Grow.
It healed 15 health points.

Name: Spatol
Element Type: FIRE
Health: 79 / 94
Speed: 5

Name: Muttle
Element Type: GRASS
Health: 52 / 93
Speed: 2

What would you like to do....
(1) - Scratch
(2) - Ember
(3) - Light
(4) - Fire Blast
Please enter your move choice: 3

Pykemon Spatol used Light.
It healed 23 health points.
Pykemon Muttle used Wrap.
The attack missed!!!

Name: Spatol
Element Type: FIRE

Health: 94 / 94
Speed: 5

Name: Muttle
Element Type: GRASS
Health: 52 / 93
Speed: 2

What would you like to do....
(1) - Scratch
(2) - Ember
(3) - Light
(4) - Fire Blast
Please enter your move choice: 4

Pykemon Spatol used FIRE BLAST!
It's SUPER effective!
It dealt 39 damage.
Pykemon Muttle used Wrap.
It dealt 27 damage.

Name: Spatol
Element Type: FIRE
Health: 67 / 94
Speed: 5

Name: Muttle
Element Type: GRASS
Health: 13 / 93
Speed: 2

What would you like to do....
(1) - Scratch
(2) - Ember
(3) - Light
(4) - Fire Blast
Please enter your move choice: 4

Pykemon Spatol used FIRE BLAST!
It's SUPER effective!
It dealt 49 damage.

Pykemon Muttle has fainted!
Press Enter to continue

OH NO! A wild Pyonx has approached!

Name: Pyonx
Element Type: FIRE
Health: 89 / 89
Speed: 9

What would you like to do....
(1) - Scratch
(2) - Ember
(3) - Light
(4) - Fire Blast
Please enter your move choice: 2

Pykemon Pyonx used Scratch.
It dealt 20 damage.
Pykemon Spatol used Ember.
It dealt 45 damage.

Name: Spatol
Element Type: FIRE
Health: 47 / 94
Speed: 5

Name: Pyonx
Element Type: FIRE
Health: 44 / 89
Speed: 9

What would you like to do....
(1) - Scratch
(2) - Ember
(3) - Light
(4) - Fire Blast
Please enter your move choice: 3

Pykemon Pyonx used Scratch.
It dealt 23 damage.
Pykemon Spatol used Light.

It healed 24 health points.

Name: Spatol
Element Type: FIRE
Health: 48 / 94
Speed: 5

Name: Pyonx
Element Type: FIRE
Health: 44 / 89
Speed: 9

What would you like to do....
(1) - Scratch
(2) - Ember
(3) - Light
(4) - Fire Blast
Please enter your move choice: 1

Pykemon Pyonx used Ember.
It dealt 28 damage.
Pykemon Spatol used Scratch.
It dealt 15 damage.

Name: Spatol
Element Type: FIRE
Health: 20 / 94
Speed: 5

Name: Pyonx
Element Type: FIRE
Health: 29 / 89
Speed: 9

What would you like to do....
(1) - Scratch
(2) - Ember
(3) - Light
(4) - Fire Blast
Please enter your move choice: 2

Pykemon Pyonx used FIRE BLAST!
It dealt 22 damage.
Pykemon Spatol has fainted!
Press Enter to continue

Poor Spatol has fainted...
But not before defeating 1 Pykemon!
Would you like to play again (y/n): n
Thank you for playing Pykemon!