CMSC 1236 – Advanced Programming in Python

Week 1–2 Assignment

Virtual Pet Simulator

Overview

Develop a Python program to simulate a virtual pet. This project will allow you to apply and demonstrate your understanding of Python basics, object-oriented programming, file handling using pickling, and user interaction through the console.

Pet Class

The `Pet` class is a core component of your Virtual Pet Simulator. It will model the behavior and state of your virtual pet. You will need to use conditional statements (`if` statements) in the methods of this class to simulate different scenarios based on the pet's attributes.

1. **Implementing the `Pet` Class:**
   - Follow the provided UML diagram to create the `Pet` class with the attributes `name`, `hunger`, `happiness`, `energy`, `color`, and `type`.
   - Initialize these attributes in the `__init__` method. `name`, `color`, and `type` will be strings provided when a new pet is created, and `hunger`, `happiness`, and `energy` should start with random values between 0 and 10.
2. **Using Conditional Statements in Class Methods:**
   - In methods like `feed`, `play`, and `rest`, you will use `if` statements to check the state of the pet and decide what action to take.

Method Implementations:

`feed` Method:

Objective

The `feed` method is designed to simulate feeding the pet. Feeding the pet has implications on its hunger and happiness levels.

Method Behavior and Logic

1. **Reducing Hunger and Increasing Happiness**:
   - When the `feed` method is called, first check the pet's current hunger level (`self.hunger`).
   - If `self.hunger` is greater than 0, it indicates that the pet is hungry. In this case, reduce `self.hunger` by 1 to represent feeding the pet. Ensure `self.hunger` does not go below 0.
   - After feeding, increase `self.happiness` by 1. However, ensure that `self.happiness` does not exceed 10.
2. **Managing Happiness When Not Hungry**:
   - If `self.hunger` is already at 0 (meaning the pet is not hungry), feeding the pet doesn't reduce hunger will affect its happiness. Reduce `self.happiness` by 1, since over-feeding will cause discomfort.
3. **Returning Feedback**:
   - Return a feedback string that reflects the pet's reaction to being fed.

`play` Method:

Objective

The `play` method simulates the interaction of playing with the pet. This activity affects the pet's happiness and energy levels.

Method Behavior and Logic

1. **Handling Sadness**:
   - Before initiating play, check if the pet's happiness level (`self.happiness`) is at 0.
   - If `self.happiness` is 0, it indicates that the pet is too sad to engage in play.
2. **Handling Tiredness**:
   - Check the pet's energy level (`self.energy`).
   - If `self.energy` is 0, it means the pet is too tired to play.
3. **Playing with the Pet**:
   - If the pet is neither too sad nor too tired (i.e., `self.happiness` is above 0 and `self.energy` is above 0), proceed to play with the pet.
   - Increase `self.happiness` by 1 to represent the positive effect of playing. However, ensure that `self.happiness` does not exceed 10.
   - Decrease `self.energy` by 1 to account for the energy expended during play. However, it's important to ensure that `self.energy` does not fall below 0.
4. **Returning Feedback**:
   - Return a feedback string that reflects the pet's reaction to playing.

`rest` Method:

Objective

The `rest` method is designed to simulate the pet resting, which affects its energy and happiness levels. Resting is essential for the pet to regain energy, but it might also affect its happiness if it becomes bored or inactive.

Method Behavior and Logic

1. **Increasing Energy**:
   - Check the pet's current energy level (`self.energy`).
   - If `self.energy` is less than 10, it indicates the pet could benefit from rest. Increase `self.energy` by 1 to represent the energy regained from resting. However, ensure that `self.energy` does not exceed 10.
2. **Managing Happiness When Energy is at Maximum**:
   - If `self.energy` is already at 10 (meaning the pet is fully rested), resting will affect its happiness.
   - If `self.energy` is already at 10 and `self.happiness` is greater than 0, reduce `self.happiness` it by 1 to represent a slight decrease in happiness due to inactivity or boredom. Be sure to prevent `self.happiness` from going below 0.
3. **Returning Feedback**:
   - Return a feedback string that reflects the pet's reaction to resting.

`customize` Method:

Objective

The `customize` method allows the user to change the pet's appearance by setting new values for the `color` and `type` attributes.

Method Behavior and Logic

1. **Setting New Attributes**:
   - The method takes two parameters, `color` and `pet_type`, and assigns them to the pet's `color` and `type` attributes.
   - This allows the user to update the pet's appearance during the simulation. Display the changed output to the user.

`status` Method:

Objective

The `status` method provides a summary of the current state of the animal, including its hunger, happiness, energy, color, and type. This method is crucial for understanding the animal's overall well-being at any given moment.

Method Behavior and Logic

1. **Creating a Status Summary**:
   - The method compiles a string that summarizes the animal's current status.
   - This summary includes the animal's name and its current hunger, happiness, energy, color, and type.
2. **String Formatting**:
   - Create a readable and well-formatted summary.
   - Each attribute (hunger, happiness, energy, color, type) should be on a separate line for clarity.
3. **Returning the Summary**:
   - The method returns the formatted string.

`is_happy` Method:

Objective

The `is_happy` method is designed to evaluate the animal's happiness level and determine if the animal is currently happy. This method provides a boolean result based on the happiness attribute.

Method Behavior and Logic

1. **Evaluating Happiness**:
   - The method assesses whether the animal's happiness level (`self.happiness`) is below or above 5.
   - If `self.happiness` is above 5 the animal is considered to be happy.
2. **Returning a Boolean Result**:
   - The method returns `True` if the animal's happiness is above 5, indicating that the animal is happy.
   - It returns `False` if the happiness level is 5 or lower, indicating that the animal is not happy.

Main Function

Overview

Your `main()` function serves as the interaction hub between the user and their virtual pet. It will manage user inputs through a menu system and trigger appropriate actions based on the user's choices.

1. **Menu System:**
   - Implement a menu system that displays a list of possible actions the user can perform. These actions include:
     - Feeding the pet
     - Playing with the pet
     - Letting the pet rest
     - Checking the pet's status
     - Customizing the pet's appearance
     - Saving the pet's state and exiting the program
   - Use an input prompt to allow the user to select an action and then call the corresponding method from the `Pet` class.

2. **Handling Invalid Input:**
   - Ensure that the program gracefully handles invalid menu choices. (This needs to be demonstrated in submission video).
   - Display an error message if the user inputs an invalid option and show the menu again for a valid selection.

Data Persistence

- Use pickling to save and load the pet's state. Implement a `save_pet` and a `load_pet` function in the `main()` for this purpose.
- Ensure the program can handle situations where no saved data exists.

Sample Output

```
Welcome to the Virtual Pet Simulator!

In this program, you will create and take care of a virtual pet.
You can feed, play with, and let your pet rest to maintain its health
 ↪  and happiness.
You can also customize your pet's appearance and save its state to
 ↪  continue later.
Use the menu to interact with your pet and make sure it stays happy
 ↪  and healthy!
```

```
Let's get started!

What is your pet's name? Joe
Choose a color for your pet (e.g., red, blue, green): Blue
Choose a type for your pet (e.g., dog, cat, bird): Dog

Here is the current status of your pet:

Joe's Status:
Color: Blue
Type: Dog
Hunger: 8
Happiness: 1
Energy: 10

What would you like to do with your pet?
1: Feed
2: Play
3: Rest
4: Check Status
5: Customize Pet
6: Save and Exit
Enter your choice: 9
Invalid choice. Please enter a number between 1 and 6.

What would you like to do with your pet?
1: Feed
2: Play
3: Rest
4: Check Status
5: Customize Pet
6: Save and Exit
Enter your choice: 4

Joe's Status:
Color: Blue
Type: Dog
Hunger: 8
Happiness: 1
Energy: 10

What would you like to do with your pet?
1: Feed
2: Play
3: Rest
4: Check Status
5: Customize Pet
6: Save and Exit
```

```
Enter your choice: 3
Joe is rested and seems a bit bored.

What would you like to do with your pet?
1: Feed
2: Play
3: Rest
4: Check Status
5: Customize Pet
6: Save and Exit
Enter your choice: 1
You fed Joe. Joe seems happier after eating!

What would you like to do with your pet?
1: Feed
2: Play
3: Rest
4: Check Status
5: Customize Pet
6: Save and Exit
Enter your choice: 4

Joe's Status:
Color: Blue
Type: Dog
Hunger: 7
Happiness: 2
Energy: 10

What would you like to do with your pet?
1: Feed
2: Play
3: Rest
4: Check Status
5: Customize Pet
6: Save and Exit
Enter your choice: 5
Choose a color for your pet: Green
Choose a type for your pet: Dog
Joe is now a Green Dog!

What would you like to do with your pet?
1: Feed
2: Play
3: Rest
4: Check Status
5: Customize Pet
6: Save and Exit
Enter your choice: 6
```

```
Pet state saved. Goodbye!
```

Starting the Program with a File Created

```
Welcome to the Virtual Pet Simulator!

In this program, you will create and take care of a virtual pet.
You can feed, play with, and let your pet rest to maintain its health
 ↪  and happiness.
You can also customize your pet's appearance and save its state to
 ↪  continue later.
Use the menu to interact with your pet and make sure it stays happy
 ↪  and healthy!

Let's get started!


Here is the current status of your pet:

Joe's Status:
Color: Green
Type: Dog
Hunger: 7
Happiness: 2
Energy: 10

What would you like to do with your pet?
1: Feed
2: Play
3: Rest
4: Check Status
5: Customize Pet
6: Save and Exit
Enter your choice: 2
Playing with Joe was fun!

What would you like to do with your pet?
1: Feed
2: Play
3: Rest
4: Check Status
5: Customize Pet
6: Save and Exit
Enter your choice: 4
Joe's Status:
Color: Green
Type: Dog
```

```
Hunger: 7
Happiness: 3
Energy: 9

What would you like to do with your pet?
1: Feed
2: Play
3: Rest
4: Check Status
5: Customize Pet
6: Save and Exit
Enter your choice: 6
Pet state saved. Goodbye!
```

UML

```
+----------------------+
|        Pet           |
+----------------------+
| - name: str          |
| - hunger: int        |
| - happiness: int     |
| - energy: int        |
| - color: str         |
| - type: str          |
+----------------------+
| + __init__(name, color, pet_type) |
| + feed()             |
| + play()             |
| + rest()             |
| + customize(color, pet_type) |
| + status(): str      |
| + is_happy(): bool   |
+----------------------+
```