Python Fundamentals (Assignment4)

Assignment Problems

> **Concept:** Classes & Objects

Q1. Create a `BankAccount` class with **attributes** account_number, owner_name, and balance.
Add **methods** to deposit, withdraw, and check balance.

> **Concept:** Classes & Objects

Q2. Create a class `Book` with the following attributes:

- title
- author
- list of reviews

And add methods to:

- add a new review
- count reviews
- display all reviews

> **Concept:** Encapsulation

Q3. Create a class `Student` with **private** attributes _name, _roll_no, and _marks. Provide **getter** and **setter** methods with validation (e.g., marks cannot be negative, roll number has to be between 1 & 100 & name cannot be empty).

> **Concept:** Function Overriding

Q4. Create a class `Shape` with a method area().
Create subclasses `Circle`, `Rectangle`, and `Triangle` that **override** the area() method.

Concept: Inheritance

Q5. Create a **base** class `Vehicle` with attributes like brand and model.
Create two **subclasses** `Car` and `Bike` that add extra attributes - seats (in Car) & engine_cc (in Bike).

Concept: Abstraction

Q6. Create an **abstract** class `Employee` with an **abstract method** calculate_salary().
Create subclasses `Intern`, `FullTimeEmployee`, and `ContractEmployee` that implement the method differently.

Concept: Constructor Overloading (with Default Parameters)

Q7. Create a class `Person` that allows the constructor to work with:

- name only
- name + age
- name + age + address

As direct constructor overloading (multiple constructors) are not allowed but we have to use **default parameters** to simulate constructor overloading.

Concept: Instance & Class Attributes

Q8. Create a class `Player` with:

- a class variable player_count
- instance variables name and level

Track how many players were created.

Concept: Multiple Inheritance

Q9. Create the following classes: `Herbivore`, `Carnivore`, `Omnivore` with some attributes & methods. Then create a class `Bear` that inherits from all the above classes to showcase how multiple inheritance works.

> **Concept:** *OOP*

Q10. Mini Project - OOP Chat System

Let's create a Chat System using OOPs concepts. We have to create classes:

- `User`
- `Message`
- `ChatRoom`

And we have to implement functions:

- sending messages
- viewing chat history
- user joining and leaving the chatroom