Assignment-1
OOP Coding Questions

Class Definition and Object Instantiation

1. **Create a class to represent a Student with attributes like name, age, and grades.**
   Example Test Case:
   - **Input:** `name = "Alice", age = 20, grades = [90, 85, 88]`
   - **Output:** `name: "Alice", age: 20, grades: [90, 85, 88]`
2. **Given a CSV file with employee details (name, position, salary), create a class to represent an Employee.**
   Example Test Case:
   - **Input:** `CSV file content = [["John Doe", "Manager", 75000], ["Jane Smith", "Engineer", 80000]]`
   - **Output:** `Employee(name="John Doe", position="Manager", salary=75000), Employee(name="Jane Smith", position="Engineer", salary=80000)`
3. **Implement a program that simulates a basic bank account using a BankAccount class.**
   Example Test Case:
   - **Input:** `account_number = "12345678", initial_balance = 1000`
   - **Output:** `account_number: "12345678", balance: 1000`
4. **Write a Python program that uses a Rectangle class to calculate the area and perimeter of a rectangle.**
   Example Test Case:
   - **Input:** `length = 5, width = 3`
   - **Output:** `area: 15, perimeter: 16`
5. **Create a class to represent a Car with attributes like make, model, and year.**
   Example Test Case:
   - **Input:** `make = "Toyota", model = "Camry", year = 2020`
   - **Output:** `make: "Toyota", model: "Camry", year: 2020`
6. **Given a JSON file with customer data, create a Customer class to store and manipulate the data.**
   Example Test Case:
   - **Input:** `JSON file content = {"name": "John Doe", "email": "john.doe@example.com"}`
   - **Output:** `Customer(name="John Doe", email="john.doe@example.com")`
7. **Write a program that uses a Person class to keep track of a person's name, age, and address.**
   Example Test Case:

- **Input:** `name = "John Doe", age = 30, address = "123 Main St"`
- **Output:** `name: "John Doe", age: 30, address: "123 Main St"`

8. **Implement a program that uses a Circle class to calculate the area and circumference of multiple circles.**
   Example Test Case:
   - **Input:** `radius = 4`
   - **Output:** `area: 50.27, circumference: 25.13`
9. **Given a CSV file with product details (name, price, quantity), create a Product class to manage the data.**
   Example Test Case:
   - **Input:** `CSV file content = [["Laptop", 1000, 5], ["Phone", 500, 10]]`
   - **Output:** `Product(name="Laptop", price=1000, quantity=5), Product(name="Phone", price=500, quantity=10)`
10. **Create a class to represent a Movie with attributes like title, director, and rating.**
    Example Test Case:
    - **Input:** `title = "Inception", director = "Christopher Nolan", rating = 8.8`
    - **Output:** `title: "Inception", director: "Christopher Nolan", rating: 8.8`

Class Hierarchies and Inheritance

1. **Create a base class Shape with methods to calculate area and perimeter, and derive classes Circle and Square.**
   Example Test Case:
   - **Input:** `Circle(radius=4), Square(side=5)`
   - **Output:** `Circle area: 50.27, Circle perimeter: 25.13, Square area: 25, Square perimeter: 20`
2. **Implement a class hierarchy to represent different types of employees (Manager, Engineer) with their attributes.**
   Example Test Case:
   - **Input:** `Manager(name="John Doe", department="Sales"), Engineer(name="Jane Smith", field="Software")`
   - **Output:** `Manager: John Doe, Department: Sales, Engineer: Jane Smith, Field: Software`
3. **Write a Python program that uses inheritance to represent a hierarchy of shapes (Triangle, Rectangle, etc.).**
   Example Test Case:

- **Input:** `Triangle(base=5, height=3), Rectangle(length=4, width=2)`
- **Output:** `Triangle area: 7.5, Rectangle area: 8`

4. **Create a class hierarchy to represent different types of animals (Bird, Fish) with their own attributes and methods.**
   Example Test Case:
   - **Input:** `Bird(name="Parrot", can_fly=True), Fish(name="Goldfish", can_swim=True)`
   - **Output:** `Bird: Parrot, Can Fly: True, Fish: Goldfish, Can Swim: True`
5. **Given a JSON file with product details (name, price, quantity), create a Product class with encapsulated attributes.**
   Example Test Case:
   - **Input:** `JSON file content = {"name": "Laptop", "price": 1000, "quantity": 5}`
   - **Output:** `Product(name="Laptop", price=1000, quantity=5)`
6. **Implement a program that uses inheritance to represent a hierarchy of vehicles (Car, Bike, Truck, etc.).**
   Example Test Case:
   - **Input:** `Car(make="Toyota", model="Camry"), Bike(make="Yamaha", model="MT-07")`
   - **Output:** `Car: Toyota Camry, Bike: Yamaha MT-07`
7. **Write a Python program that uses encapsulation to protect sensitive information in a User class.**
   Example Test Case:
   - **Input:** `User(username="john_doe", password="secure123")`
   - **Output:** `username: "john_doe", password: "******"`

Assignment-2

Class Definition and Object Instantiation

1. **Create a class `Library` with attributes like `name`, `address`, and a list of books. Implement methods to add and remove books.**
    - **Example Test Case:**
        - **Input:** `name = "City Library", address = "123 Main St", books = []`
        - **Method Calls:** `add_book("Book1"), add_book("Book2"), remove_book("Book1")`
        - **Output:** `name: "City Library", address: "123 Main St", books: ["Book2"]`
2. **Create a class `House` with attributes like `address`, `num_rooms`, and `price`. Implement a method to display the house details.**
    - **Example Test Case:**
        - **Input:** `address = "456 Elm St", num_rooms = 4, price = 350000`
        - **Method Call:** `display_details()`
        - **Output:** `address: "456 Elm St", num_rooms: 4, price: 350000`

Instance Methods

3. **Create a class `Book` with attributes `title`, `author`, and `price`. Implement a method `discount` to apply a discount to the book's price.**
    - **Example Test Case:**
        - **Input:** `title = "Python 101", author = "John Doe", price = 29.99`
        - **Method Call:** `discount(0.1)`
        - **Output:** `title: "Python 101", author: "John Doe", price: 26.99`
4. **Create a class `Restaurant` with attributes `name`, `cuisine_type`, and `rating`. Implement a method `update_rating` to change the restaurant's rating.**
    - **Example Test Case:**
        - **Input:** `name = "Sushi Place", cuisine_type = "Japanese", rating = 4.5`
        - **Method Call:** `update_rating(4.8)`

        - **Output:** `name: "Sushi Place", cuisine_type: "Japanese", rating: 4.8`

Class Variables

5. **Create a class `School` with a class variable `total_students` and instance variables `name` and `students`. Implement a method to enroll students and update the total count.**
    - **Example Test Case:**
        - **Input:** `name = "Greenwood High", students = 300`
        - **Method Call:** `enroll_students(50)`
        - **Output:** `name: "Greenwood High", students: 350, total_students: 350`
6. **Create a class `Company` with a class variable `industry` and instance variables `name` and `num_employees`. Implement a method to update the number of employees.**
    - **Example Test Case:**
        - **Input:** `name = "TechCorp", num_employees = 200`
        - **Method Call:** `update_employees(220)`
        - **Output:** `name: "TechCorp", num_employees: 220, industry: "Technology"`

Static Methods

7. **Create a class `MathUtils` with a static method `is_prime` to check if a number is prime.**
    - **Example Test Case:**
        - **Input:** `number = 17`
        - **Method Call:** `is_prime(17)`
        - **Output:** `True`
8. **Create a class `TemperatureConverter` with a static method `celsius_to_fahrenheit` to convert Celsius to Fahrenheit.**
    - **Example Test Case:**
        - **Input:** `celsius = 25`
        - **Method Call:** `celsius_to_fahrenheit(25)`
        - **Output:** `77.0`

Inheritance

9. **Create a base class `Employee` with attributes `name` and `salary`. Create a subclass `Developer` that adds an attribute `programming_language`.**
    - **Example Test Case:**
        - **Input:** `name = "Alice", salary = 70000, programming_language = "Python"`
        - **Output:** `name: "Alice", salary: 70000, programming_language: "Python"`
10. **Create a base class `Appliance` with a method `turn_on`. Create a subclass `WashingMachine` that overrides the `turn_on` method.**
    - **Example Test Case:**
        - **Input:** `Appliance.turn_on(), WashingMachine(model="LG").turn_on()`
        - **Output:** `"Appliance is turned on", "Washing Machine LG is turned on"`

Encapsulation

11. **Create a class `PrivateData` with private attributes `username` and `password`. Implement methods to get and set these attributes.**
    - **Example Test Case:**
        - **Input:** `username = "user1", password = "pass123"`
        - **Method Calls:** `get_username(), set_password("newpass123"), get_password()`
        - **Output:** `username: "user1", password: "newpass123"`
12. **Create a class `Account` with private attributes `account_number` and `balance`. Implement methods to deposit and withdraw money.**
    - **Example Test Case:**
        - **Input:** `account_number = "987654321", balance = 5000`
        - **Method Calls:** `deposit(1500), withdraw(2000), get_balance()`
        - **Output:** `account_number: "987654321", balance: 4500`

Polymorphism

13. **Create a function `operate_vehicle` that takes a vehicle object and calls its `move` method. Create classes `Car` and `Bike` that implement `move` differently.**

- **Example Test Case:**
  - **Input:** `Car(model="Toyota").move(), Bike(model="Yamaha").move()`
  - **Output:** `"Toyota is driving", "Yamaha is riding"`

14. **Create a function `operate_device` that takes a device object and calls its `operate` method. Create classes `Laptop` and `Smartphone` that implement `operate` differently.**
    - **Example Test Case:**
      - **Input:** `Laptop(model="Dell").operate(), Smartphone(model="iPhone").operate()`
      - **Output:** `"Dell is operating", "iPhone is operating"`

Abstraction

15. **Define an abstract class `Transport` with an abstract method `travel`. Create subclasses `Bus` and `Train` that implement the `travel` method.**
    - **Example Test Case:**
      - **Input:** `Bus(route="10A").travel(), Train(route="Express").travel()`
      - **Output:** `"Bus 10A is traveling", "Train Express is traveling"`
16. **Define an abstract class `Payment` with an abstract method `process`. Create subclasses `CreditCard` and `PayPal` that implement the `process` method.**
    - **Example Test Case:**
      - **Input:** `CreditCard(number="1234").process(), PayPal(account="user@example.com").process()`
      - **Output:** `"Processing credit card payment for 1234", "Processing PayPal payment for user@example.com"`