Foundation Training '25

CC Programming-Foundations

Instructions

- Write Pseudo code for the below problems
- Test the Pseudo code with the tests provided in the problem statements to validate the logic
- Start coding only after the first two steps are completed.

Overview

Complete all problems.

Task 1 : Bank Account System (OOP with Inheritance)

Problem Statement

Create a base class `BankAccount` with:

- Attributes: `account_number`, `account_holder`, `balance`
- Methods: `deposit()`, `withdraw()`, `display_balance()`

Create a derived class `SavingsAccount` with:

- An additional attribute: `interest_rate`
- Method: `apply_interest()`

Expected output

Example:

```
Input: deposit 500, withdraw 100, interest_rate 4%
Output:
Deposited: 500
Withdrawn: 100
Interest Applied: 16.0
Current Balance: 416.0
```

Task 2 : Inventory Management (Encapsulation & Polymorphism)

Problem Statement

Create a class `Product` with private variables for:

- `product_id`, `name`, `quantity`, and `price`

Implement:

- Getter and setter methods to update inventory securely
- A method `get_value()` that returns total stock value (qty × price)

Create a list of products and:

- Display total inventory value
- Demonstrate polymorphism if extended into subclasses (e.g., `PerishableProduct`)

Task 3 : Modularize a Codebase

Problem Statement

Refactor your Inventory Management program (from Task 2) by separating the code into multiple modules for better structure and maintainability.

Your goal is to create a modular project with:

- `main.py`
    - Entry point of the program
    - Handles user interaction and displays output
- `product.py`
    - Contains the `Product` class and any subclasses (e.g., `PerishableProduct`)
    - Includes encapsulation and polymorphism logic
- `inventory_utils.py`
    - Contains helper functions (e.g., calculate total inventory value, filter low stock items, etc.)

Use `import` and maintain proper structure.

Task 4 : Regex Email and Phone Validator

Problem Statement

Write a program using regex to validate:

- Email addresses (should contain `@`, `.`, valid domain)
- Indian phone numbers (start with 7/8/9, 10 digits)

Use:

- `re.match()`, `re.search()`, and `re.findall()`
- User input with validation and appropriate messages

Expected output

Example:

```
Input: user@example.com
Output: Valid email

Input: 9876543210
Output: Valid phone number
```

Task 5 : Find All Emails in a Paragraph

Problem Statement

Take a block of text and extract all valid email addresses using regex.

Expected output

Example:

```
Input: "Send details to test@example.com and info@company.org"
Output: ['test@example.com', 'info@company.org']
```
