**1. Banking System**

Design a program to simulate a simple banking system.

1. Create a BankAccount class with attributes like account number, account holder name, and balance.
2. Add methods to deposit money, withdraw money (ensuring sufficient balance), and check the balance.
3. Create subclasses like SavingsAccount and CurrentAccount, each with additional rules (e.g., minimum balance for savings, overdraft limit for current).
4. Allow multiple users to interact with the system.

**Concepts:** Classes, inheritance, conditionals, and arithmetic operations.

**2. Car Rental System**

Create a program to manage a car rental service.

1. Create a Car class with attributes like make, model, registration number, and availability status.
2. Create a RentalSystem class to manage the cars, allowing users to rent or return a car.
3. Track the rental status and calculate the rental cost based on the number of days.
4. Implement a penalty for late returns.

**Concepts:** Encapsulation, file handling (to save rental data), and conditionals.

**3. Inventory Management System**

Simulate an inventory system for a store.

1. Create a Product class with attributes like product ID, name, price, and stock quantity.
2. Allow operations like adding new products, updating stock, and viewing inventory.
3. Create a Sale class to handle selling products, updating stock, and generating bills.
4. Add a feature to calculate total sales and display low-stock products.

**Concepts:** OOP, lists or dictionaries, loops, and arithmetic operations.

**4. Online Exam System**

Develop a program to simulate an online exam platform.

1. Create a Question class with attributes like the question text, options, and the correct answer.
2. Create a Quiz class that manages a set of questions.
3. Allow users to take the quiz, record their answers, and calculate their final score.
4. Implement a timer feature to limit the quiz duration.

**Concepts:** Classes, lists, file handling (to load questions from a file), and loops.

**5. Restaurant Billing System**

Simulate a restaurant’s billing system.

1. Create a MenuItem class to represent dishes with attributes like name, price, and category.
2. Create a Restaurant class to manage the menu and handle customer orders.
3. Generate an itemized bill with the total amount, including taxes and service charges.
4. Allow customers to modify or cancel orders before generating the final bill.

**Concepts:** OOP, dictionaries or lists, loops, and arithmetic operations.

**6. Train Ticket Booking System**

Build a train ticket booking system.

1. Create a Train class with attributes like train name, train number, source, destination, and available seats.
2. Allow users to view available trains, book tickets, and cancel tickets.
3. Track passenger details and the number of tickets booked for each train.
4. Ensure users cannot book more seats than available.

**Concepts:** OOP, lists or dictionaries, conditionals, and loops.

**7. Employee Management System**

Design a system to manage employee details.

1. Create an Employee class with attributes like employee ID, name, department, and salary.
2. Create methods to add, remove, and update employee details.
3. Create subclasses for specific employee types (e.g., Manager, Engineer) with additional attributes like bonus or project allocation.
4. Generate reports for payroll and department-wise employee details.

**Concepts:** Classes, inheritance, file handling, and dictionaries.

**8. Hospital Management System**

Simulate a hospital management system.

1. Create classes for Patient and Doctor with relevant details.
2. Allow patients to book appointments with available doctors.
3. Manage appointment schedules, ensuring no double booking.
4. Add a feature to generate bills for patient treatment, including consultation fees and medicine charges.

**Concepts:** Encapsulation, inheritance, file handling, and conditionals.

**9. E-commerce Order Processing System**

Build a program to handle an e-commerce platform’s order system.

1. Create a Product class with attributes like product ID, name, price, and stock.
2. Allow users to browse products, add items to their cart, and place orders.
3. Deduct the ordered quantity from the stock and calculate the total price.
4. Add discount rules for bulk purchases or special items.

**Concepts:** Classes, lists or dictionaries, and arithmetic operations.

**10. Parking Lot Management System**

Design a parking lot simulation.

1. Create a ParkingSlot class to represent each parking space with attributes like slot number and availability.
2. Create a ParkingLot class to manage a collection of parking slots.
3. Allow users to park or remove their vehicles and track the parking fee based on duration.
4. Add a feature to display available slots and generate daily revenue reports.

**Concepts:** OOP, lists, conditionals, and time handling (datetime module).

**11. Weather Data Analysis System**

Build a program to analyze weather data.

1. Create a WeatherData class to store daily data like temperature, humidity, and rainfall.
2. Allow users to input data for multiple days and perform operations like:
   * Calculate average temperature, humidity, and rainfall.
   * Identify the day with the highest rainfall or lowest temperature.
3. Generate a summary report for a given time range.

**Concepts:** OOP, file handling, loops, and conditionals.

**12. Movie Ticket Booking System**

Create a system to book movie tickets.

1. Create a Movie class to store movie details like name, showtimes, and available seats.
2. Allow users to view available movies and book tickets for specific showtimes.
3. Ensure that ticket booking reduces the seat count and prevent overbooking.
4. Add a feature to cancel bookings and update the seat count.

**Concepts:** Classes, dictionaries, loops, and conditionals.

**13. Fitness Tracker**

Develop a program to track a user's fitness activities.

1. Create a User class with attributes like name, age, weight, and height.
2. Allow users to log activities like running, cycling, or swimming with duration and calories burned.
3. Generate a weekly summary of total activities and calories burned.
4. Add a BMI calculator method for the user.

**Concepts:** OOP, arithmetic operations, and file handling.

**14. Personal Budget Manager**

Create a program to help users manage their personal budget.

1. Create a Budget class with attributes like income, expenses, and savings.
2. Allow users to add income sources and expense categories.
3. Generate a summary of the monthly balance and savings.
4. Add visualizations using the matplotlib library (optional).

**Concepts:** Classes, loops, and arithmetic operations.

**15. Airline Reservation System**

1. Create a Flight class with attributes like flight number, source, destination, and available seats.
2. Allow users to book tickets, cancel tickets, and view available flights.
3. Ensure that no overbooking occurs and maintain a record of passenger details for each flight.
4. Add pricing rules based on class (e.g., Economy, Business).

**Concepts:** Encapsulation, lists or dictionaries, inheritance (e.g., flight classes), and loops.

**16. Auction System**

1. Create a Product class to represent items up for auction with attributes like item name, description, and base price.
2. Create a Bid class to represent bids placed by users with attributes like bidder name and bid amount.
3. Allow users to place bids on items and ensure the highest bid wins.
4. Add a feature to display the auction winner after the bidding closes.

**Concepts:** OOP, lists, conditionals, and sorting algorithms.

**17. Virtual Classroom System**

1. Create a Classroom class with attributes like course name, instructor, and a list of students.
2. Allow the instructor to add or remove students, assign grades, and generate progress reports.
3. Include a Student class to represent individual students with attributes like name, roll number, and grades.
4. Add features for attendance tracking and displaying the class topper.

**Concepts:** Classes, inheritance, file handling (to save records), and conditionals.

**18. Voting System**

1. Create a Candidate class to represent candidates in an election.
2. Create a VotingSystem class to manage the voting process, including:
   * Adding candidates.
   * Casting votes.
   * Counting votes.
3. Display the election winner and handle ties gracefully.
4. Prevent duplicate votes by tracking voter IDs.

**Concepts:** OOP, lists or dictionaries, and loops.

**19. Railway Timetable Management System**

1. Create a Train class to represent trains with attributes like train number, source, destination, and departure time.
2. Allow users to view train schedules and search for trains by source or destination.
3. Add methods to modify schedules or cancel trains.
4. Ensure proper formatting for time display and sorting schedules by time.

**Concepts:** Encapsulation, file handling, sorting, and string manipulation.

**20. Real Estate Management System**

1. Create a Property class with attributes like property ID, location, type (e.g., apartment, house), and price.
2. Allow users to add new properties, search properties by location, and mark properties as sold.
3. Create a Customer class to handle buyers, including storing buyer details.
4. Add features for generating sales reports and calculating the total revenue.

**Concepts:** Classes, inheritance, dictionaries, and file handling.

**21. School Fee Management System**

1. Create a Student class with attributes like name, roll number, and class.
2. Allow users to pay fees for students and maintain records of paid and pending fees.
3. Generate reports showing which students have pending fees.
4. Implement discounts for students based on specific criteria (e.g., scholarships).

**Concepts:** Classes, conditionals, file handling, and arithmetic operations.

**22. Chat Application Simulation**

1. Create a User class to represent a user with attributes like username and password.
2. Create a Chat class to simulate chat messages between users.
3. Allow users to log in, send messages, and view chat history.
4. Add a feature to block users or delete conversations.

**Concepts:** OOP, lists or dictionaries, and string manipulation.

**23. Library Fine Calculator**

1. Extend the library management system by adding a feature to calculate overdue fines for borrowed books.
2. Use a datetime module to track borrowing and return dates.
3. Ensure fines are calculated correctly based on the number of overdue days.
4. Generate a detailed fine report for all users.

**Concepts:** Classes, date and time handling, arithmetic operations, and loops.

**24. Vehicle Management System**

1. Create a Vehicle class to represent cars, bikes, or trucks with attributes like registration number, type, and owner details.
2. Allow users to add vehicles, update owner details, and search for vehicles by registration number.
3. Add a feature to generate a report of all registered vehicles.
4. Ensure data is saved to and retrieved from a file.

**Concepts:** Encapsulation, file handling, and conditionals.

**25. Expense Tracker**

1. Create a Category class to represent expense categories like food, travel, and shopping.
2. Allow users to add expenses, assign them to categories, and set monthly budgets for each category.
3. Track the total expenses for each category and warn the user if they exceed the budget.
4. Generate monthly reports showing total expenses and remaining budget.

**Concepts:** OOP, dictionaries, loops, and arithmetic operations.

**26. Taxi Booking System**

1. Create a Taxi class with attributes like taxi ID, driver name, location, and availability status.
2. Allow users to book taxis by providing their current location and destination.
3. Calculate the fare based on distance and track the taxi’s availability status.
4. Add a feature to display all taxis and their current status (available or booked).

**Concepts:** Encapsulation, arithmetic operations, and string manipulation.

**27. Recipe Management System**

1. Create a Recipe class to represent recipes with attributes like name, ingredients, and preparation steps.
2. Allow users to add, view, and search recipes.
3. Add a feature to filter recipes based on available ingredients.
4. Save recipes to a file and allow users to load them later.

**Concepts:** Classes, lists, file handling, and string manipulation.

**28. Task Manager**

1. Create a Task class to represent tasks with attributes like task name, deadline, and priority.
2. Allow users to add, update, and delete tasks.
3. Display tasks sorted by priority or deadline.
4. Add a feature to mark tasks as complete and view completed tasks separately.

**Concepts:** Classes, sorting, date and time handling, and file handling.

**29. Weather Forecast Application**

1. Create a City class with attributes like city name, current temperature, and weather conditions.
2. Allow users to search for weather details by city.
3. Add a feature to predict weather for the next day based on basic statistical rules.
4. Save weather data to a file for historical reference.

**Concepts:** OOP, file handling, and basic statistics.

**30. Smart Calculator**

1. Create a Calculator class with methods for basic arithmetic operations.
2. Extend the class to support advanced operations like square roots, exponents, and logarithms.
3. Allow users to input expressions (e.g., "2 + 3 \* 5") and evaluate them.
4. Add a history feature to store and retrieve past calculations.

**Concepts:** Classes, string manipulation, arithmetic operations, and lists.

**31. Hospital Management System**

1. Create a Patient class with attributes like patient name, age, ailment, and doctor assigned.
2. Create a Doctor class to manage doctors with attributes like specialization and availability.
3. Allow patients to book appointments, view their doctor’s schedule, and update appointment details.
4. Generate patient reports, including past consultations and prescriptions.

**Concepts:** Classes, dictionaries, file handling, and conditional logic.

**32. Online Quiz Application**

1. Create a Question class with attributes like the question, options, and correct answer.
2. Allow the admin to add and modify quiz questions.
3. Create a User class that tracks user names and quiz scores.
4. Allow users to take quizzes, and display scores and correct answers at the end.

**Concepts:** OOP, file handling, loops, and conditional statements.

**33. Inventory Management System**

1. Create a Product class to represent items with attributes like product ID, name, quantity, and price.
2. Allow users to add new products, update stock, and remove discontinued products.
3. Generate sales invoices and keep track of revenue.
4. Ensure data persistence using file handling or JSON.

**Concepts:** Classes, file handling, dictionaries, and loops.

**34. Banking System with Loan Feature**

1. Extend a simple banking system to include a Loan class.
2. Allow users to apply for loans and calculate monthly installments based on interest rates.
3. Include methods to check loan eligibility (e.g., based on income or credit score).
4. Keep track of loan repayment history.

**Concepts:** OOP, arithmetic operations, and file handling.

**35. Task Reminder Application**

1. Create a Task class to represent tasks with attributes like title, description, and due date.
2. Allow users to add, update, and delete tasks.
3. Implement a feature to notify users of overdue tasks using the datetime module.
4. Save tasks to a file and load them upon restarting the application.

**Concepts:** OOP, date and time handling, and file I/O.

**36. E-Commerce Platform**

1. Create a User class to represent customers with attributes like name and purchase history.
2. Create a Product class for items with attributes like name, price, and stock.
3. Allow users to browse products, add items to a cart, and place orders.
4. Add a discount system for bulk purchases or special offers.

**Concepts:** Classes, dictionaries, file handling, and conditional logic.

**37. Car Rental System**

1. Create a Car class with attributes like model, registration number, rental price, and availability.
2. Allow users to book cars for a specific number of days and calculate rental costs.
3. Update availability after each booking and handle cancellations.
4. Display a report of all rented cars and their current status.

**Concepts:** OOP, arithmetic operations, and file I/O.

**38. Online Exam Portal**

1. Create a Subject class with attributes like subject name and topic list.
2. Allow the admin to create and update question banks for each subject.
3. Allow students to take exams by selecting a subject, and randomize questions during the exam.
4. Generate a detailed score report with feedback on weak topics.

**Concepts:** Classes, lists, randomization, and file handling.

**39. Movie Ticket Booking System**

1. Create a Movie class with attributes like title, genre, showtimes, and available seats.
2. Allow users to view showtimes, book tickets, and select seats.
3. Include dynamic seat allocation and ensure no double bookings.
4. Add a feature to display earnings and occupancy rates for each show.

**Concepts:** Classes, dictionaries, loops, and conditional logic.

**40. Cryptocurrency Portfolio Manager**

1. Create a Coin class to represent cryptocurrencies with attributes like name, symbol, and current price.
2. Allow users to add coins to their portfolio, update holdings, and calculate the portfolio’s total value.
3. Use an API to fetch live cryptocurrency prices (optional).
4. Display a graph showing portfolio performance over time.

**Concepts:** OOP, APIs, and plotting libraries like matplotlib.

**41. Fitness Tracker**

1. Create a User class to track user details like name, age, and daily activity goals.
2. Allow users to log activities like walking, running, or cycling with durations and calories burned.
3. Generate weekly and monthly activity reports.
4. Provide recommendations for achieving fitness goals.

**Concepts:** Classes, arithmetic operations, and date handling.

**42. Personal Finance Manager**

1. Create a Transaction class with attributes like date, amount, category, and description.
2. Allow users to log income and expenses and view their financial summary.
3. Add a feature to generate monthly reports and categorize spending.
4. Provide graphs for spending trends using matplotlib.

**Concepts:** OOP, file handling, and visualization.

**43. Music Library Manager**

1. Create a Song class with attributes like title, artist, album, and duration.
2. Allow users to add, search, and delete songs from their library.
3. Add a feature to create playlists and shuffle songs within a playlist.
4. Save the library and playlists to a file for persistence.

**Concepts:** Classes, file handling, and randomization.

**44. Multiplayer Tic-Tac-Toe**

1. Create a Board class to manage the tic-tac-toe grid.
2. Create a Player class for players with attributes like name and symbol (X or O).
3. Implement game logic to handle turns, check for winners, and manage the game state.
4. Add a replay feature to play multiple rounds.

**Concepts:** Classes, nested loops, and conditional logic.

**45. Recipe Recommendation System**

1. Create a Recipe class to store details like name, ingredients, and preparation time.
2. Allow users to search recipes based on available ingredients or preparation time.
3. Add a feature to recommend recipes based on user preferences.
4. Save and load recipes from a file for long-term storage.

**Concepts:** OOP, file handling, and string manipulation.