Unit - I
Experiment 1

1. Student Record System
Scenario: You are developing a college system that maintains student records. Each student has a name, roll number, and three subject marks. You need to calculate their average and display formatted details.
Rubrics:
● Use __init__ to initialize student data.
● Create instance methods for average and display.
● Override __str__ for clean print.
● Instantiate multiple students.

---

2. ATM Simulation
Scenario: Simulate an ATM machine for a bank. A user can deposit, withdraw, and check their balance. The balance should not be directly accessible or modifiable from outside the class.
Rubrics:
● Use private variables for balance.
● Use instance and class variables correctly.
● Add validation for withdrawal limits.
● User-friendly interaction methods.

---

3. Bank with Static Interest Rate
Scenario: A bank wants to apply the same interest rate for all savings accounts. Create a class that calculates compound interest and allows updating the interest rate across all accounts.
Rubrics:
● Use static/class variables for interest rate.
● Define class method to update rate.
● Calculate interest with instance method.
● Show effect of change in interest rate.

---

4. Library Book Borrowing
Scenario: Model a library system where members can borrow books. Each book has a title, author, and availability status. A member should be able to borrow/return books and view their borrowed list.
Rubrics:
● Use two classes: Book and Member.
● Use composition (Member has Book).
● Track book status (borrowed/available).
● Implement borrowing/returning logic.

---

5. Shape Hierarchy

Scenario: Build a geometry toolkit. Define a base class Shape and derived classes like Circle, Rectangle, and Triangle with area calculations.

Rubrics:

● Inherit common behavior.

● Override area() method using polymorphism.

● Use base class reference for derived objects.

● Implement constructor chaining using super().

---

6. Car vs ElectricCar

Scenario: Create a simulation for a car showroom system. Define a class Car with properties like fuel type, and override behavior in ElectricCar to simulate electric-specific features.

Rubrics:

● Use inheritance and method overriding.

● Differentiate refuel() in base vs derived class.

● Demonstrate constructor reuse with super().

● Realistic attributes and methods.

---

7. Movie Ticket Booking

Scenario: Design a simple ticket booking system. Each theater has a limited number of seats. A customer can book a seat, and the system must track which seats are available.

Rubrics:

● Use static variable to track seat availability.

● Handle double booking with exception or message.

● Modular design with classes like Ticket, Customer.

● Display booking summary.

---

8. Product with Magic Methods

Scenario: Create a Product class that allows adding products together (e.g., combining total cost), comparing prices, and printing readable output.

Rubrics:

● Implement __add__, __str__, __repr__.

● Allow object addition to simulate combining products.

● Pretty output using magic methods.

● Input/output interaction.

---

9. Thermostat Controller

Scenario: Simulate a smart thermostat for a room that maintains temperature between 18°C and 30°C. It should not allow setting values beyond the allowed range.

Rubrics:

● Use encapsulation and validation via setters.

● Implement feedback on invalid input.

● Display current status using __str__.

● Private temperature variable.

10. School Staff and Students

Scenario: A school management system should differentiate between Staff and Students while sharing some common features like name and address. Staff has subjects taught, while students have grades.

Rubrics:

● Use inheritance from base Person class.

● Override methods in child classes.

● Demonstrate polymorphic display method.

● Use of isinstance() or issubclass().

Unit - I

Experiment 2

1. Game Character with Equipment

Scenario: Build a simple RPG system where each Character is equipped with a Weapon and Armor. These equipment pieces affect the character's total attack/defense power.

Rubrics:

● Use composition: Character has-a Weapon and Armor.

● Modular design of each class.

● Aggregated calculation of stats.

● Output full character summary.

---

2. Email Validation System

Scenario: Create a class that stores valid email addresses. The system should reject and not store invalid formats (e.g., missing '@', domain, etc.).

Rubrics:

● Use static methods for validation.

● Input list of emails and filter.

● Store only valid ones.

● Print valid email list.

---

3. Animal Zoo with Polymorphism

Scenario: Build a zoo simulator. Each animal has a speak() method. Animals like Lion, Monkey, and Snake inherit from the base Animal class and override behavior.

Rubrics:

● Implement class hierarchy.

● Demonstrate polymorphism using speak().

● Use a list of mixed animal objects.

● Loop to invoke behavior dynamically.

---

4. Payment Gateway with Abstraction

Scenario: Simulate a payment gateway with different modes like UPI, NetBanking, and CreditCard, each having their own implementation of the pay() method.

Rubrics:

● Use abstract base class Payment.

● Implement all derived classes.

● Show runtime selection of payment mode.

● Realistic flow and method design.

---

5. Object Counter

Scenario: Create a system that tracks how many user accounts have been created in a system.

Rubrics:

● Class variable to count instances.

● Class method to retrieve count.

● Use multiple objects to test.

● Display total objects created.

---

6. Function Logger Decorator

Scenario: Design a decorator that logs the function name, input, and output every time a function is called (e.g., for auditing).

Rubrics:

● Use @decorator.

● Capture arguments and return values.

● Include timestamp (optional).

● Apply to multiple functions.

---

7. Fibonacci Sequence Generator

Scenario: Write a generator function that yields values of the Fibonacci sequence indefinitely or until a given limit.

Rubrics:

● Use yield and maintain internal state.

● Output first N values.

● Memory-efficient implementation.

● Option to break after a limit.

---

8. Email Filter with Iterator

Scenario: Write a custom iterator that iterates through a list of strings and yields only valid email addresses.

Rubrics:

● Implement __iter__, __next__.

● Validate email in iteration.

● Stop iteration when list ends.

● Print all valid emails.

---

9. Temperature Conversion Closure

Scenario: Design a closure that generates a function to convert Celsius to Fahrenheit and vice versa, based on user input.

Rubrics:

● Use of closure with internal state.

● Convert both directions.

● Multiple closures for each direction.

● Clean function design.

---

10. Retry Logic with Decorators

Scenario: Decorate a function that simulates an API call to retry on failure (randomly generated failure simulation).

Rubrics:

● Retry mechanism in decorator.

● Use of try/except.

● Simulate random failures.

● Retry with limit.

Unit - I

Experiment 3

1. Shopping Cart with Generator

Scenario: Build a shopping cart system where total price is calculated using a generator expression.

Rubrics:

- Use generator in sum() call.
- Modular design with Cart and Product.
- Add/remove items.
- Print detailed bill.

---

2. Multiplication Table Iterator

Scenario: Create an iterator class that prints a multiplication table of any number up to 10.

Rubrics:

- Define custom iterator methods.
- Accept input number.
- Loop through results.
- Clear and structured output.

---

3. Discount Coupon Closure

Scenario: Generate discount functions using closures (e.g., 10%, 20% discounts).

Rubrics:

● Closure returns discount function.

● Apply on different item prices.

● Multiple closures with different rates.

● Output final price after discount.

---

4. Singleton Logger

Scenario: Build a logging system where all logs are saved in a single shared object.

Rubrics:

● Implement Singleton pattern.

● Use single instance for all logs.

● Store and print logs.

● Demonstrate uniqueness with id().

---

5. Notification Factory

Scenario: Create a factory that returns different notification classes based on input: Email, SMS, or Push.

Rubrics:

● Factory returns correct class.

● Each notification type has send() method.

● Use of inheritance or interfaces.

● Easy to extend with new types.

---

6. Weather Station Observer

Scenario: Simulate a weather station where multiple displays update temperature when the sensor changes.

Rubrics:

● Use observer pattern with subscribe/unsubscribe.

● Notify all observers on change.

● Decouple sensor from displays.

● Output updated data for each display.

---

7. Strategy Pattern Payment

Scenario: Allow users to select different payment strategies: CreditCard, UPI, Wallet.

Rubrics:

● Define strategy classes.

● Runtime strategy switch.

● Context class to execute.

● Realistic use case.

---

8. Vehicle Factory

Scenario: Simulate a rental service that returns different vehicle classes using a factory based on type (Bike, Car, Truck).

Rubrics:

● Factory returns appropriate object.

● Polymorphic rent() method.

● Clean vehicle hierarchy.

● Easy future expansion.

---

9. Sorter Strategy

Scenario: Implement different sorting algorithms (bubble, quick) as strategies that can be applied at runtime.

Rubrics:

● Implement strategy classes.

● Context class applies selected strategy.

● Demonstrate both sorting methods.

● Analyze pros and cons.

---

10. Stock Price Notifier

Scenario: Design a stock monitoring app that notifies all subscribed users when the price of a stock changes.

Rubrics:

● Use Observer pattern.

● Add/Remove subscribers.

● Notify on stock update.

● Print logs for each update.

Unit - II
Experiment 4

1. Scenario:- Climbing Stairs: You are developing a fitness app that tracks different ways users can climb a staircase. A user can climb either 1 step or 2 steps at a time. Your task is to calculate how many unique ways there are to climb a staircase with n steps.

Rubrics:
- Demonstrates overlapping subproblems
- Correct logic
- Measures inefficiency

2. Scenario:- You are climbing a staircase with n steps. You can take either 1 step or 2 steps at a time. Your task is to calculate how many unique ways there are to climb a staircase to reach the top.

Rubrics:
- Correct memoization usage
- Performance improves
- Understands top-down

3. Scenario:
A weather station records temperature every hour for a day (24 hours). Analyze daily trends.

Rubrics:
- Generate temperature data
- Compute min, max, and average

4. Scenario:
A robotic vacuum logs distance covered every minute during a cleaning session (45 minutes). Evaluate efficiency.
Rubrics:
- Generate distance data
- Compute total distance
- Find idle minutes (0 movement)
- Identify 5-minute window with max movement

5. Scenario:
An online learning platform records quiz scores across 20 sessions. Track user progress.

Rubrics:
- Generate random score data
- Slice the last 5 sessions
- Find min, max, and average scores
- Identify consistent improvement using difference between sessions

6. Scenario:
A finance app logs daily expenses for 30 days. Summarize spending behavior.

Rubrics:
- Generate random expense data
- Slice week 1 and week 4
- Calculate weekly totals and averages
- Identify no-spend days and streaks

7. Scenario:
A weather sensor logs temperature every hour over 24 hours. Detect anomalies.

Rubrics:

Generate hourly temperature readings

● Divide into morning/afternoon/evening

● Compute average for each time block

● Filter rapid temperature changes (difference > 5°C/hour)

8. Scenario:

A robot vacuum tracks movement in meters every minute over 60 minutes. Analyze cleaning efficiency.

Rubrics:

● Generate random movement data

● Slice first 20 minutes

● Compute total and average movement

● Filter idle minutes (0 meters moved)

9. Scenario:

A diet app tracks calorie intake per meal over a week (3 meals/day). Review diet balance.

Rubrics:

● Generate calorie data (21 values)

● Index specific meals (e.g., breakfast)

● Compute average per meal type

● Filter high-calorie meals (>700 calories)

10. Scenario:

A traffic monitoring system logs vehicle count per hour at a junction over 48 hours. Analyze traffic flow.

Rubrics:

● Generate vehicle count data

● Compare day vs night periods

● Compute moving averages (using DP concept)

● Filter congestion hours (count > 150 vehicles/hour)

Unit - II

Experiment 5

1. Scenario:

A fitness band records heart rate every minute for 10 minutes. Analyze this data.

Rubrics:

- Generate random heart rate data
- Index first 5 minutes
- Compute min, max, and average
- Filter high heart rate values (>120 bpm)

2. Scenario:

A language learning app tracks correct answers in each of 20 quiz sessions. Identify longest improvement streak (LCS-style problem).

Rubrics:

- Generate random correct answer counts
- Compare with a sorted version for pattern detection
- Use dynamic programming to find longest increasing sequence
- Track streaks of improvement

3. Scenario:

A delivery drone logs battery consumption for 15 trips. Maximize delivery value under limited battery (Knapsack scenario).

Rubrics:

- Generate trip values and battery costs
- Set a maximum battery capacity
- Select optimal subset of trips
- Compare results from top-down and bottom-up methods

4. Scenario:

A student revision tracker logs time spent per subject each day for 7 days. Find optimal study plan (similar to Knapsack).

Rubrics:

● Generate study hours and effectiveness score per subject

● Limit total hours per day

● Maximize total effectiveness

● Solve using bottom-up dynamic programming

5. Scenario:

A traffic monitoring system counts cars each minute at a junction. Find the longest stable flow (LCS-style).

Rubrics:

● Generate minute-wise traffic counts

● Define a stable flow range (e.g., 40–60 cars)

● Identify the longest sequence within range

● Use DP to track stable streaks

6. Scenario:

A music streaming app records listening durations for 50 tracks. Optimize playlist length (Fibonacci-style recurrence with choices).

Rubrics:

● Generate track durations

● Set a time constraint (e.g., 60 minutes)

● Find maximum number of non-adjacent tracks to fit

● Use memoization to store results of subproblems

7. Scenario:

A finance tracker logs daily expenses for 30 days. Find longest decreasing spending pattern (LCS-style variation).

Rubrics:

● Generate daily expense values

● Compare with decreasing sequence

● Use DP to find longest match

● Analyze savings behavior

8. Scenario:

A robot vacuum has to cover sections of a room with different energy costs. Maximize area cleaned under battery limit (Knapsack).

Rubrics:

● Generate cost and area for each section

● Set total energy budget

● Choose optimal subset of sections

● Use tabulation to optimize memory use

9. Scenario:

A stock app tracks price changes hourly. Predict future growth using previous trends (Fibonacci-style recurrence).

Rubrics:

● Generate hourly price changes

● Model price growth using recurrence relation

● Predict next value using DP

● Store computed values to avoid repetition

---

10. Scenario:

A travel app plans a road trip through cities with rewards at each stop. Maximize total reward without visiting adjacent cities (House Robber DP pattern).

Rubrics:

● Generate reward values per city

● Avoid adjacent city selection

● Apply bottom-up DP to calculate maximum reward

● Compare iterative vs. recursive approaches

Unit - II

Experiment 6

1. Scenario:

A fitness band records heart rate every minute for 10 minutes. Analyze this data.

Rubrics:

- Generate random heart rate data
- Index first 5 minutes
- Compute min, max, and average
- Filter high heart rate values (>120 bpm)

---

2. Scenario:

A delivery app calculates the shortest delivery time across multiple routes between warehouses and customers.

Rubrics:

- Generate a 2D matrix representing travel time between points
- Extract diagonal, row-wise, or column-wise paths
- Compute minimum path time using DP
- Compare different routes using bottom-up approach

---

3. Scenario:

A video editing app merges multiple video effects that have different processing costs. Optimize the order of merging (Matrix Chain Multiplication).

Rubrics:

- Generate a list of video segments with render costs
- Define multiplication cost between effects

● Apply optimal parenthesis placement logic

● Track minimum processing time using DP table

---

4. Scenario:

A genetics tool compares two DNA sequences to identify mutations (Sequence Alignment).

Rubrics:

● Generate two random DNA strings (A, T, G, C)

● Compute alignment score using gap penalties

● Track insertions, deletions, and substitutions

● Store scores in a 2D matrix using tabulation

---

5. Scenario:

An e-learning platform aligns students' answers to correct solutions to detect how far off their responses were (Edit Distance).

Rubrics:

● Generate two lists: student answers and correct answers

● Align sequences using edit operations (insert/delete/replace)

● Calculate minimum edit distance

● Visualize with a matrix or DP table

---

6. Scenario:

A navigation app calculates the fastest walking path through a grid of city blocks, considering blocked paths and delays.

Rubrics:

● Generate a 2D grid with random weights (walking times)

● Apply shortest path using bottom-up DP

● Mark blocked or delayed paths

● Return optimal total time and path index

---

7. Scenario:

A text comparison tool aligns two paragraphs to measure similarity (similar to LCS or sequence alignment).

Rubrics:

● Convert paragraphs into word/token sequences

● Use DP to compute longest common subsequence

● Generate similarity score

● Filter out unmatched tokens

---

8. Scenario:

A robot in a warehouse moves from top-left to bottom-right of a grid. Minimize total travel cost based on terrain difficulty.

Rubrics:

● Generate a 2D terrain cost matrix

● Track robot's movement path

● Use DP to compute minimum cost path

● Highlight obstacles or high-cost cells

---

9. Scenario:

A biomedical tool identifies the optimal way to fold proteins based on energy costs (Matrix Chain Multiplication analogy).

Rubrics:

● Generate a list of folding steps with transition costs

● Define multiplication cost based on energy levels

● Minimize total folding energy

● Track step-by-step energy reduction using DP

---

10. Scenario:

	A game AI plans movement across zones to reach a target with minimal health loss (Shortest Path with risk cost).											
	Rubrics:											
	● Generate grid with risk values per cell											
	● Define start and goal positions											
	● Compute path with lowest cumulative risk											
	● Visualize health drop along the path											