Experiment 1 - Bucket List

Q1	Define a class named Car with attributes make, model, and year. Create a constructor to initialize these attributes. Create an instance of Car and display its attributes. Concepts Covered: Class, Object, Constructor, Instance Variables
Q2	Create a class Calculator with two instance variables a and b. Implement instance methods to perform addition and subtraction. Add a static method to multiply two numbers without using the instance variables. Create an object of Calculator and demonstrate the usage of instance and static methods. Concepts Covered: Instance Methods, Static Methods
Q3	Define a class Person with a class variable population_count. Implement an __init__ method to initialize a person's name. Add a class method increment_population to increase population_count whenever a new person is created. Demonstrate the usage of the class method by creating multiple instances of Person. Concepts Covered: Class Method, Class Variables
Q4	Create a class Rectangle with attributes width and height. Implement the __init__ method to initialize these attributes. Override the __str__ and __repr__ methods to provide a string representation of the rectangle. Create an instance and print it to see the custom string representation. Concepts Covered: Magic Methods (__init__, __str__, __repr__)
Q5	Define a base class Animal with an attribute name and a method make_sound(). Create a derived class Dog that inherits from Animal and overrides the make_sound() method to print "Woof". Instantiate a Dog object and demonstrate calling the overridden method. Concepts Covered: Inheritance, Method Overriding
Q6	Implement a class BankAccount with private attributes balance and account_number. Provide public methods to deposit and withdraw money, and to check the balance. Ensure that the balance cannot be set directly from outside the class. Concepts Covered: Encapsulation, Private Variables
Q7	Create an abstract base class Shape with an abstract method area(). Define two derived classes, Circle and Rectangle, that implement the area() method. Instantiate both classes and print their areas. Use the abc module for abstract base class implementation. Concepts Covered: Abstraction, Abstract Base Class
Q8	Create a base class Shape with a method draw(). Define two derived classes, Circle and Square, each implementing the draw() method in their own way. Create a list of Shape objects and call the draw() method on each object to demonstrate polymorphism. Concepts Covered: Polymorphism, Method Overriding
Q9	Define a class Engine with attributes horsepower and type. Create a class Car that contains an Engine object as an attribute. Implement methods in Car to get engine details and display car details along with engine details. Concepts Covered: Composition
Q10	Create a class Library with a class variable book_count to keep track of the number of books. Implement an instance method add_book() that increases the book_count. Create a method to get the total number of books from the class variable. Demonstrate adding books and retrieving the count. Concepts Covered: Static Variables, Class Variables, Instance Methods

Experiment 2 - Bucket List

Q1	Define a decorator function add_10 that adds 10 to the result of the function it decorates. Use this decorator on a function get_number that returns a number. Demonstrate the usage by calling the decorated function and printing the result. Concepts Covered: Decorators
Q2	Create a class Logger with a method log that prints a message before and after the execution of a function. Use an instance of Logger as a decorator for a method compute in a class MathOperations. Demonstrate by calling the decorated method. Concepts Covered: Class-based Decorators
Q3	Define a class Countdown that takes a starting number and counts down to 0. Implement the __iter__ and __next__ methods to make Countdown an iterator. Create an instance and use a for-loop to iterate through the countdown sequence. Concepts Covered: Iterators
Q4	Create a class Fibonacci that generates the Fibonacci sequence up to a specified number of elements. Implement the __iter__ and __next__ methods to make Fibonacci an iterator. Print the first 10 numbers in the Fibonacci sequence. Concepts Covered: Custom Iterable, Iterators
Q5	Define a generator function square_numbers that yields the squares of numbers from 1 to 5. Iterate through the generator to print each squared number. Concepts Covered: Generators
Q6	Implement a generator function fibonacci_gen that yields Fibonacci numbers up to a specified limit. Use this generator to print the Fibonacci numbers up to 100. Concepts Covered: Generators
Q7	Create a closure function make_multiplier that takes a number n and returns a function that multiplies its input by n. Test this closure by creating a multiplier function for doubling and tripling values, and demonstrate its usage. Concepts Covered: Closures
Q8	Define a class Counter with a method increment that increases a count by a specified value. Use a decorator to print the current count before and after the increment operation. Implement a closure to ensure the count is properly incremented within the method. Concepts Covered: Class with Decorator and Closure
Q9	Create a generator function count_up_to that counts from 1 to a given number. Maintain internal state in the generator to keep track of the current number. Use this generator to count up to 7 and print the numbers. Concepts Covered: Generators with State
Q10	Write a decorator repeat that takes a parameter times and repeats the execution of the decorated function say_hello a specified number of times. Demonstrate the decorator by calling the say_hello function with times set to 3. Concepts Covered: Decorators with Parameters

Experiment 3 - Bucket List

Q1	Implement a Singleton class named DatabaseConnection that ensures only one instance of the class can exist. Create a method get_instance that returns the single instance of the class. Demonstrate that multiple calls to get_instance return the same instance. Concepts Covered: Singleton Pattern
Q2	Extend the DatabaseConnection Singleton class to include a method connect that prints a connection message. Demonstrate that only one instance is used even when calling connect from different parts of the program. Concepts Covered: Singleton Pattern with Methods
Q3	Define a ShapeFactory class with a method create_shape that returns different shape objects (Circle or Square) based on the input string. Implement Circle and Square classes with a method draw that prints the shape name. Demonstrate creating and drawing shapes using the factory. Concepts Covered: Factory Pattern
Q4	Create an abstract base class Animal with a method make_sound. Implement two concrete classes Dog and Cat that inherit from Animal and override make_sound. Define an AnimalFactory with a method create_animal to return Dog or Cat instances based on input. Demonstrate the factory in action. Concepts Covered: Factory Pattern with Abstract Base Class
Q5	Implement a simple observer pattern with a Subject class that manages a list of Observer objects. Implement an Observer interface with an update method. Create concrete Observer classes (EmailNotifier and SMSNotifier) that implement the update method. Demonstrate the observer pattern by notifying observers of a change in the subject. Concepts Covered: Observer Pattern
Q6	Extend the observer pattern example to include a method add_observer in the Subject class to add observers. Implement a method notify_observers that calls update on each observer. Demonstrate adding and notifying multiple observers. Concepts Covered: Observer Pattern with Observer Management
Q7	Implement a SortStrategy interface with a method sort. Create two concrete strategies, BubbleSort and QuickSort, each implementing the sort method. Create a Sorter class that uses a SortStrategy to sort a list. Demonstrate sorting a list with both strategies. Concepts Covered: Strategy Pattern
Q8	Extend the strategy pattern example to include a Context class that uses a SortStrategy instance to sort data. Implement a method set_strategy to change the sorting strategy dynamically. Demonstrate changing strategies at runtime. Concepts Covered: Strategy Pattern with Context
Q9	Implement a Singleton class Configuration that reads configuration settings from a file. Ensure only one instance of Configuration exists and is used throughout the application. Add a method get_setting to retrieve configuration values. Demonstrate accessing configuration settings. Concepts Covered: Singleton Pattern with Configuration
Q10	Define a VehicleFactory class with a method create_vehicle that creates Car or Bike objects based on input parameters (e.g., type). Implement Car and Bike classes with a drive method. Demonstrate creating and using vehicles based on factory parameters. Concepts Covered: Factory Pattern with Parameters

Experiment 4 - Bucket List

Q1	Write a program to find the longest common substring between two strings.
Q2	Implement a program to calculate the minimum number of operations (insertions, deletions, substitutions) required to convert one string into another
Q3	Design a program to find the longest subsequence in a string that is also a palindrome.
Q4	Write a program to find the shortest sequence that has both input sequences as subsequences.
Q5	Develop a program to find the longest subsequence in an array of integers that is strictly increasing.
Q6	Implement a program to find the longest subsequence that appears more than once in a given string.
Q7	Create a program to find the contiguous subarray within a one-dimensional array of numbers that has the largest sum.
Q8	Write a program to determine the maximum value that can be obtained by putting items in a knapsack with a given weight capacity.
Q9	Develop a program to find the most efficient way to multiply a chain of matrices.
Q10	Design a program to determine whether a subset of a given set of numbers adds up to a specific target sum.

Experiment 5 - Bucket List

Q1	Implement a program to solve the fractional knapsack problem where you can take fractions of items.
Q2	Write a program to solve the knapsack problem where you can take an unlimited number of each item.
Q3	Design a program to determine the minimum number of coins needed to make a specific amount of money from a given set of coin denominations.
Q4	Develop a program to determine the maximum profit obtainable by cutting a rod of a given length into pieces with given values.
Q5	Implement a program to find whether a subset of a given set of integers sums up to a given target.
Q6	Write a program to determine whether a given set can be partitioned into two subsets with equal sums
Q7	Design a program to find the maximum profit you can make by scheduling jobs within their deadlines, considering their respective profits.
Q8	Create a program to find the length of the longest palindromic subsequence in a given string.
Q9	Implement a program to find the length of the shortest common supersequence of two strings.
Q10	Develop a program to maximize the value obtained by cutting a rod into various lengths based on given prices for each length

Experiment 6 - Bucket List

Q1	Implement a program to calculate the nth Tribonacci number, where each term is the sum of the previous three terms.
Q2	Write a program to determine how many distinct ways you can climb a staircase with n steps, where you can take 1 or 2 steps at a time.
Q3	Create a program to calculate the number of unique paths from the top-left corner to the bottom-right corner of a grid, moving only right or down
Q4	Develop a program to calculate the nth Catalan number, which appears in various combinatorial problems.
Q5	Write a program to determine the number of ways to tile a 2 x n board using 2 x 1 tiles.
Q6	Implement a program to find the length of the longest increasing subsequence in an array of numbers
Q7	Design a program to generate all combinations of n pairs of balanced parentheses.
Q8	Create a program to find the minimum number of jumps needed to reach the end of an array, where each element represents the maximum jump length at that position.
Q9	Develop a program to find the minimum number of attempts needed to find the critical floor in a building with k floors using n eggs.
Q10	Write a program to calculate the nth Bell number, which represents the number of ways to partition a set of n elements.