**🧩 Part A – Basic Class and Object Concepts**

1. **Create a class** Car with attributes brand and model, and a method display\_info() to print the details.
2. Write a Python class Book that takes title and author as input and has a method show\_details() to display them.
3. Define a class Rectangle that has attributes length and width, and methods to calculate **area** and **perimeter**.
4. Create a class Employee with name and salary. Add a method get\_salary() to return the salary and a method increment() to increase salary by 10%.
5. Write a class Temperature with methods to convert from **Celsius to Fahrenheit** and **Fahrenheit to Celsius**.

**⚙️ Part B – Constructor and Self**

1. Create a class Laptop with attributes brand, RAM, and price. Use \_\_init\_\_() constructor to initialize them and a method show\_config() to print details.
2. Write a class Circle with a constructor that accepts radius, and methods to calculate **area** and **circumference** using the formula πr² and 2πr.

**🔒 Part C – Encapsulation**

1. Create a class BankAccount with private attribute \_\_balance. Add methods deposit(amount), withdraw(amount), and get\_balance(). Prevent direct access to \_\_balance.
2. Write a class StudentMarks with private data members for marks in 3 subjects. Add a method calculate\_percentage() and ensure marks can be updated only through a setter method.

**🧬 Part D – Inheritance**

1. Create a base class Vehicle with method show\_type(), and a derived class Bike that adds a method show\_model(). Demonstrate inheritance.
2. Define a parent class Person with attributes name and age. Create a child class Employee that adds employee\_id and method show\_details() which calls parent’s method using super().
3. Write a program using **multilevel inheritance**: Person → Student → GraduateStudent. Each should have a display() method showing its own info.

**🌀 Part E – Polymorphism**

1. Create two classes Cat and Dog, both having a speak() method. Write a common function animal\_sound(animal) that calls speak() on any object passed to it.
2. Create a base class Shape with a method area(). Derive two classes Rectangle and Circle that override the area() method differently.

**⚠️ Part F – Exception Handling & Custom Exceptions**

1. Write a custom exception LowBalanceError that is raised when withdrawal amount is greater than balance. Use a try-except block to handle it in a BankAccount class.