1. Introduction: From Procedural to Object-Oriented Programming

1.1 What is Object-Oriented Programming?

Object-Oriented Programming (OOP) is a programming paradigm that organizes code around **objects** rather than functions and logic. An object is a data structure that contains both **data** (attributes) and **code** (methods) that operates on that data.

Key Paradigm Comparison:

Aspect	Procedural (C)	Object-Oriented (C++)
Focus	Functions and procedures	Objects and classes
Data & Functions	Separate	Bundled together
Code Organization	By functionality	By entities/objects
Data Protection	Limited (global/local)	Strong (access specifiers)
Code Reuse	Function reuse	Inheritance & polymorphism
Maintenance	Harder for large projects	Easier through modularity

1.2 The Four Pillars of OOP

1. **Encapsulation**: Bundling data and methods that operate on that data within a single unit (class), hiding internal details
2. **Abstraction**: Showing only essential features while hiding implementation details
3. **Inheritance**: Creating new classes from existing classes, promoting code reuse
4. **Polymorphism**: Ability of objects to take many forms, allowing different implementations of the same interface

1.3 Real-World Analogy

Think of a **car**:

- **Object**: Your specific car (e.g., a red Toyota Camry 2020)
- **Class**: The blueprint/design for all Toyota Camry cars
- **Attributes** (data): color, model, year, speed, fuel level
- **Methods** (functions): start(), accelerate(), brake(), turn()

- **Encapsulation**: You don't need to know how the engine works internally; you just use the steering wheel and pedals
- **Abstraction**: The dashboard shows you speed and fuel, hiding complex engine computations

---

Revision #3
Created 2025-09-10 01:51:03 UTC by YP
Updated 2025-11-09 13:14:01 UTC by DS