Design Patterns in Python

Academic Syllabus and Module Overview

Instructor: Ayush Singh
Email: ayush@secondbrainlabs.com

Academic Term: 2025–2026

Abstract

This course introduces the most essential design patterns in Python through a thinking-first and project-based approach. Students will learn to identify bad code structures, understand why they fail in real-world scenarios, and apply appropriate design patterns to refactor them. Each module focuses on a single pattern using hands-on mini-projects, allowing learners to internalize the value and practicality of design patterns in software engineering.

COURSE OBJECTIVES

- Understand the purpose and application of key software design patterns.
- Recognize code smells and architectural pitfalls in real-world projects.
- Develop the ability to refactor poorly designed code into maintainable and extensible structures.
- Apply design principles such as SOLID and DRY through practical coding exercises.
- Build software with greater readability, reusability, and scalability.

TARGET AUDIENCE

This course is designed for advanced school students, university undergraduates, coding bootcamp learners, and early-career developers who have a basic understanding of Python and want to learn real-world software design.

COURSE STRUCTURE

Each module follows a four-step pedagogy:

1. **Problem Realization** — Demonstrate flawed design in a practical scenario.
2. **Concept Discovery** — Explore the design pattern that solves the problem.
3. **Code Refactoring** — Apply the pattern with a real mini-project.
4. **Reflection** — Review benefits and trade-offs of the applied pattern.

COURSE MODULES

Module 1: Factory Method

Mini Project: Report Exporter (PDF, CSV, JSON)
Core Objective: Replace conditionals with a factory-driven object generator.
Key Concepts: Open-Closed Principle, object creation control.

Module 2: Builder Pattern

Mini Project: Resume Generator
Core Objective: Simplify the construction of complex objects using fluent APIs.
Key Concepts: Step-by-step creation, telescoping constructor issue.

Module 3: Singleton Pattern

Mini Project: Logger / Config Loader
Core Objective: Enforce single-instance logic for shared resources.
Key Concepts: Global state, resource control, service objects.

Module 4: Adapter Pattern

Mini Project: Unified Payment Gateway
Core Objective: Normalize incompatible external interfaces.
Key Concepts: Interface mapping, dependency isolation.

Module 5: Strategy Pattern

Mini Project: Discount Engine
Core Objective: Inject behavior dynamically at runtime.
Key Concepts: Algorithm encapsulation, behavioral flexibility.

Module 6: Decorator Pattern

Mini Project: Web Route Enhancements
Core Objective: Add responsibilities dynamically without altering class code.
Key Concepts: Wrapper chaining, flexible enhancement.

Module 7: Observer Pattern

Mini Project: Notification Dispatcher
Core Objective: Build a loosely-coupled event system.
Key Concepts: Publish-subscribe, reactive architecture.

Module 8: State Pattern

Mini Project: Media Player
Core Objective: Represent dynamic behavior with internal state objects.
Key Concepts: State transitions, polymorphic behavior.

Module 9: Command Pattern

Mini Project: Task Scheduler with Undo/Redo
Core Objective: Encapsulate user actions as first-class objects.
Key Concepts: Action logs, reversibility, macros.

Module 10: Proxy Pattern

Mini Project: Lazy Image Loader / Access Guard
Core Objective: Add control, delay, or access restrictions.
Key Concepts: Lazy loading, logging proxy, auth proxy.

CAPSTONE MODULE: PATTERN INTEGRATION PROJECT

- Combine at least 3 design patterns into a single cohesive mini-application.
- Students will choose from: Quiz App, E-commerce Cart, CMS Blog, or propose their own.
- Final deliverables include source code, pattern diagram, and a written reflection.

ASSESSMENT CRITERIA

- Pattern-based project submissions – 40%
- Concept quizzes and code comprehension – 15%
- Peer code reviews and participation – 15%
- Capstone project – 30%

COURSE TOOLS AND REQUIREMENTS

- Python 3.10+
- GitHub for submissions and version control
- VS Code or PyCharm IDE
- Online compilation tools (optional): Replit, Jupyter Notebooks

Prepared by: Ayush Singh
SecondBrain Labs, 2025