SOLID Principles

SOLID is a set of five object-oriented design principles that can help you write more maintainable, flexible, and scalable code based on well-designed, cleanly structured classes. These principles are a fundamental part of object-oriented design best practices.

SOLID is an acronym that groups five core principles that apply to object-oriented design.

These principles are the following:

- **Single-responsibility principle (SRP)**
  A class should have only one reason to change, meaning that it should have only one responsibility.

- **Open–closed principle (OCP)**
  A class should be open for extension but closed for modification. This means that you should be able to extend the behavior of a class without changing its source code.

- **Liskov substitution principle (LSP)**
  Subtypes must be substitutable for their base types. This means that if you have a class hierarchy, you should be able to use any subclass in place of its superclass without affecting the correctness of the program.

- **Interface segregation principle (ISP)**
  Clients should not be forced to depend on interfaces they do not use. This means that you should split large interfaces into smaller ones, so that clients only need to depend on the interfaces that are relevant to them.

- **Dependency inversion principle (DIP)**
  High-level modules should not depend on low-level modules. Both should depend on abstractions. Abstractions should not depend on details. Details should depend on abstractions. This means that you should use dependency

injection and inversion of control to decouple your code and make it more testable and maintainable.

By following these principles, you can create code that is easier to understand, maintain, and extend over time.