1. Single Responsibility Principle (SRP)

Each class should have only one reason to change, meaning it should have only one job or responsibility.

2. Open/Closed Principle (OCP)

Software entities (classes, modules, functions, etc.) should be open for extension but closed for modification. This means you can add new functionality without altering existing code.

3. Liskov Substitution Principle (LSP)

Objects of a superclass should be replaceable with objects of a subclass without affecting the correctness of the program. Essentially, subclasses should behave in a way that won't break the functionality of the parent class.

4. Interface Segregation Principle (ISP)

No client should be forced to depend on interfaces it does not use. It's better to have many small, specific interfaces rather than one large, general-purpose interface.

5. 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 (concrete implementations) should depend on abstractions.

These principles help in creating more maintainable, flexible, and scalable software systems by encouraging good design practices.