How to avoid problems that require using SOLID principles

The SOLID principles are a set of design principles that can help you write code that is more modular, maintainable, and scalable. Here are some tips for avoiding problems that require using the SOLID principles:

1. **Keep your functions and classes small:** Large functions and classes can be difficult to understand and maintain. By keeping your functions and classes small and focused on a single responsibility, you can reduce the risk of code complexity and make it easier to modify or extend your code.
2. **Follow the single responsibility principle:** The single responsibility principle (SRP) states that each function or class should have a single responsibility. By following this principle, you can reduce the risk of code complexity and ensure that your code is more modular and easier to maintain.
3. **Use dependency injection:** Dependency injection (DI) is a technique that allows you to inject dependencies into a class rather than hard-coding them. By using DI, you can reduce the risk of tight coupling between your classes and make it easier to modify or extend your code.
4. **Avoid global state:** Global state can make your code hard to understand and debug, and can cause unexpected behavior. By avoiding global state and using local variables instead, you can reduce the risk of side effects and make your code more predictable.
5. **Use interfaces:** Interfaces are a powerful tool for decoupling your code and making it more modular. By using interfaces, you can reduce the risk of tight coupling between your classes and make it easier to modify or extend your code.