The Differences in Program Design Between Functional Programming and Object-Oriented Programming:

1. **State Management**:
    - In object-oriented programming (OOP), state is typically encapsulated within objects. Objects have private attributes and methods to modify these attributes. The change in state is a core concept of OOP, and an object's state can evolve throughout its lifecycle.
    - In functional programming (FP), immutability and pure functions are emphasized. Data is immutable, and functions neither modify external states nor rely on them. Each function call produces new data rather than altering existing data.
2. **Data and Behavior**:
    - In OOP, data and the methods that manipulate it are encapsulated within objects. This tightly couples state and behavior.
    - In FP, data and behavior are handled separately. Data is generally immutable, and behavior (functions) operates on the data. This separation makes functions easier to compose and reuse.
3. **Code Reuse**:
    - OOP relies on class inheritance to share and extend behavior. Inheritance often results in deep class hierarchies, which can increase code complexity.
    - FP, on the other hand, tends to use function composition and higher-order functions to build complex behaviors instead of inheritance. This approach reduces coupling and enhances modularity.

Changes Made to Follow Functional Programming Principles:

1. **Removal of the StyleChecker Class**:
    - Methods of the StyleChecker class were converted into independent functions. These functions no longer depend on the state of class instances but instead receive the required data through parameters.
2. **Replacement of self.content**:
    - The instance variable self.content was replaced with function parameters. Each function call receives fresh data instead of modifying an existing state.
3. **Pure Function Design**:

- All functions were designed as pure functions, meaning they do not depend on external state and interact with the outside world solely through their inputs and outputs.

4. **Isolation of I/O Operations**:
    - All I/O operations were confined to the save_report and analyze_tree functions. These functions are solely responsible for data input and output, while the business logic is handled separately.