**Code quality review according to** [**Code Review Checklist**](https://www.codereviewchecklist.com/)

**Pyramic XO:**

**Requirements**

• Met Requirements: The code appears to fulfill the basic requirements of implementing PyramicTicTacToe.

• Correct Formatting: Code is correctly formatted.

• Unnecessary Whitespace: Whitespace could be further reduced in some areas.

**Best Practices**

• Single Responsibility: Code seems to follow this principle, separating concerns into different classes.

• Error Handling: Basic error handling exists but might need improvement for more comprehensive error prevention.

• Logging Errors/Warnings: No explicit logging for errors or warnings in the provided code.

• Magic Values: Constants could replace some hardcoded values.

• Comments: Comments are present but might be insufficient for complex logic understanding.

• Nesting: Generally minimal nesting.

**Maintainability**

• Readability: Code readability is moderate, but more comments could enhance it.

• Code Duplication: Some sections might benefit from refactoring to remove duplication, but they are all necessary for functionality.

• Method/Class Length: Functions are all manageable in size.

**Performance**

• Performance Acceptance: Basic operations seem fine.

**Architecture**

• Separation of Concerns: Different aspects are separated into classes, following a basic separation of concerns.

**Four-in-a-row:**

1. **Requirements:**

• The code seems to fulfill the required functionality.

1. **Code Formatting:**

• The code has some inconsistencies in formatting, especially in the display\_board function.

•Unnecessary white spaces are removed.

1. **Best Practices:**

• Single Responsibility Principle (SRP): The ConnectFourBoard class handles the game board and related logic, adhering to the SRP.

• Error Handling: Some error handling is present, but it could be more explicit, especially in the update\_board function.

• Magic Values: Some magic values that can be replaced with constants are present.

• Comments: There are sufficient comments explaining logic of the code.

• Nesting: The nesting is minimal, and the code is generally easy to follow.

1. **Maintainability:**

• The code is relatively easy to read.

• The methods in the class are not excessively long, contributing to maintainability.

1. **Performance:**

• The code is straightforward and performs without any issues.

1. **Architecture:**

• There are no apparent security risks in the code

• The parameters are hard-coded in some places instead of being passed as arguments.

1. **Testing:**

•The code passes manual test plans