Refactoring Report

Overview

The code in main.py was reviewed and improved through the application of appropriate refactoring techniques. The main goal was to enhance readability, reduce duplication, and ensure maintainability while preserving the existing functionality.

Refactoring Techniques Applied

1. Extract Function - Repeated logic for discount checks was moved into helper functions: is_eligible_for_discount and is_potential_discount_customer.

2. Extract Variable - Complex expressions, such as the tax calculation in calculate_total_with_tax, were assigned to clearly named variables like taxed_price to improve readability.

3. Rename Variable - Variable names were updated for clarity. For example, loop variables were renamed from generic terms to descriptive names like purchase to better indicate their purpose.

4. Decompose Conditional - The conditional logic in generate_report was split into simpler, clearer branches, reducing nested structures and improving readability.

5. Parametrise Function - The shipping fee logic was generalized using a function that accepts a condition and fee values as parameters. This enabled reusing the same logic for heavy and fragile item shipping fee calculations.

6. Move Function - The shipping fee functions were placed appropriately to reduce duplication and maintain logical grouping.

Verification of Behaviour

To confirm that the refactored code preserved the original behavior, existing test cases were reviewed and additional tests were added to cover previously untested branches. This included tests for: Customers eligible for potential future discounts, Customers not eligible for any discounts, Customers qualifying as VIP or Priority based on their total spending, Shipping fee calculations for both matching and non-matching conditions. Running the full test suite after refactoring confirmed that all observable behavior remained consistent.

Benefits and Drawbacks Benefits

Improved readability and clarity of the code. Reduced duplication by centralizing reusable logic. Easier future maintenance and extension, such as adding new shipping fee rules. Full test coverage ensures reliability.

Potential Drawbacks:

Introducing additional helper functions slightly increases the number of abstractions, which may require a short learning curve for new developers unfamiliar with the codebase.

Reflections

The refactoring process was beneficial in cleaning up the code and making it more maintainable. The improved structure reduces the risk of bugs in future changes and makes it easier to understand and extend the system. Full test coverage ensures confidence that behavior remains unchanged.