Code smells are indicators in your codebase that suggest there might be deeper problems in its structure, design, or implementation. They aren't bugs or errors that cause your code to break; instead, they point to suboptimal practices that can lead to maintenance issues, reduced readability, or increased complexity over time. Identifying and addressing code smells helps in improving code quality and maintainability.

Common Examples of Code Smells:

1. **Duplicated Code**
    - The same code exists in multiple places.
    - *Why it's bad:* Hard to maintain. If a change is needed, it must be updated everywhere.
    - Example: Copy-pasting the same logic in multiple functions.
2. **Long Method**
    - A method does too many things and has grown too large.
    - *Why it's bad:* Hard to read, understand, and debug.
    - Example: A 200-line function handling multiple unrelated tasks.
3. **Large Class**
    - A class is overloaded with responsibilities.
    - *Why it's bad:* Violates the Single Responsibility Principle, making it harder to maintain and test.
    - Example: A User class that handles database operations, business logic, and UI formatting.
4. **Feature Envy**
    - A method in one class relies heavily on the data or methods of another class.
    - *Why it's bad:* Indicates poor cohesion and that functionality might belong elsewhere.
    - Example: A Cart class frequently accessing Product attributes instead of delegating tasks to Product.
5. **Long Parameter List**
    - A function or method requires too many parameters.

- *Why it's bad:* Makes the code harder to understand and prone to errors.
- Example: calculate_salary(base, bonus, tax, overtime, deductions, hours_worked)

6. **Data Clumps**
    - Related data items frequently appear together but aren't grouped into a structure (e.g., a class).
    - *Why it's bad:* Leads to inconsistent handling of the same set of variables.
    - Example: Passing x, y, and z coordinates around instead of encapsulating them in a Point object.
7. **Dead Code**
    - Unused variables, functions, or code blocks.
    - *Why it's bad:* Adds clutter and confusion.
    - Example: A function that's defined but never called.
8. **God Object**
    - A single object/class that knows too much or does too much.
    - *Why it's bad:* Becomes a bottleneck for modifications and testing.
    - Example: A Manager class that handles data fetching, UI rendering, and API calls.
9. **Shotgun Surgery**
    - A small change in one place forces changes in multiple unrelated parts of the code.
    - *Why it's bad:* Indicates tight coupling and lack of modularity.
    - Example: Changing the format of a single data field requires updating code in dozens of files.
10. **Primitive Obsession**
    - Overuse of basic data types instead of creating more meaningful abstractions.
    - *Why it's bad:* Leads to loss of context and increases the chance of errors.
    - Example: Using String to represent a phone number instead of a

PhoneNumber class.

How to Fix Code Smells:

- Refactor your code by applying principles of good software design, like **DRY** (Don't Repeat Yourself), **SOLID**, and **KISS** (Keep It Simple, Stupid).
- Use techniques like:
    - Extract Method (pull out smaller methods from large methods).
    - Introduce Parameter Object (replace long parameter lists with a single object).
    - Move Method (place a method in the class it interacts with most).
    - Replace Conditional with Polymorphism (eliminate large if-else chains).

Code smells aren't inherently "bad," but they are warnings that the code could benefit from improvement. Regular refactoring and adhering to best practices help mitigate code smells.