Assignment1: Create an infographic illustrating the Test-Driven Development (TDD) process. Highlight steps like writing tests before code, benefits such as bug reduction, and how it fosters software reliability.

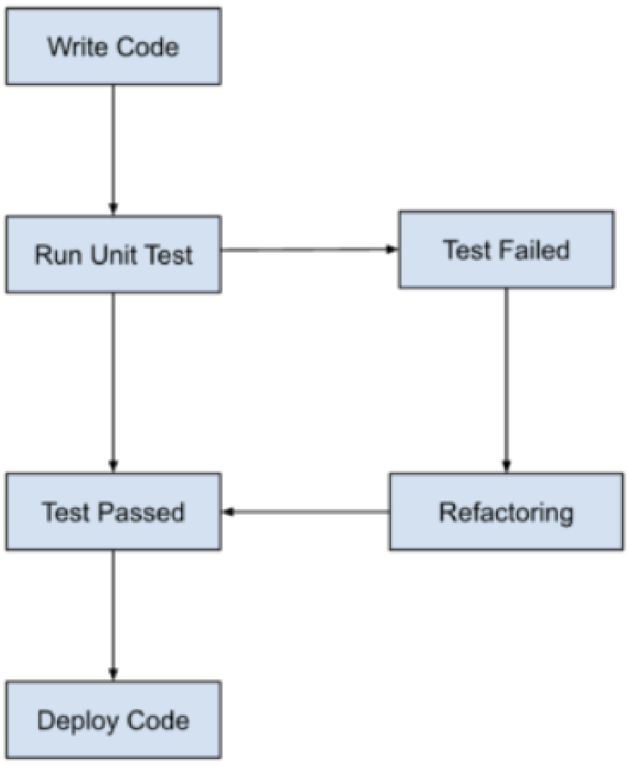

Fig: **TDD approach toward development**

- Test-Driven Development (TDD) is a software development approach where tests are written before the code is implemented.

Key Steps of TDD:

1. **Write a Test**:
    - Developers write a test case that defines the desired behaviour of a small unit of code.
    - Test initially fails since the code to implement the functionality hasn't been written yet.
2. **Write the Code**:
    - Implement the minimal amount of code necessary to make the test pass.
    - Focus on meeting the requirements of the test case.
3. **Run the Test**:

- Execute all tests to ensure new test case passes and existing functionality remains intact.
- Immediate feedback helps catch bugs early in the development process.

4. **Test Failed**:
    - If the test fails, developers revise the code until it passes the test.
    - This iterative process continues until the test passes.
5. **Test Passed**:
    - Once the test passes, the code meets the requirements specified by the test case.
    - Developers can proceed to the next step with confidence.
6. **Refactor**:
    - Optimize and improve code structure without changing its external behaviour.
    - Ensure code remains clean, maintainable, and adheres to coding standards.
7. **Deploy Code**:
    - After passing all tests, deploy the code to the production environment.
    - Continuous integration and automated deployment pipelines streamline this process.

Benefits of TDD:

- **Bug Reduction**: By writing tests before code, developers catch bugs early in the development process, reducing the likelihood of introducing defects.
- **Increased Reliability**: Comprehensive test suites provide confidence that the code works as intended and helps prevent regressions during future changes.
- **Improved Design**: TDD encourages modular and loosely coupled designs, leading to more maintainable and scalable software.
- **Faster Development**: Immediate feedback from tests allows developers to iterate quickly, resulting in faster development cycles.

Assignment 2: Produce a comparative infographic of TDD, BDD, and FDD methodologies. Illustrate their unique approaches, benefits, and suitability for different software development contexts. Use visuals to enhance understanding.

1. **Test-Driven Development (TDD)**:
    - **Approach**: Write tests before writing code, following a cycle of "Red-Green-Refactor" where tests are written, code is implemented to pass those tests, and then code and tests are refactored as needed.

- **Benefits**:
    - Ensures that code is thoroughly tested from the outset.
    - Promotes a modular and loosely coupled codebase.
    - Provides living documentation in the form of tests.
- **Suitability**:
    - Ideal for projects with clear and well-defined requirements.
    - Effective for building maintainable and robust codebases.
    - Particularly useful for small to medium-sized projects.

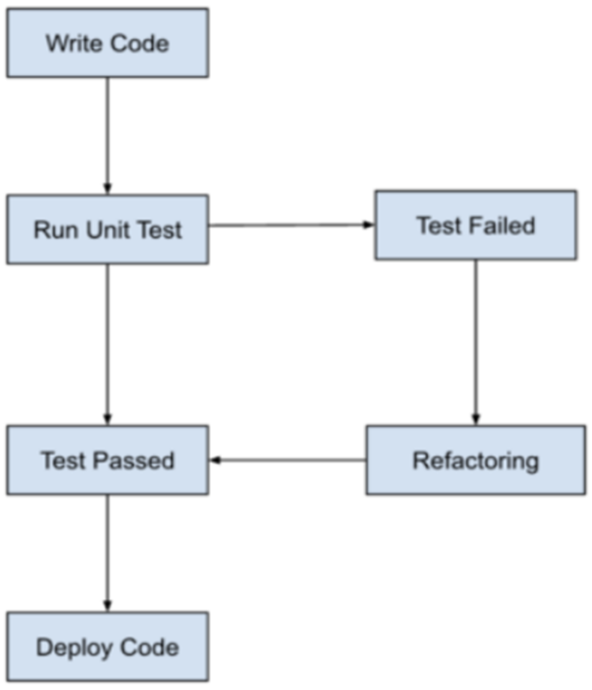

Fig: **TDD approach toward development**

2. **Behaviour-Driven Development (BDD)**:
    - **Approach**: Focuses on defining behaviour through examples written in a domain-specific language (DSL), such as Gherkin syntax, to facilitate collaboration between developers, testers, and business stakeholders.
    - **Benefits**:
        - Encourages collaboration and shared understanding between stakeholders.
        - Helps ensure that software behaviour aligns with business objectives.
        - Provides executable specifications that serve as living documentation.

- **Suitability**:
    - Well-suited for projects with complex business logic or user interactions.
    - Particularly effective for projects where clear communication between stakeholders is essential.
    - Useful for driving development from the perspective of user behaviour and business requirements.

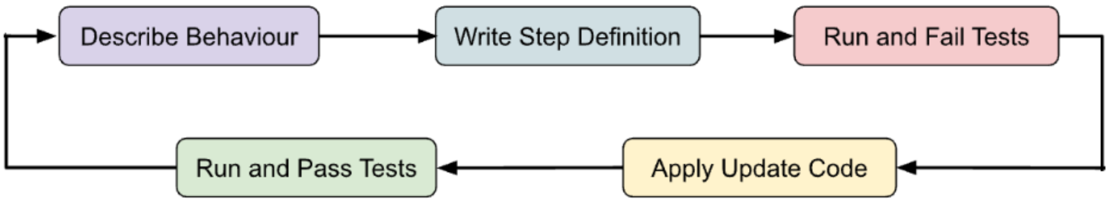

Fig: BDD approach toward development.

3. **Feature-Driven Development (FDD)**:
    - **Approach**: Divides development into small, feature-focused iterations, with a strong emphasis on domain modelling, feature lists, and progress tracking.
    - **Benefits**:
        - Promotes a disciplined and structured approach to development.
        - Facilitates efficient progress tracking and management of feature delivery.
        - Supports the scalability of development teams and projects.
    - **Suitability**:
        - Suitable for large-scale projects with complex requirements.
        - Effective for projects with a need for clear feature prioritization and tracking.
        - Particularly useful for distributed development teams or projects with multiple stakeholders.

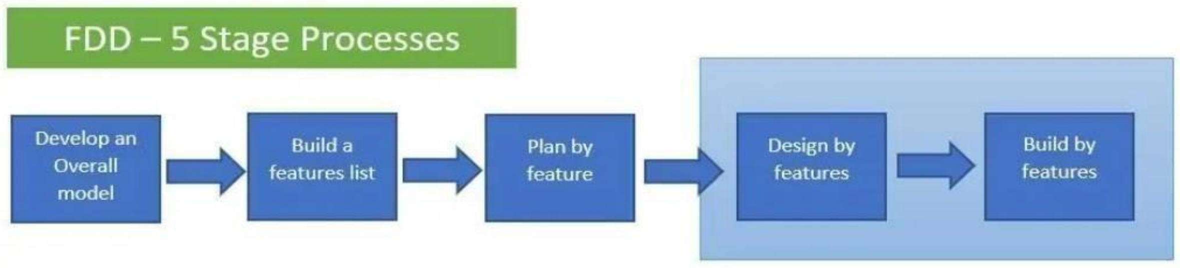

Fig: FDD approach toward development.