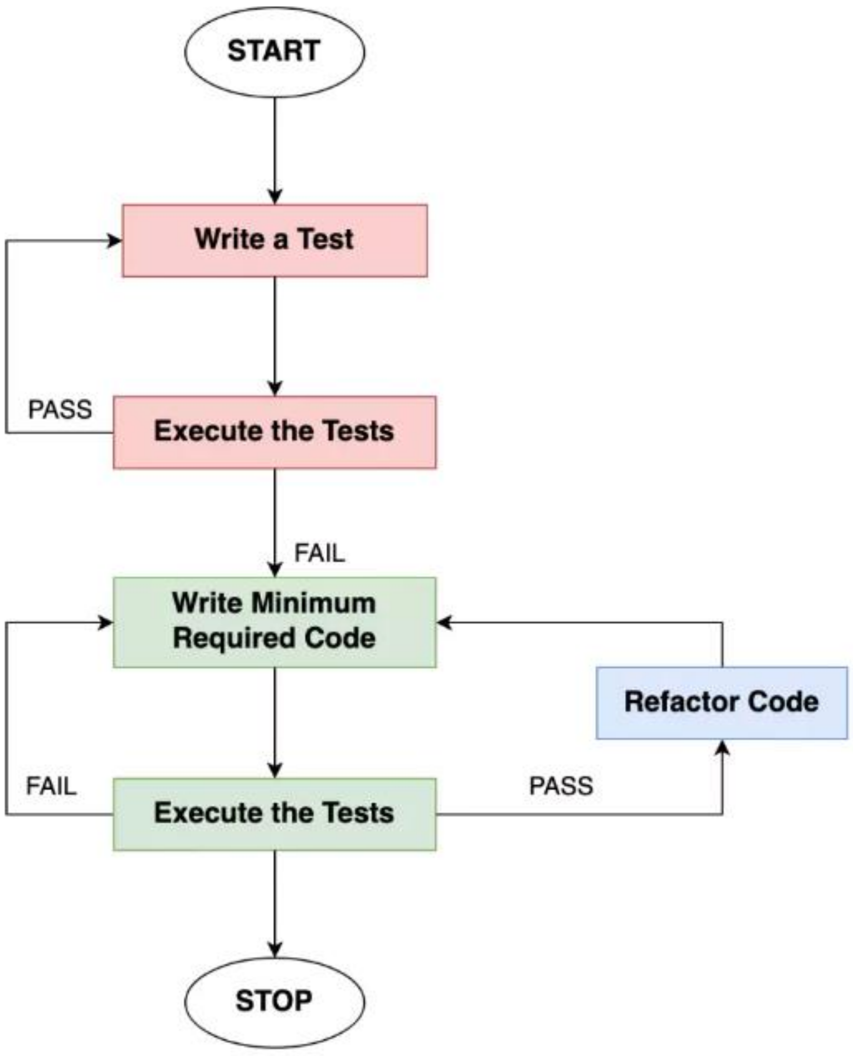
# Assignment-1

1. 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.

**Solution:**

Test-Driven Development (TDD**)** process:



**1.Write Tests First:**

- Begin by writing test cases for the functionality you want to implement.
- These tests should fail initially since the code doesn't exist yet.

**2.Write Minimal Code:**

- Write the minimum amount of code necessary to make the tests pass.
- Focus on functionality, not perfection.

**3.Run Tests:**

- Execute the tests to verify that the code behaves as expected.
- If any test fails, iterate on the code until all tests pass.

**4.Refactor:**

- Refactor the code to improve its design, readability, and maintainability.
- The tests act as a safety net during refactoring.

**5.Repeat:**

- Continue this cycle: write a new test, write minimal code, run tests, and refactor.
- TDD encourages incremental development and ensures reliable software.
-

**Benefits of TDD:**

**Bug Reduction:** Catch issues early, preventing them from reaching production.

**Reliability:** Code is thoroughly tested, leading to more robust applications.

Test-Driven Development is an iterative process, with developers continuously writing tests, implementing code, and refining both to produce high-quality software.

## Assignment-2

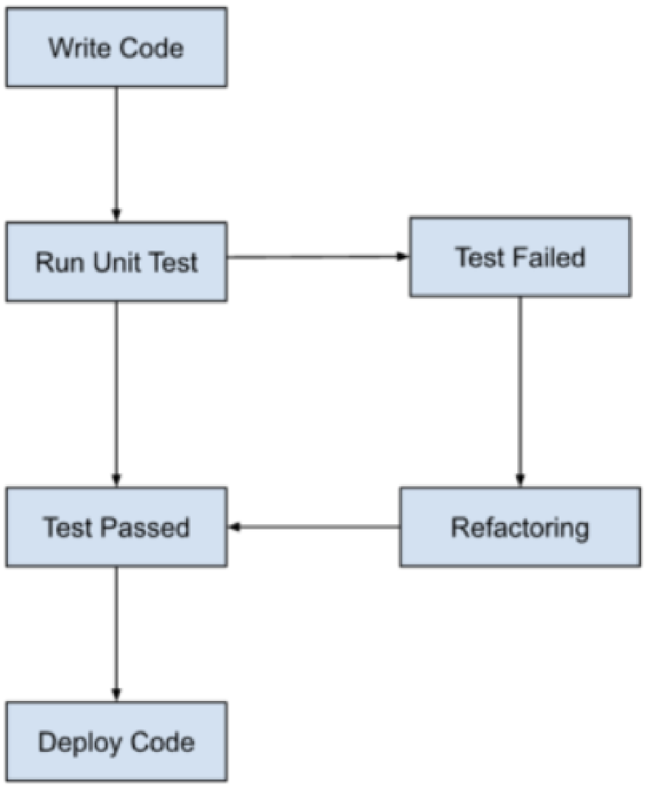
1. Produce a comparative infographic of TDD, BDD, and FDD methodologies. Illustrate their unique approaches, benefits, and

suitability for different software development contexts. Use visuals to enhance understanding.

**Solution:**

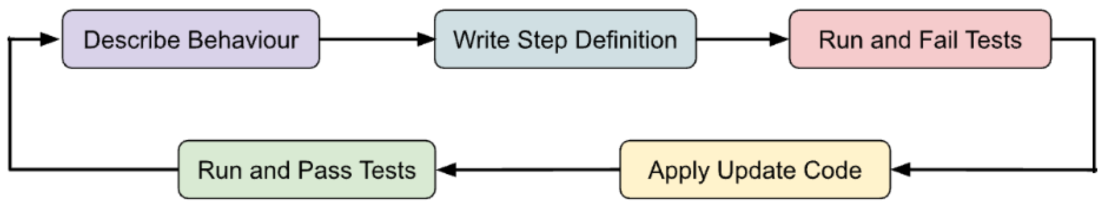
Comparative Infographic of TDD, BDD, and FDD Methodologies

1. **Test-Driven Development (TDD):**
   **Approach:** Begins with writing tests before writing code, focusing on small units of functionality.
   **Benefits:**
   Early bug detection.
   Improved code quality.
   Modular and maintainable codebase.
   **Suitability:** Best suited for projects where requirements are clear and stable, and for developers who prefer a bottom-up approach to development.

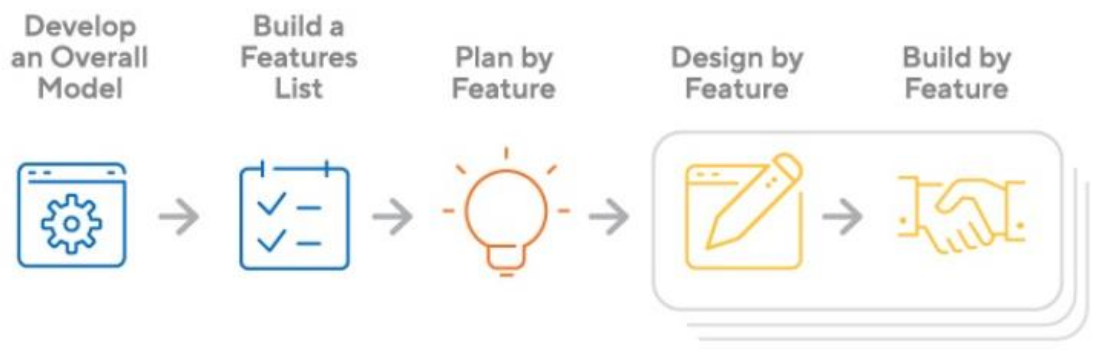


2. **Behavior-Driven Development (BDD):**
   **Approach:** Focuses on the behavior. of the system from the perspective of its stakeholders, using a common language.

**Benefits:**
Improved collaboration between developers, testers, and stakeholders.
Clearer understanding of requirements.
Encourages a user-centric approach.
**Suitability:** Ideal for projects with complex business logic, where communication and collaboration between team members and stakeholders are crucial.



3. **Feature-Driven Development (FDD):**
**Approach:** Emphasizes breaking down the project into manageable features, each with its own development cycle.
**Benefits:**
Scalable for large projects.
Clear focus on delivering tangible features.
Encourages frequent releases.
**Suitability:** Well-suited for large-scale projects with multiple teams, where features can be developed and delivered independently.



**Visual Representation:**

- **TDD:** Illustration of writing tests before code.

- **BDD:** Visual depiction of collaboration between developers, testers, and stakeholders.
- **FDD:** Representation of breaking down the project into features with their own development cycles.

Each methodology offers its unique approach to software development, catering to different project requirements and team dynamics. It's essential to choose the methodology that best aligns with the project's goals, team structure, and stakeholders' needs.