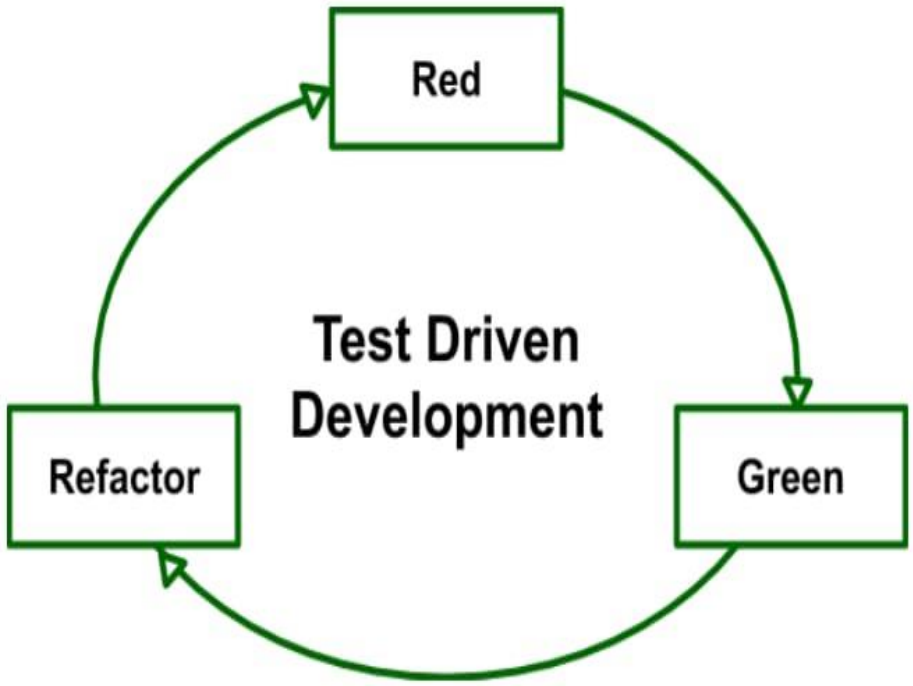
Assignment 1:

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

SOL:

Test-Driven Development (TDD) Infographic

1. **Write Test**:
    - Developers write automated tests based on the requirements before writing any code.
    - Start by writing a test that defines the behaviour or functionality you want to implement. This test should fail initially since you haven't written any code yet.
2. **Run Test**:
    - Tests are run immediately after writing them. They should fail initially as the corresponding code doesn't exist yet.
    - Execute the test to confirm that it fails as expected. This verifies that your test is correctly identifying the absence of the desired functionality.
3. **Write Code**:
    - Developers write code to make the failing tests pass. The code is written with the sole purpose of passing the tests.
    - Write the simplest code possible to make the failing test pass. The code should fulfil the requirements of the test and no more.
4. **Run All Tests Again**:
    - Once the code is written, all tests are run again. They should all pass now, indicating that the code meets the specified requirements.
    - Once all tests pass, refactor your code to improve its design, readability, and performance. Refactoring ensures that the code remains clean and maintainable.
5. **Refactor Code**:
    - After passing the tests, developers refactor the code to improve its design, readability, and performance without changing its behaviour.
    - Repeat the process for each new feature or functionality you want to add, starting with a failing test and iteratively writing code to pass the test.



Benefits of TDD:

- **Bug Reduction**:
- By writing tests before code, developers catch bugs early in the development process, reducing the likelihood of introducing defects.
- **Improved Software Reliability**:
- TDD ensures that the software behaves as expected by continually testing it against predefined criteria. This fosters reliability and confidence in the software's functionality.
- **Modular and Maintainable Code**:
- TDD encourages modular design and clean code practices since developers write small units of testable code. This leads to more maintainable and extensible software systems.
- **Faster Debugging**: TDD provides a suite of tests that can be run automatically, enabling quick identification and debugging of issues.
- **Design Improvement:** TDD encourages modular and loosely coupled code design, promoting better architecture and easier maintenance.
- **Living Documentation**:
- Tests serve as living documentation, providing insights into the software's behaviour and serving as a reference for future development and maintenance tasks.

To enhance the readability and engagement of the infographic, visual elements like icons representing each step of the TDD process, color coding to distinguish between different phases, and clear, concise text explanations can be included. Additionally, incorporating statistics or case studies demonstrating the effectiveness of TDD in real-world scenarios can further reinforce its benefits. This infographic provides a visual overview of the Test-Driven Development (TDD) process, highlighting its steps and benefits, such as bug reduction and improved software reliability. Visual elements like icons and color coding can enhance the infographic's readability and engagement.

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 give proper information.

Test-Driven Development (TDD)

Approach: Write tests before writing code. Red-Green-Refactor cycle.
Benefits:

- Early detection of defects.
- Encourages modular design.
- Provides living documentation.

Suitability:

- Works well for small to medium-sized projects.
- Projects where requirements are well-defined and unlikely to change frequently.

Behaviour-Driven Development (BDD)

Approach: Focuses on behaviour from the user's perspective using Given-When-Then scenarios.

Benefits:

- Promotes collaboration between developers, testers, and business stakeholders.
- Improves understanding of user requirements.
- Reduces ambiguity in requirements.

Suitability:

- Particularly useful for projects with complex business logic.
- Projects where stakeholders have diverse backgrounds and need a common language to discuss requirements.

Feature-Driven Development (FDD)

Approach: Iterative and incremental development based on features.
Benefits:

- Emphasizes domain object modelling.
- Allows for rapid development and delivery of features.
- Encourages regular communication and collaboration among team members.

Suitability:

- Suitable for large projects with multiple teams.
- Projects where feature delivery needs to be prioritized. **Comparison Table :**

Aspect	TDD	BDD	FDD
Focus	Testing	Behaviour	Features
Approach	Write tests first	Given-When-Then scenarios	Iterative feature development
Collaboration	Minimal	Extensive	Moderate
Documentation	Living documentation	Clear behaviour specs	Feature-centric
Project Size Suitability	Small to Medium	Any size	Large

This infographic provides a visual comparison of TDD, BDD, and FDD methodologies, highlighting their unique approaches, benefits, and suitability for different software development contexts. Visual elements such as icons, charts, and color coding can be added.