**Assignment 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.

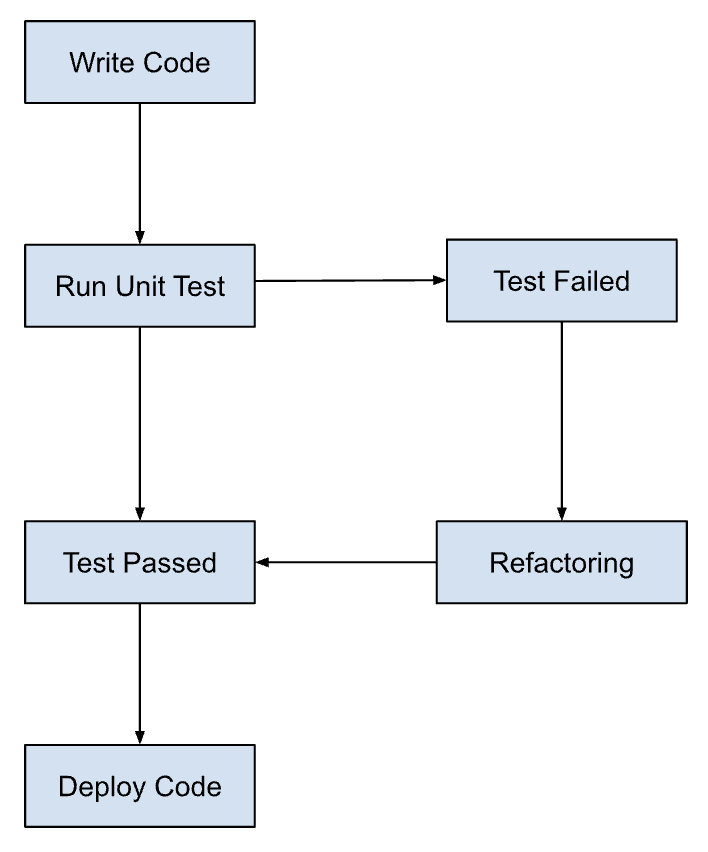
Test Driven Development (TDD) is a software development practice that focuses on creating unit test cases before developing the actual code. It is an iterative approach combining programming, unit test creation, and refactoring.

* The TDD approach originates from the Agile manifesto principles and Extreme programming.
* As the name suggests, the test process drives software development.
* Moreover, it’s a structuring practice that enables developers and testers to obtain optimized code that proves resilient in the long term.
* In TDD, developers create small test cases for every feature based on their initial understanding. The primary intention of this technique is to modify or write new code only if the tests fail. This prevents duplication of test scripts.

Three Phases of Test Driven Development

1. **Create precise tests:** Developers need to create exact Unit test to verify the functionality of specific features. They must ensure that the test compiles so that it can execute. In most cases, the test is bound to fail. This is a meaningful failure as developers create compact tests based on their assumptions of how the feature will behave.
2. **Correcting the Code:** Once a test fails, developers must make the minimal changes required to update the code to run successfully when re-executed.
3. **Refactor the Code:** Once the test runs successfully, check for redundancy or any possible code optimizations to enhance overall performance. Ensure that refactoring does not affect the external behavior of the program.

The image below represents a high-level TDD approach toward development:



Advantage Of DDT: -

1. Fosters the creation of optimized code.
2. It helps developers better analyze and understand client requirements and request clarity when not adequately defined.
3. Adding and testing new functionalities become much easier in the latter stages of development.
4. Test coverage under TDD is much higher compared to conventional development models. The TDD focuses on creating tests for each functionality right from the beginning.
5. It enhances the productivity of the developer and leads to the development of a codebase that is flexible and easy to maintain.

**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.

**ABOUT TDD**

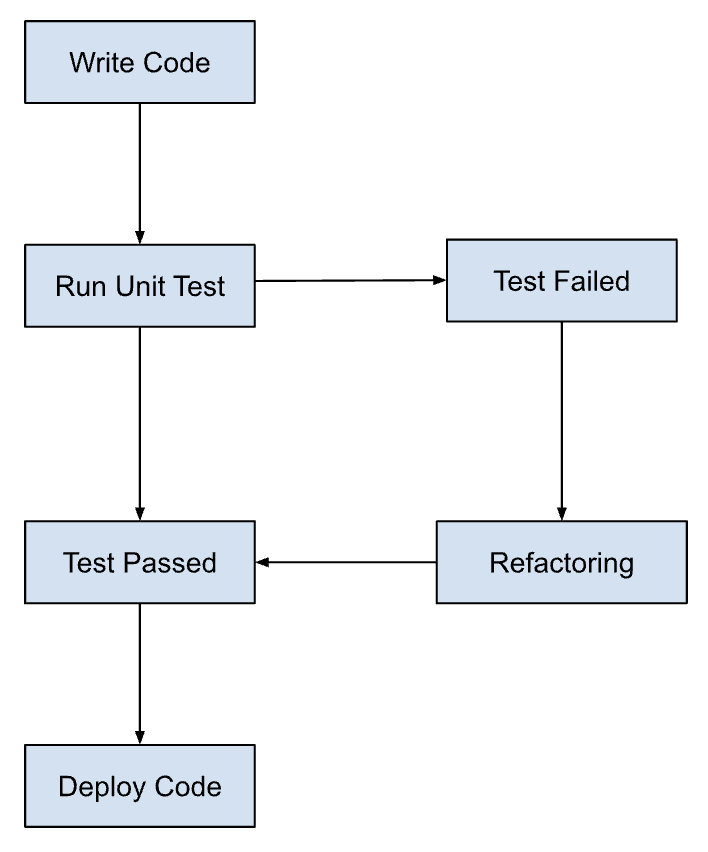
Test Driven Development (TDD) is a software development practice that focuses on creating unit test cases before developing the actual code. It is an iterative approach combining programming, unit test creation, and refactoring.

The main intention of this technique is to modify or write a fresh code only when the test fails. Hence it results in lesser duplication of test scripts. This technique is prevalent mainly in agile development ecosystems. In a TDD approach, automated test scripts are written before functional pieces of code. The TDD methodology involves the following steps:

1. A developer writes an Automated Test Case based on the requirements specified in the documents.
2. These tests are executed, and in some cases, they fail as they are developed before the development of an actual feature.
3. The development team then re-factors the code for the test to pass successfully.
4. TDD can be done by a single developer while writing both tests and application code side by side to complete a feature.

#### **Benefits of TDD**

* Reduces the amount of time required for rework
* Explores bugs or errors very quickly
* Faster feedback
* Encourages the development of cleaner and better designs
* Enhances the productivity of the programmer
* Allows any team member to start working on the code without a specific team member. This encourages knowledge-sharing and collaboration.
* It gives the programmer confidence to change an application’s large architecture quickly.
* Results in the creation of extensive code that is flexible and easy to maintain



**ABOUT BDD**

BDD is a testing approach derived from the (TDD) methodology. In BDD, tests are mainly based on systems behavior. This approach defines various ways to develop a feature based on its behavior. In most cases, the***Given-When-Then*** approach is used for writing test cases.

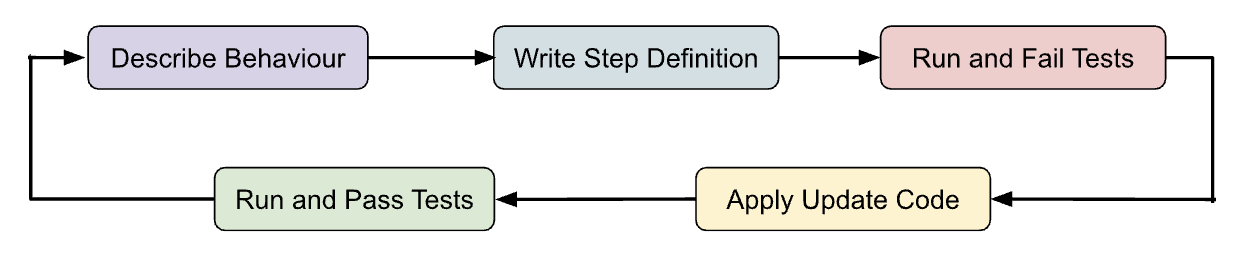
**In Most of The Cases in BDD**

* **Given** the user has entered valid login credentials
* **When** a user clicks on the login button
* **Then** display the successful validation message

#### **Key benefits of BDD**

* Helps reach a wider audience through the usage of non-technical language
* Focuses on how the system should behave from the customer’s and the developer’s perspective
* BDD is a cost-effective technique
* Reduces efforts needed to verify any post-deployment defects

**Diagram of BDD operations**:



**ABOUT FDD**

FDD, or Feature-Driven Development, is an iterative and incremental Agile methodology that focuses on delivering working software in short cycles, with a strong emphasis on features.

**Core Principles of FDD:**

* **Feature-centric:** Projects are broken down into manageable features, which become the building blocks for development.
* **Iterative development:** Work is delivered in short, time-boxed iterations, typically 2-week cycles.
* **Strong focus on planning:** FDD emphasizes upfront planning activities like domain model creation and feature prioritization.
* **Continuous inspection:** Regular inspections are conducted throughout the process to ensure quality and adherence to plans.
* **Collaboration:** While FDD is developer-driven, it encourages collaboration with customers and other stakeholders.

**FDD Process:**

1. **Feature Decomposition:** The project team collaborates to create a comprehensive model of the system, identifying features and their sub-features.
2. **Planning by Feature:** Features are prioritized, and effort is estimated for each. The development work is then divided into short iterations with specific feature sets targeted for completion.
3. **Design by Feature:** Within each iteration, detailed design documents are created for the features on the current iteration plan.
4. **Develop by Feature:** Developers implement the features according to the design documents, writing unit tests and integrating code with the main codebase.
5. **Inspect by Feature:** Completed features undergo rigorous inspections by developers, customers, and other stakeholders to ensure they meet quality standards and align with expectations.

**Benefits of FDD:**

* **Reduced project risk:** By focusing on features and iterations, FDD helps manage complexity and identify issues early.
* **Increased predictability:** The iterative nature of FDD provides better visibility into project progress and allows for course correction if needed.
* **Improved project control:** Through comprehensive planning and inspection, FDD empowers teams to stay on track and deliver features within timeframes.

**Suitability of FDD:**

* 1. **Strong upfront planning is crucial.**
  2. **Feature prioritization and iteration management are critical.**
  3. **Close collaboration between developers and stakeholders is desired.**