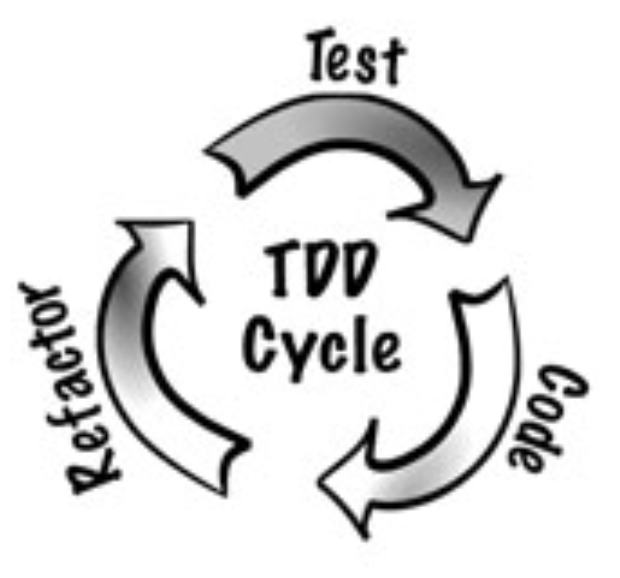
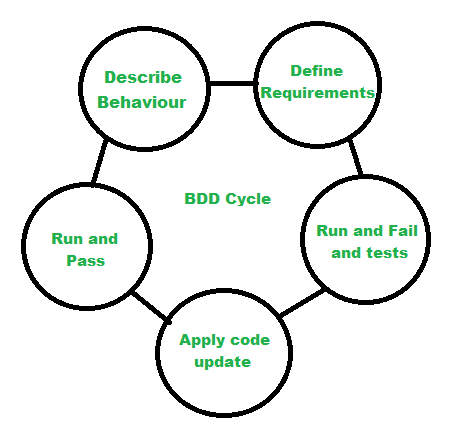
TDD,BDD,FDD Methodologies

Overview : **Test-Driven Development (TDD)**, **Behaviour-Driven Development (BDD)**, and **Feature-Driven Development (FDD)** methodologies, along with their distinctive approaches, benefits, and contexts.

1. **Test-Driven Development (TDD)**:
   * **Approach**: In TDD, developers write tests before writing the actual code. The process follows a **red-green-refactor** cycle:
     + **Red**: Write a failing test for the desired functionality.
     + **Green**: Write the minimum code needed to make the test pass.
     + **Refactor**: Optimize and clean up the code while ensuring tests still pass.
   * **Benefits**:
     + Improved code quality and fewer bugs due to thorough testing.
     + Enhanced code design, as writing tests first forces careful consideration of the interface and design.
     + Easier maintenance and refactoring, thanks to the safety net provided by tests.
     + [Faster development velocity in the long run, as early bug detection prevents issues downstream](https://fullscale.io/blog/tdd-vs-bdd/)[1](https://fullscale.io/blog/tdd-vs-bdd/).
   * **Suitability**: TDD works well for small, co-located developer-centric teams.



1. **Behaviour-Driven Development (BDD)**:
   * **Approach**: BDD extends TDD by focusing on the system’s behaviour from various stakeholders’ perspectives. Desired behaviours are expressed through examples and scenarios using a domain-specific language (e.g., Gherkin). These scenarios serve as executable specifications validated by automated tests.
   * **Benefits**:
     + Strong alignment between business and technical teams.
     + Faster feedback and better visibility.
     + Higher quality and fewer defects due to upfront clarification of requirements.
     + [Efficient test coverage directly tied to documented behaviors](https://fullscale.io/blog/tdd-vs-bdd/)[1](https://fullscale.io/blog/tdd-vs-bdd/).
   * **Suitability**: BDD is ideal for projects where communication, collaboration, and understanding of requirements are critical.



1. **Feature-Driven Development (FDD)**:
   * **Approach**: FDD focuses on building features incrementally. It involves creating a high-level feature list, breaking features into smaller tasks, and assigning them to developers. Each feature is developed iteratively.
   * **Benefits**:
     + Clear feature-based planning and tracking.
     + Efficient use of resources by prioritizing features.
     + Improved collaboration among team members.
     + Emphasis on delivering valuable features to users.
   * [Suitability: FDD suits larger projects with well-defined features and a need for structured development2](https://medium.com/@mohsenny/bringing-theory-to-practice-a-real-world-guide-to-tdd-bdd-and-atdd-with-cypress-f4593aec40a8).
   * 