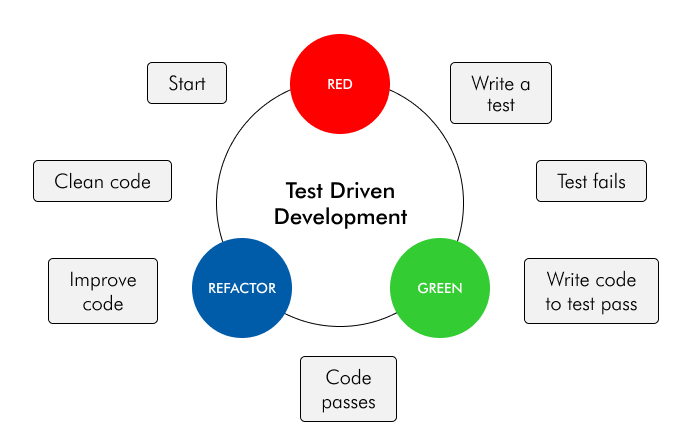
**Test-Driven Development (TDD)** flips the traditional coding script. Instead of writing code first and testing later, TDD prioritizes tests, creating a reliable development cycle.



**The TDD Cycle:**

1. **Write a Failing Test:**
   * Define the desired behavior of your code with a test case.
   * Initially, this test will fail because the code you'll be writing doesn't exist yet.
2. **Write Minimal Code to Pass the Test:**
   * Focus on implementing just enough code to make the failing test pass.
   * Don't worry about elegance or efficiency yet.
3. **Refactor the Code:**
   * Now that the test is green (passing), clean up and improve your code's structure and readability.
   * Refactoring ensures maintainability and prevents introducing regressions (bugs caused by changes).
4. **Repeat:**
   * Write a new failing test for the next feature or functionality.
   * Rinse and repeat, building a robust codebase with a safety net of tests.

**Benefits of TDD:**

* **Reduced Bugs:** Early and frequent testing catches errors before they become bigger problems.
* **Improved Design:** Focus on meeting test criteria often leads to cleaner and more focused code.
* **Enhanced Maintainability:** Tests serve as living documentation, making it easier to understand and modify code later.
* **Increased Confidence:** A strong test suite provides a safety net, giving developers confidence in their code's reliability.

**TDD is a powerful approach for:**

* Individual developer tasks.
* Projects with well-defined requirements.

**Remember:** TDD is an iterative process. Embrace the cycle, write failing tests, build minimal code, refactor, and repeat for a solid foundation of reliable software.