**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.

**TDD FLOW CHART**

WRITE A TEST

RUN THE TEST

WRITE CODE

RUN ALL TESTS

REFACTOR CODE

REPEAT

**Infographic Title: Test-Driven Development (TDD) Process**

-----------------------------------------------------------

**1. Introduction to TDD**

Definition: Test-Driven Development (TDD) is a software development approach where tests are written before the actual code.

Goal: Ensure code quality and reliability by validating functionality through automated tests.

**2. TDD Cycle**

**Write a Test**

**Description**: Write a test for the new functionality. The test should fail initially because the functionality isn’t implemented yet.

**Icon**: Pen writing on a document with a red cross symbol.

**Run the Test**

**Description**: Run the test to ensure it fails. This step verifies that the test is detecting the absence of the feature.

**Icon**: Computer screen showing a failing test (red cross).

**Write Code**

**Description**: Write the minimal amount of code necessary to pass the test.

**Icon**: Code editor with some lines of code.

**Run All Tests**

**Description**: Run all tests to ensure the new code passes the test and doesn’t break existing functionality.

**Icon**: Computer screen showing passing tests (green checkmarks).

**Refactor Code**

**Description**: Clean up the code while ensuring all tests still pass. Improve code structure and remove any duplication.

**Icon**: Code editor with cleaner, more organized code.

**Repeat**

**Description**: Repeat the cycle for new functionality.

**Icon**: Circular arrow indicating iteration.

**3. Benefits of TDD**

**Bug Reduction**

**Description**: Catch bugs early by writing tests first. This leads to fewer bugs in production.

**Icon**: Bug with a prohibition sign over it.

**Improved Code Quality**

**Description**: Encourages clean, maintainable code through refactoring.

**Icon**: Sparkling clean code document.

**Reliable Software**

**Description**: Automated tests ensure continuous verification of code functionality.

**Icon**: Shield symbolizing reliability.

**Faster Development**

**Description**: Less time spent debugging and fixing issues, leading to quicker development cycles.

**Icon**: Stopwatch with a forward arrow.

**4. TDD Best Practices**

**Start Small**: Begin with simple tests and gradually add complexity.

**Stay Consistent**: Maintain the TDD cycle consistently throughout development.

**Write Meaningful Tests**: Ensure tests cover various edge cases and scenarios.

**Refactor Regularly**: Keep the codebase clean and efficient by regularly refactoring.

**5. TDD Advantages**

* **Improved code quality:** TDD ensures that code is clean and optimized, and that each stage of the code works as it progresses.
* **Fewer bugs:** TDD helps prevent bugs and errors, which means developers spend less time fixing them.
* **Faster development:** TDD allows developers to make changes quickly without worrying about breaking existing function, cause test can be return after each code change.
* **Easier debugging**: TDD allows for faster debugging and minor debugging.