**Assignment 4**

**Test-Driven Development (TDD) Process Info graphic**

Title: Test-Driven Development (TDD) Process

* Introduction

Brief explanation of TDD and its significance in software development.

TDD Process Overview

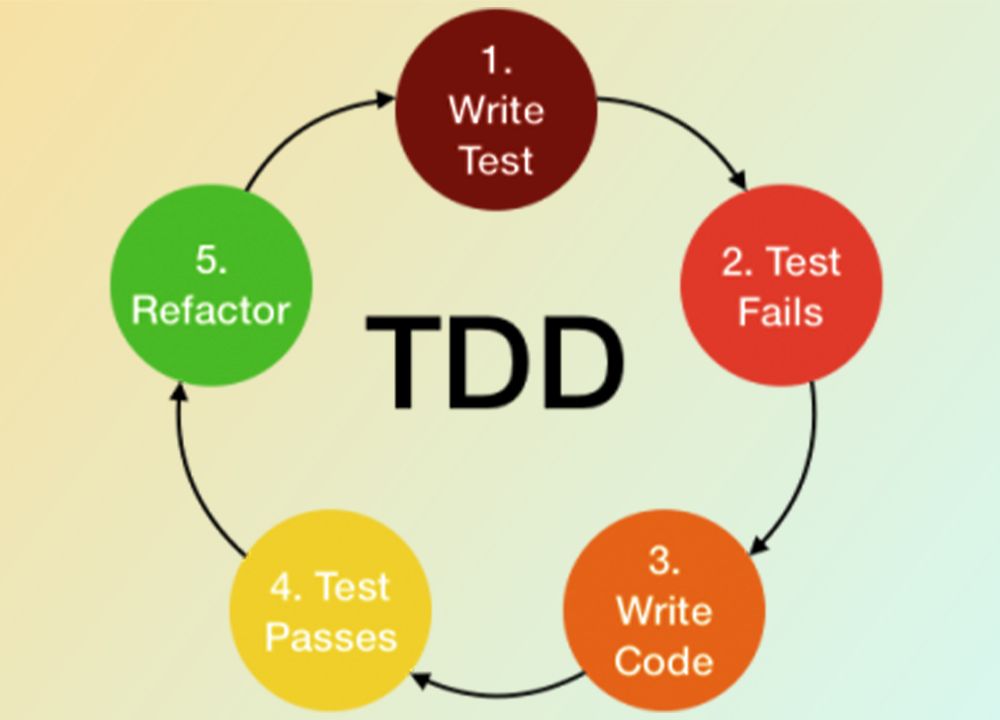
Test-driven development overturns conventional growth and testing. So, instead of first writing your code and then summarily fitting a test to verify the block of malware you just wrote, test-driven creation requires that you first write the test and then apply code modifications before the test you have already written passes your code.

You first write the unit test in TDD, watch it fail, and then enforce code modifications before the test passes. Sounds backwards, huh? But when you use this  technique the code you generate is safer and less susceptible to breaking in the long run.

A unit test is essentially a test that, like an algorithm, for example, covers a small portion of logic. Checks on units should be probabilistic. I mean that unit tests should never have adverse effects, such as calls to external API’s that provide unpredictable or changing data, when I say “deterministic.” Instead of data that might alter over time, you’d use mock data.

Visual depiction of the three main steps: Write Test, Write Code, Refactor.

Arrows showing the iterative nature of TDD.



Step 1: Write Test

Visual representation of writing test cases before writing any code.

Icon or illustration of a test case being written.

Step 2: Write Code

Visual depiction of writing code to make the test cases pass.

Icon or illustration of a developer writing code.

Step 3: Refactor

Visual representation of refactoring code to improve its structure without changing its behavior.

Icon or illustration of code being optimized.

## **What are the Tools Used for Test Driven Development**

The following are some popular and most used frameworks/tools for unit testing that support the TDD approach:

1. ****CsUnit****: A research tool for an open-source unit that provides a TDD unit test system for. Net projects
2. ****Dorset****: For Python, a very basic, easy-to-learn the unit testing system
3. ****JUnit****: A test system for a Java TDD unit.
4. ****NUnit****: This one is used for. Net projects once again
5. ****PHPUnit****: For PHP projects, this one is used
6. ****PyUnit****: For Python, a standard unit testing application
7. ****TestNG****: A Java testing framework which overrides JUnit’s limitations.
8. ****RSpec****: A system for projects with Ruby.

* Benefits of TDD

1. First of all, writing the tests allows you to understand what you want from the code.
2. You’ll get short reviews.
3. A comprehensive specification is created by TDD.
4. TDD decreases the time spent reworking.
5. In the debugger, you waste less time.
6. You can easily recognize mistakes and issues.
7. TDD informs you whether your last update (or refactoring) has broken the working code.
8. TDD makes it possible for the design to develop and respond to your evolving understanding of the problem.
9. The extreme simplification of the code is forced by TDD. In answer to the requirements of the tests, you can only write code.
10. You are compelled to write tiny classes based on one issue.
11. TDD produces SOLID applications.
12. A clean GUI supports TDD.
13. TDD produces code that is easily extensible, scalable, and maintainable.
14. The subsequent unit testing are easy and serve as code validation.
15. Since TDD use cases are written as tests, the tests can be used by other programmers as usage samples of how the code is supposed to operate.
16. The marketing time for production is shorter.
17. The productivity of the program is improved.
18. Costs for construction are being cut.
19. Efficiency is strengthened.
20. It eliminates bugs.

* **Conclusion**: Summary of the TDD process and its benefits.

Call to action or further resources for learning TDD.