# Introduction

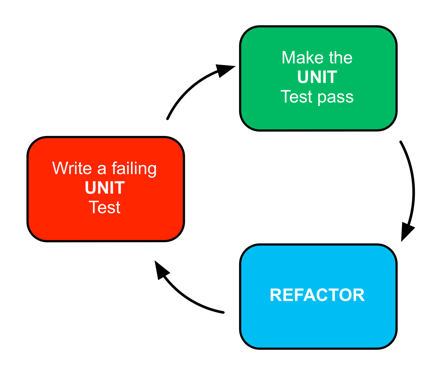
**Test Driven Development** (TDD) is a powerful software development practice for combining software design, unit testing, and coding in a continuous process to increase quality, maintainability, and productivity. Using TDD, developers build software in short development cycles following a workflow pattern called Red - Green - Refactor:

Figure 1Red-Green-Refactor

1. **RED** - The developer writes a failing automated test case that defines a new function or improvement
2. **GREEN**) - Write minimal code to pass that test
3. **REFACTOR** - Refactors the new code or existing code to acceptable standards

The developer repeats this process many times until the unit is complete and fully tested.

During this workshop, you will use code 'katas' and pair-programming to practice the techniques. You will work on three code labs specifically crafted to encourage object-oriented thinking and design through test-first and merciless refactoring. Subsequent pairing will include working through real-world client feature development scenarios and how Test Driven Development could be applied in specific situations.

You can use any IDE and language that you have available to you, as long as the micro-testing framework (e.g., JUnit, NUnit, Xunit, rSpec) is also available. You will explore the TDD discipline at your own pace, in a low-stress environment, with frequent opportunity to get personalized assistance from the instructor and others. You will need to supply your own workstation.

Participants should have strong programming skills and be familiar with an object-oriented language and programming techniques. They should bring a laptop installed with their favorite programming language and IDE and come prepared to write code. You will need to install your chosen test framework prior to the tutorial.

## Audience

Developers who are interested in introducing TDD into their team’s tool-box of powerful software engineering practices. Perhaps they have tried to encourage the TDD discipline already but want more first-person experience in order to address concerns within the organization. Participants should be proficient in a modern object-oriented language.

## Schedule

The emphasis will be on learning by doing. There will be some preliminary demonstration and discussion, but we will forgo lengthy debate in favor of spending enough time on the labs. Each participant will receive personal attention from our team.

**Day 1**

* (60 min) Morning session presentation
  + Introduction
  + The Case for TDD
  + Types of Testing
  + Example
* (120 min) Pairing session. 30-minute sessions executing any of several code katas.
* (60 min) Afternoon session presentation
  + Design for testability
  + How to get started on my project
  + TDD in legacy applications or mature applications without unit tests
* (120 min) Pairing session, Legacy refactor.

**Day 2**

Pair with individual developers or teams to work through existing examples where TDD could be applied. Talk through design considerations and design principles that make code easier to test. Work through refactoring scenarios that ensure new code is testable.

## Learning Objectives

* Learn how to use one of the xUnit family of testing frameworks (for this workshop, both JUnit and NUnit will be available).
* Learn test-first as a thinking process.
* Learn the advantages of automating micro-tests.
* Learn how TDD has been known to reduce defect rates by 80%.
* Learn merciless refactoring as a code design tool.
* Learn how TDD enables Emergent Design and can improve architectural-change cycle times.
* Learn how TDD can be applied in a project no matter the pre-existing work that has been done.