# Lab 2: Test Driven Development and Behavior Driven Development

## Objective

To understand and implement the principles of **Test-Driven Development (TDD)** and **Behavior Driven Development (BDD)** by writing tests before writing functional code and ensuring behavior-focused development through examples and scenarios.

## Tools & Technologies Used

* Programming Language: e.g., Python
* TDD Framework: e.g., PyTest, JUnit
* BDD Framework: e.g., behave (Python), Cucumber
* IDE: e.g., VS Code, IntelliJ
* Version Control: Git

## Introduction

**Test Driven Development (TDD)** is an Agile software development practice where test cases are written **before** the actual code. The TDD cycle typically follows:

1. **Red** – Write a failing test.
2. **Green** – Write the minimum code to pass the test.
3. **Refactor** – Improve the code while keeping tests green.

**Behavior Driven Development (BDD)** builds on TDD by focusing on the **behavior of the application from the end-user’s perspective**, using **plain language scenarios** (e.g., Given-When-Then).

## Implementation

* **Test case 1: Addition**  
  test('Add two numbers', () {

expect(add(2, 3), equals(5));

});

* **Code to pass the test**int add(int a, int b) => a + b;

**Note:** The full implementation and test cases are provided under the **Lab2/implementation** folder of this repository.

## Observations

* Writing tests before code helps define clear objectives.
* BDD makes communication easier between developers, testers, and stakeholders.
* Both methods promote clean, reliable, and testable code.

**A screen shot of a computer screen

AI-generated content may be incorrect.**

## Conclusion

In this lab, we developed a basic calculator program and learned Agile testing practices like TDD and BDD.