# TDD Report: Calculator Operations

Name: Muhammad Zidan Fatonie  
NIM: 2702358235  
  
Introduction

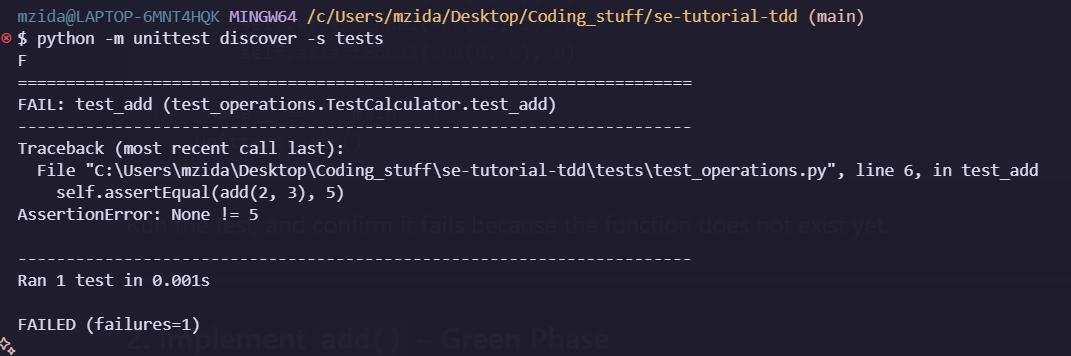
This report details the Test-Driven Development (TDD) cycle applied to a simple Python calculator module implementing four basic operations: add, subtract, multiply, and divide. Each function was developed following the classic Red‑Green‑Refactor process and is illustrated with the actual test-run screenshots you provided.

1. Red: Write a unit test that fails (because the function is not implemented or incorrect).
2. Green: Implement the minimal code needed to make the test pass.
3. Refactor: Clean up code (remove redundancy, improve clarity) while ensuring all tests still pass.

## 1. add(a, b)

### 1.1 Red

* Test written (test\_add in tests/test\_operations.py):  
  self.assertEqual(add(2, 3), 5)
* Screenshot of failure:



### 1.2 Green

Implementation added in calculator/operations.py:  
def add(a, b):

* return a + b
* Test re-run: addition and add tests pass (3 tests).



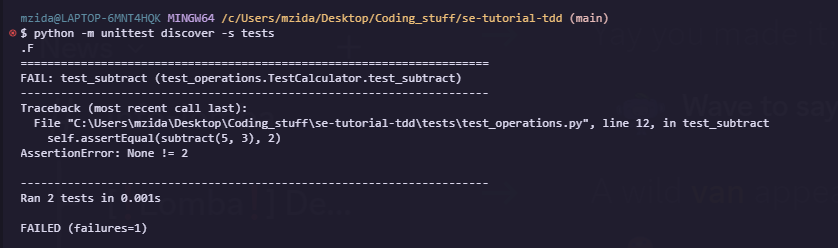
### 1.3 Refactor

* Removed the stray pass statement.
* Verified PEP8 compliance and no change in behavior; tests remain green.

## 2. subtract(a, b)

### 2.1 Red

* Test written (test\_subtract):  
  self.assertEqual(subtract(5, 3), 2)
* Screenshot of failure:



### 2.2 Green

Implementation in calculator/operations.py:  
def subtract(a, b):

* return a - b
* Test re-run: subtraction and add tests pass (3 tests).



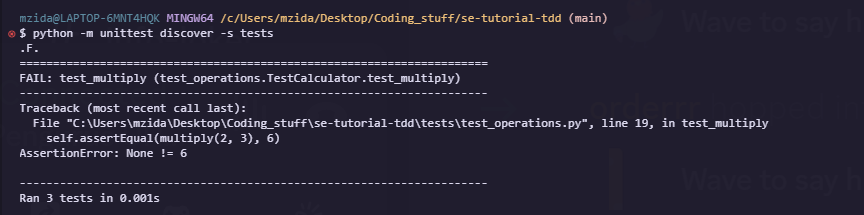
### 2.3 Refactor

* Removed redundant pass.
* Cleaned up comments.

## 3. multiply(a, b)

### 3.1 Red

* Test written (test\_multiply):  
  self.assertEqual(multiply(2, 3), 6)
* Screenshot of failure:



### 3.2 Green

Implementation:  
def multiply(a, b):

* return a \* b
* Test re-run: multiplication, subtract, and add tests pass (4 tests).



### 3.3 Refactor

* Deleted the trailing pass.
* Checked formatting and consistency.

## 4. divide(a, b)

### 4.1 Red (First iteration)

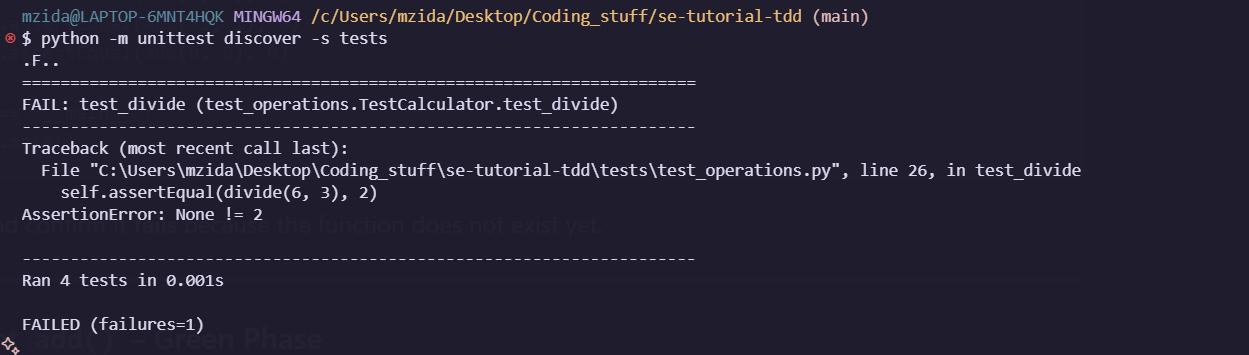
Test written (test\_divide):  
self.assertEqual(divide(6, 3), 2)

with self.assertRaises(ValueError):

* divide(1, 0)
* Screenshot of Python raising ZeroDivisionError (unexpected):



* Screenshot of missing return for divide(6,3) before implementing logic:



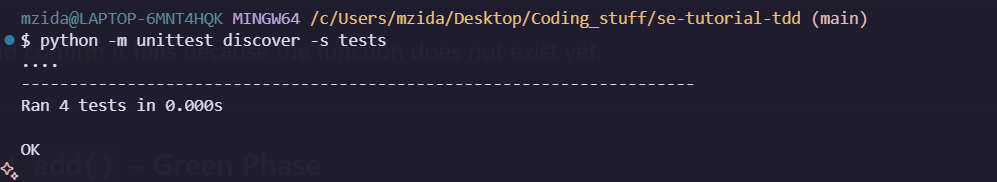
### 4.2 Green

Implementation updated in calculator/operations.py:  
def divide(a, b):

if b == 0:

raise ValueError("Cannot divide by zero")

* return a / b
* Test re-run: all four assertions pass.



### 4.3 Refactor

* Removed pass.
* Added docstring explaining exception behavior.

## Conclusion

All four operations now satisfy their unit tests via the TDD process:

| Function | Status |
| --- | --- |
| add | ✔️ Returns correct sum (see add success) |
| subtract | ✔️ Returns correct difference (see subtract success) |
| multiply | ✔️ Returns correct product (see multiply success) |
| divide | ✔️ Returns correct quotient and handles zero divide |

By strictly following Red‑Green‑Refactor and referring to actual test-run evidence, we ensured each feature was both tested and implemented correctly before moving to the next. Future refactoring or extensions remain safely covered by this robust suite of unit tests.