## What is a unit test?

A 'unit test' tests very specific, much focused, single aspect of functionality.

### Characteristics of a unit test:
- Focused, making only one claim at a time.
- Fast to run.
- Independent of
  - Each other.
  - The environment.
  - The order in which it is run.
- Same high quality as the production code.
- Automatic.

### When it is not a unit test?

A test that does not operate in isolation is not a unit test. It is safe to assume that a test that connects to the network or a database or a real file is not a unit test. Use mock objects or stubs to test in isolation.

### Being test-driven
- Being test-driven is not about testing, but about evolving the design to meet the requirements.
- Each unit test corresponds to a single requirement that the code must satisfy.

## Test-Driven Cycle:

- Prepare a list of test cases.
- Follow the TDD Rhythm:
  - Pick a test to implement.
  - Write a failing test.
  - Quickly, make the test green.
  - Refactor to eliminate duplication.

### How do I pick the next test to implement?

Pick a test that
- You are confident that you can implement.
- You know is the next logical step and will give you the most learning.

### Before writing the next test, ask:

- Is there any duplication? Do I need to refactor first?
- What are the correct inputs? How am I going to check for the effect of these inputs?
- Where does the operation belong?

### What should I test?

Test everything that could possibly break.

### What not to test?

- Do not test getters and setters.
- It is almost impossible to unit test the GUI, so keep the GUI as thin as possible.

## Should I test private methods?

Testing the public interface of a class should be sufficient. Public interface should invoke the private methods.

## Things to always do:

- Write test first.
- Maintain a to-do list.
- Write new code, only if, there is a failing automated unit test.
- Run all the tests all the time and not just a single isolated test.
- Keep the test code of the highest quality without any code smells.

## Things to never do:

- Write multiple failing tests at a time.
- Write new test without eliminating duplication.
- Commit code with failing tests.

## What is a to-do list?

- To-do list contains
  - Examples of test cases.
  - Any refactoring that needs to be done.
- To-do list should be kept current. At any time, it should tell the following:
  - Current test or refactoring being carried out.
  - Pending tests.
  - Pending refactorings.