The TDD Rhythm

The TDD Rhythm:

1. Quickly add a test
2. Run all tests and see the new one fail
3. Make a little change
4. Run all tests and see them all succeed
5. Refactor to remove duplication

Essential TDD Principles

TDD Principle: **Test First**

When should you write your tests? Before you write the code that is to be tested.

TDD Principle: **Test List**

What should you test? Before you begin, write a list of all the tests you know you will have to write. Add to it as you find new potential tests.

TDD Principle: **One Step Test**

Which test should you pick next from the test list? Pick a test that will teach you something and that you are confident you can implement.

TDD Principle: **Isolated Test**

How should the running of tests affect one another? Not at all.

TDD Principle: **Evident Tests**

How do we avoid writing defective tests? By keeping the testing code evident, readable, and as simple as possible.

TDD Principle: **Fake It ('Til You Make It)**

What is your first implementation once you have a broken test? Return a constant. Once you have your tests running, gradually transform it.

TDD Principle: **Triangulation**

How do you most conservatively drive abstraction with tests? Abstract only when you have two or more examples.

TDD Principle: **Assert First**

When should you write the asserts? Try writing them first.

TDD Principle: **Break**

What do you do when you feel tired or stuck? Take a break.

TDD Principle: **Evident Data**

How do you represent the intent of the data? Include expected and actual results in the test itself, and make their relationship apparent. You are writing tests for the reader, not just for the computer.

TDD Principle: **Obvious Implementation**

How do you implement simple operations? Just implement them.

TDD Principle: **Representative Data**

What data do you use for your tests? Select a small set of data where each element represents a conceptual aspect or a special computational processing.

TDD Principle: **Automated Test**

How do you test your software? Write an automated test.

TDD Principle: **Test Data**

What data do you use for test-first tests? Use data that makes the tests easy to read and follow. If there is a difference in the data, then it should be meaningful. If there isn't a conceptual difference between 1 and 2, use 1.

TDD Principle: **Child Test**

How do you get a test case running that turns out to be too big? Write a smaller test case that represents the broken part of the bigger test case. Get the smaller test case running. Reintroduce the larger test case.

TDD Principle: **Do Over**

What do you do when you are feeling lost? Throw away the code and start over.

TDD Principle: **Regression Test**

What's the first thing you do when a defect is reported? Write the smallest possible test that fails and that, once run, will be repaired.