1. First write tests, then work with the method under test to make it pass that test, and pass all previous tests.

2. I can see how writing tests before you write the method can be useful, but personally, I think I'd much rather write out all the test cases, then write out the entire method, then use the tests to fix whatever's wrong rather than use the tests to write it from scratch. It seems to me you'd be able to logically grind out a reasonably good method just by knowing what you want the method to do, making the method work for each individual test case seems like a waste of time.

3. TDD is a terrible method, it doesn't make sense to not logically think out a method that will be able to pass future test cases, in the instructional video we're even told to think ahead, yet it's really hard to think of a method that will pass a couple future test cases but not ALL future test cases, couldn't really use TDD on the second one, because the only way to pass test case two other than literally hard coding in the return was to implement a method... which isn't following TDD. What's the problem with thinking through a method, AFTER writing many test cases, then testing the method you wrote? In the end my test cases were wrong while my method was right...