Principles and Practices

1. Start testing first
   - 1.1 Test the ideas before any design or code implementation starts.
   - 1.2 Tests are about feedback loops and giving us a way to correct early, so add testing from the start.
2. Mock external dependencies but remove the mocks if you can
   - 2.1 For example, focus on the UI logic while you mock out the code that provides the data to display. This reduces complexity while developing (and testing)
   - 2.2 Beware that excessive mocking can introduce a brittle design.
3. Start with TDD as early as possible because it is much more difficult to add tests later.
4. Run tests with build script from day 0
5. Test the right code
   - 5.1 To test all your code is sometimes not the right goal, especially when using application frameworks.
   - 5.2 At the very least test the business and domain logic code.
   - 5.3 You typically do not need to test basic language features, like getters and setters.
   - 5.4 Test the public interfaces (i.e. public methods and constructors)
   - 5.5 Don't test generated code and SDKs.
6. Tests can indicate design flaws
   - 6.1 If small changes in code breaks lots of tests, it can point to bad design. Refactor that code, it might be trying to do too much.
7. Cultivate testing a mindset in team
   - 7.1 It saves time if you do it from the start
   - 7.2 Thinking that tests slow you down is also a false sense of productivity. You will always pay more with manual testing later.

Testing is not an optional add-on, it is an integral part of how we create software.