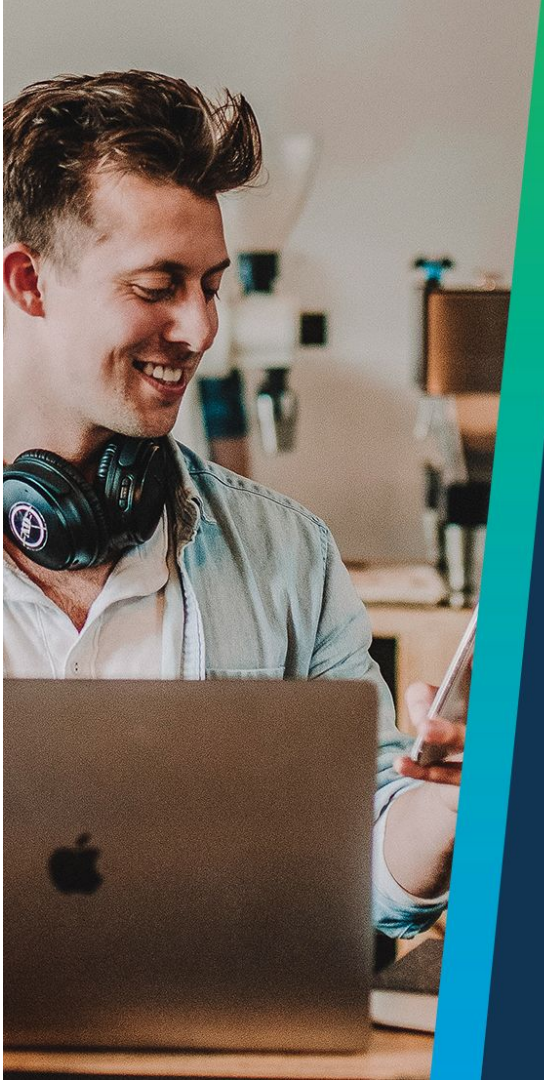


Software Engineering Bootcamp

Hyperiondev

Red-Green-Refactoring Cycle & Hypothesis-driven Debugging

Lecture – Housekeeping

- The use of disrespectful language is prohibited in the questions, this is a supportive, learning environment for all - please engage accordingly.
- No question is daft or silly - **ask them!**
- There are Q/A sessions midway and at the end of the session, should you wish to ask any follow-up questions.
- You can also submit questions here: http://hyperiondev.com/sbc4-se-questions
- For all non-academic questions, please submit a query: www.hyperiondev.com/support
- Report a safeguarding incident: http://hyperiondev.com/safeguardreporting
- We would love your feedback on lectures: https://hyperionde.wufoo.com/forms/zsgv4m40ui4i0g/

Github Repository – Lecture Examples/Slides

https://github.com/HyperionDevBootcamps/C4_SE_lecture_examples

Pytest documentation

https://docs.python.org/3/library/unittest.html

https://docs.pytest.org/en

Objectives

1. The Red-Green-Refactor Cycle
    a. A brief introduction to the RGR cycle
    b. The 3 phases of the RGR cycle
    c. Benefits of the RGR cycle
2. Hypothesis-Driven Debugging
    a. Key steps involved in HD debugging
    b. Benefits of HD debugging

Recap

What is unit testing?

- Unit testing is a software testing method where individual units or components of a software application are tested in isolation to ensure they work as intended. The goal of unit testing is to verify that each unit of the software performs as designed and that all components are working together correctly.

- Unit testing helps developers catch bugs early in the development process, when they are easier and less expensive to fix. It also helps ensure that any changes made to the code do not cause unintended consequences or break existing functionality.

What is the AAA pattern?

The AAA pattern is a common pattern used in unit testing to structure test cases. It stands for Arrange, Act, Assert.

- ➔ Arrange: Set up any necessary preconditions or test data for the unit being tested.
- ➔ Act: Invoke the method or code being tested.
- ➔ Assert: Verify that the expected behavior occurred.

Using the AAA pattern helps make unit tests more readable and easier to maintain. It also helps ensure that all necessary steps are taken to properly test the unit being tested.

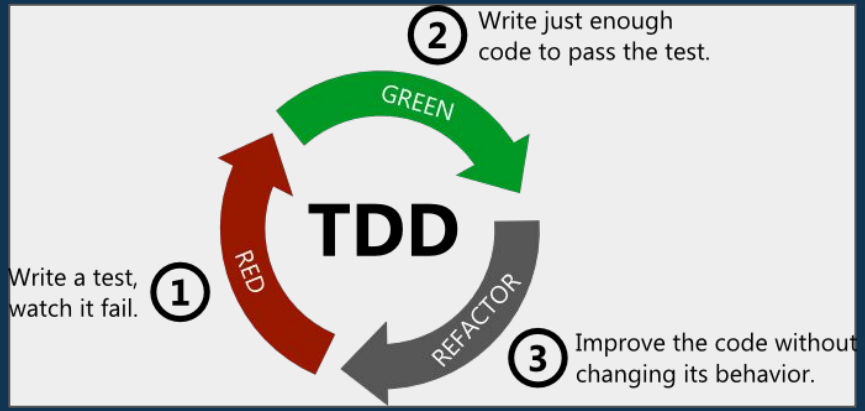
Red-Green-Refactor

A brief introduction to the concept of the Red-Green-Refactor Cycle

The Red-Green-Refactor Cycle is a fundamental practice in test-driven development (TDD) that promotes writing automated tests before writing the actual code. It consists of three phases: Red, Green, and Refactor.

The 3 phases of the Red-Green-Refactoring Cycle

- Red: Write a failing test.
- Green: Write the minimum amount of code to make the test pass.
- Refactor: Improve the code without changing its behavior.



Benefits of the Red-Green-Refactor Cycle

- Ensures that code is testable and has good test coverage:

  Starting with a failing test (Red phase) ensures that the code is testable from the beginning.

- Encourages a focus on writing simple, modular, and maintainable code:

  During the Green phase, developers write the minimum amount of code necessary to make the tests pass. This encourages a mindset of keeping code concise, focused, and free from unnecessary complexity

- Provides a safety net for making changes without introducing regressions:

  - ❏ One of the key benefits of having a comprehensive test suite is the safety net it provides when making changes to the codebase.
  - ❏ When you have a solid set of tests, you can run them after making modifications (refactoring or adding new features) to ensure that everything still functions as expected.
  - ❏ If any of the tests fail, it indicates a regression, signaling that something has broken.
  - ❏ This early detection of regressions allows developers to quickly identify and fix issues, preventing potential bugs from reaching production and minimizing the risk of unintended side effects.

Tips for Effective Application

- Start with small, incremental tests.
- Keep a feedback loop of running tests frequently.
- Prioritize writing tests that cover critical functionality.
- Refactoring is crucial to improve code quality.

Hypothesis-Driven Debugging

Hypothesis-Driven Debugging

The key steps involved in this approach:

- Identify the problem or failure.
- Formulate a hypothesis about the cause.
- Design and conduct experiments to test the hypothesis.
- Analyze the results and refine the hypothesis if necessary.
- Repeat the process until the issue is resolved.

Benefits of Hypothesis-Driven Debugging

The advantages of using this approach in debugging:

- Provides a structured and systematic approach to problem-solving.
- Helps in narrowing down the potential causes of the issue.
- Saves time and effort by focusing on relevant experiments.
- Facilitates learning from debugging experiences and improving future debugging skills.

Hyperiondev

Q & A Section

Please use this time to ask any questions relating to the topic explained, should you have any

Hyperiondev

Thank you for joining us

Stay hydrated
Avoid prolonged screen time
Take regular breaks
Have fun :)