Software Engineering Bootcamp

Hyperiondev

Test Triangulation

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/unittest documentation

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

https://docs.pytest.org/en

Objectives

1. Recap
    a. Red-Green-Refactor cycle
    b. Hypothesis Driven Debugging
2. Test Triangulation
    a. What is test triangulation
    b. How do we implement it

Recap

Hyperiondev

Red-Green-Refactor

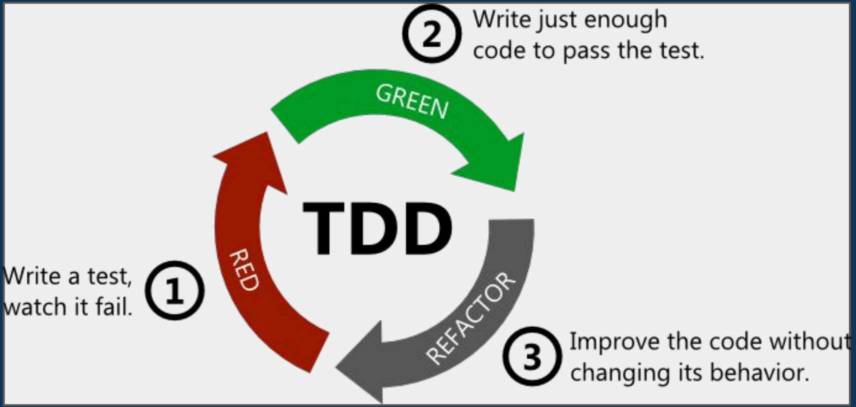
Hyperiondev

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.



2 Write just enough code to pass the test.
GREEN
TDD
RED
REFACTOR
1 Write a test, watch it fail.
3 Improve the code without changing its behavior.

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.

Test Triangulation

Test Triangulation

- When writing tests it is easy to become stuck and not know the next test to write
- This occurs when we write a test that forces us to write to much implementation/code
- We can also feel like we are going sideways when we use the most basic code to get to the green phase
- This is where test triangulation come into play

Test Triangulation

- It's the 3rd cycle of Test-Driven-Development called Mini Cycle of TDD
- TDD is an iterative software development process that involves writing tests before writing the actual code.
- Test triangulation emphasizes the idea of using multiple tests to validate and verify the behavior of the code being developed.
- Test triangulation is where we use our refactoring phase to make the implementation more general

Test Triangulation

- We will use small steps combined with specific tests to make our code more general
- When writing a test for the next requirement, we are introducing the fact that we need an if statement with a certain body
- We have to add a condition to the if statement and we add a very specific condition to pass our tests
- Next, we are proving that this condition is too specific by writing a test, and then making it pass with a more generic solution

Advantages of Test Triangulation

- **Ensuring comprehensive coverage:**

  By adding multiple tests, you can cover a wider range of scenarios, including edge cases, corner cases, and exceptional inputs. This helps uncover potential bugs or issues that might not be apparent with just a single test.

- **Validating code behavior:**

  The presence of multiple tests that cover different aspects of the code's behavior increases the confidence in the correctness of the implementation. If all tests pass, it provides stronger evidence that the code is functioning as intended.

Advantages of Test Triangulation

- **Facilitating code evolution:**

  As you add more tests, you can confidently refactor the code, knowing that the tests act as a safety net. The tests will help catch any regressions or unintended side effects of code modifications.

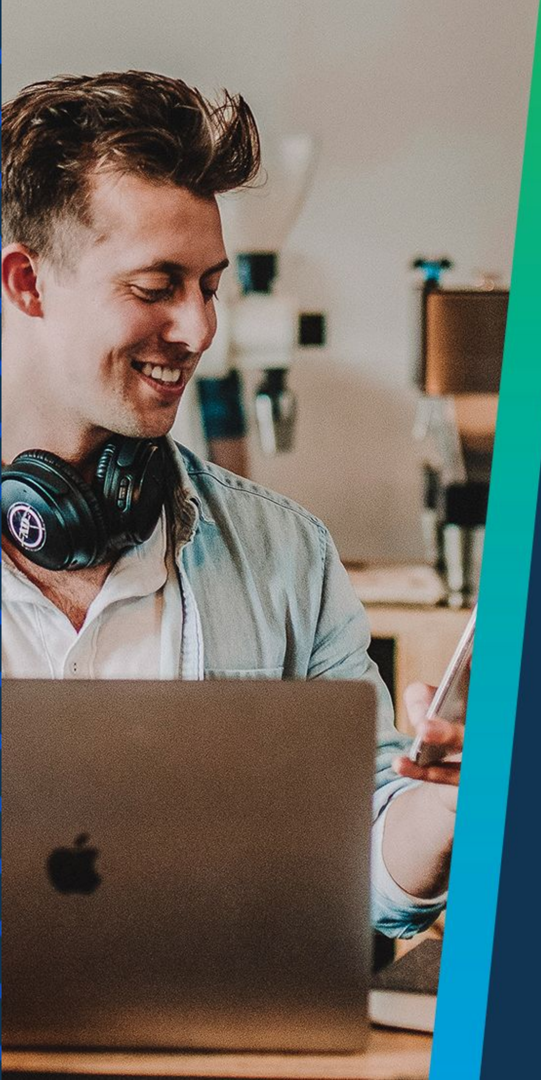
- **Enhancing maintainability:**

  A suite of well-written tests can serve as documentation for the codebase, providing insights into its expected behavior. This makes it easier for developers to understand and maintain the code over time.

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 :)