Test-Driven Development and Continuous Integration

17-316/616 Fall 2025

AI Tools for Software Development

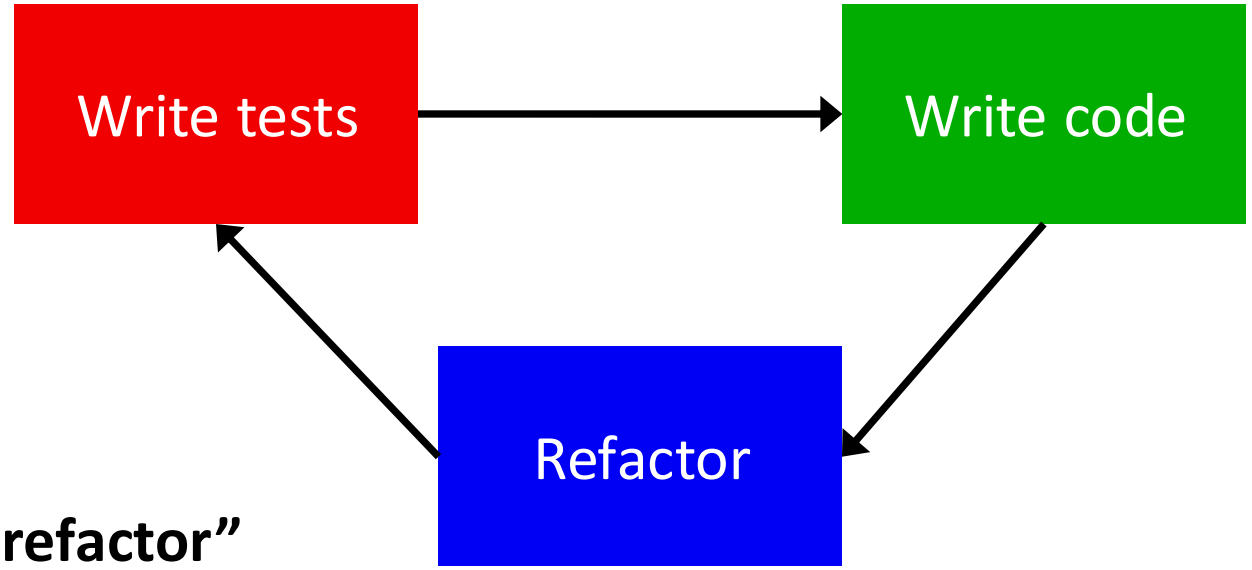
https://ai-developer-tools.github.io

Austin Henley and Andrew Begel

Today

- Test-Driven Development
- Code coverage
- Continuous integration using GitHub Actions
- I will ask one person from each team to build and run your app to prove that it works

Test-Driven Development



”red, green, refactor”

- Write tests that fail
- Write code that passes tests
- Refactor to make the code better

Test-Driven Development

- No product code except to make a failing test pass.
- Write test cases before product code

- Initially, all tests will fail since the code it is testing does not yet exist.
- Keep tests small.
- Write simplest code to pass the tests.

- After all tests pass, refactor!

Code Coverage

- Degree to which source code of a program is tested by a test suite

- Statement coverage
  - Each program statement (line or basic block) is executed at least once
- Branch coverage
  - Each Boolean expression in control structures evaluates to true at least once and to false at least once
- Path coverage
  - Each possible path through a program is taken at least once

Continuous Integration

- Team uses version control and commits frequently
- Automated tests run after code is committed or merged
- Automated build after code is committed or merged
- Team is notified is tests or build fail

- You can configure any number of automations to trigger
  - Linter, Discord/Slack notification,, unit tests, browser tests, Docker image build

Remember to use “good” prompts…

- Provide context and be explicit
- Use strategies we have covered in class
- “Fix it” is not a good prompt

Activity: Continuous Integration

- Get with your project group
- Use an AI to help you setup GitHub Actions in your repo
- Your unit tests should run automatically after committing
- A linter should run automatically after committing

Upcoming

- Discussion!
- P5 is live