Software Engineering Bootcamp

Hyperiondev

Unit Testing & The Arrange-Act-Assert Pattern for Structuring Tests

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.pytest.org/en

Objectives

1. Unit Testing
    a. What is unit testing?
    b. How do we use it to write better code?
2. The Arrange-Act-Assert Pattern (AAA)
    a. What is the AAA pattern?
    b. How do we structure tests according to the AAA pattern?
    c. Running tests using pytest.

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.

Using unit tests to write better code

Advantages:

- Catch errors early
- Improve code quality
- Refactor with confidence
- Document code behavior
- Facilitate collaboration

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.

Writing a unit test in Python using the AAA pattern

Let's have a look at an example of how to write a unit test in Python using the AAA pattern.

Consider a simple function that adds two numbers:

```
def add_numbers(a, b):
    return a + b
```

To test this function, we would create a new function called test_add_numbers (note that the name must start with test_ for the Python test runner to find it).

```
def test_add_numbers():
    # Arrange
    a = 2
    b = 3

    # Act
    result = add_numbers(a, b)

    # Assert
    assert result == 5
```

In this example, we've set up the test data (Arrange) by creating two variables a and b with the values 2 and 3.

We then invoke the function being tested (Act) and store the result in a variable called result.

Finally, we assert that the result is equal to the expected value of 5 (Assert).

Running the test

- We can then run this test using a Python test runner like **pytest**. If the test passes, we can be confident that the add_numbers function is working correctly.
- Note: If you're using a virtual environment for your Python projects, make sure you activate the environment before installing pytest and running your tests.

To create a virtual environment run:	***python -m venv <venv_name>***
To activating the venv run:	***.\<venv_name>\Scripts\activate***

How to install pytest

1. Open a command prompt or terminal on your computer.
2. Ensure that you have pip installed by running the following command: pip --version

   If pip is not installed, you can install it by following the instructions at https://pip.pypa.io/en/stable/installation/.
3. Install pytest using pip by running the following command: pip install pytest
4. Once the installation is complete, verify that pytest is installed by running the

   following command: pytest --version

   This should display the version of pytest that you just installed.

Now you're ready to start writing and running tests with pytest!

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 :)