Introduction to Software Development Week 9: Introduction to Software Testing

1. Learning Objectives

- Explain the importance of software testing in the SDLC.
- Define "unit testing".
- Write simple assertion-based tests to verify the correctness of functions.
- Understand the concept of Test-Driven Development (TDD).

2. Core Concepts

- **Why Test?**
  - To ensure the software meets its requirements.
  - To find bugs early, when they are cheaper to fix.
  - To provide confidence when refactoring or adding new features.
  - To serve as documentation for what the code is supposed to do.
- **Unit Testing:**
  - A level of software testing where individual units or components of the software are tested in isolation.
  - For our purposes, a "unit" is typically a single function or method.
  - **Assertion:** A statement that a condition is true. If the condition is false, the test fails.
- **The AAA Pattern for Tests:**
  - **Arrange:** Set up the necessary preconditions and inputs.
  - **Act:** Call the function or method being tested.
  - **Assert:** Verify that the output or result is what you expected.
- **Test-Driven Development (TDD) - A Glimpse:**
  - A development process where you write the tests *before* you write the code.
  - Cycle: **Red** (write a failing test) -> **Green** (write the minimum code to make the test pass) -> **Refactor** (clean up the code).

3. Code Examples (using Python's built-in `assert`)

```
# The function to be tested

def add(a, b):

   return a + b


# A simple test for the add function

def test_add_positive_numbers():

   # Arrange

   num1 = 5

   num2 = 10
```

```
    # Act
    result = add(num1, num2)
    # Assert
    assert result == 15
    print("Test passed!")

def test_add_negative_numbers():
    # Arrange
    num1 = -2
    num2 = -2
    # Act
    result = add(num1, num2)
    # Assert
    assert result == -4
    print("Test passed!")

# Run the tests
test_add_positive_numbers()
test_add_negative_numbers()
```

4. Summary

Testing is not an afterthought; it is an integral part of professional software development. Unit tests form the foundation of a robust testing strategy, ensuring that the core components of your application work as intended.