

## Python Unit Testing

**Level:** medium

**Length:** 8 hours

**Course Objective:** enter the unit testing in software development world: become familiar with the available tools, how they are used to construct test suites, the importance of integrating them in CI/CD (Continuous Integration/Continuous Delivery) pipelines.

### What You Will Learn

- Understand the role and importance of unit testing in software development.
- Learn how to set up and write unit tests using Python's built-in unittest module.
- Explore pytest, a more advanced and popular testing framework.
- Understand test-driven development (TDD) concepts.
- Be able to mock external dependencies using unittest.mock.
- Testing and CI/CD workflows.
- Gain hands-on experience writing effective unit tests for real-world Python applications

### Who can participate

- Software developers (beginner to intermediate) who want to improve the reliability of their Python applications.
- Quality assurance engineers interested in automating tests.
- DevOps professionals looking to incorporate testing into their CI/CD pipelines.

**Prerequisites:** comfortable with Python at least at the basic level

**Required infrastructure:** workstation with Python installed and an IDE for writing code, for example Community PyCharm

**Related Courses:** Fundamentals of Python, Advanced Python Programming

## Contents

### Introduction to Unit Testing

- What is unit testing?
- Why is unit testing important?
- Unit testing vs. integration testing vs. system testing.
- Key principles: Isolation, repeatability, and automation.

### Writing Your First Unit Test with unittest

- Structure of a test case.
- Assertions and how they work.
- Running and organizing tests using unittest.

### Advanced Unit Testing with unittest

- Grouping and organizing tests using test suites.
- Handling setup and teardown with setUp() and tearDown().
- Skipping tests and expecting failures.

### Introducing pytest

- Why choose pytest?
- Key features of pytest (simplicity, fixtures, parameterization).
- Writing and running tests with pytest.
- Using pytest plugins for code coverage, etc.

### Mocking and Patching

- What is mocking, and when to use it?
- Using unittest.mock to replace objects in your code.
- Mocking functions, classes, and external APIs.
- Practical examples of mocking in unit tests.

### Test-Driven Development (TDD) Approach

- Introduction to TDD.
- Writing tests first: Red, Green, Refactor.
- Benefits of TDD in maintaining code quality and preventing regressions.
- Hands-on examples of TDD in Python.

### Continuous Integration and Testing

- Overview of CI/CD and automated testing in pipelines.
- Integrating unit tests with tools like Jenkins, Travis CI, or GitHub Actions.
- Running tests automatically on each commit.

### Best Practices for Writing Unit Tests

- What makes a good test? (Readable, maintainable, fast, isolated).
- Common pitfalls and how to avoid them.
- Refactoring tests as code evolves.