

Selenium WebDriver Test Automation with Python

Browser interactions, pytest and reusable data-driven tests

FA-0750 · 4 days

Software Development · DevOps, Cloud & Infrastructure

Introduction

Develop automated browser tests using Python and Selenium WebDriver. Begin with the Python constructs needed for tests, then practise locators, browser interactions, waits, assertions and selected end-to-end examples.

Organise a small test suite with pytest and page objects, and introduce Excel-based test data, logging, Git/GitHub and CI execution. The course is intended for learners with some programming experience rather than as a complete first programming course.

Learning outcomes

- Use selected Python syntax, collections, control flow, functions, exceptions, files and classes.
- Locate elements and automate selected Chrome/Firefox/Edge interactions.
- Apply suitable waits and browser/session cleanup.
- Organise tests with pytest, reusable fixtures, assertions and reports.
- Use page objects and a supplied Excel test-data example.
- Add Python logging and explain Git/GitHub and Jenkins integration in a test workflow.

Prerequisites

- Programming experience in any language and basic file-system/internet knowledge.

- A supported Python environment, compatible browsers and IDE, with approved dependency-installation permissions.
- Reliable internet and authorised access to the designated practice application.
- Remote/SSH access only where the CI or remote-execution demonstration requires it.

Training outline

1. Day 1 — Python for Test Code

- Identifiers, reserved words, indentation, comments and basic CLI/IDE use.
- Numbers, lists, tuples, dictionaries, sets, copying, strings/formatting and regular-expression overview.
- Operators and precedence; conditions, loops, break and continue.
- Functions, parameters, documentation, collections, variable arguments, scope, map/filter/lambda.
- Imports, aliases and namespaces; exception handling/raising.
- File modes and read/write examples; classes, attributes, methods, constructors, inheritance and polymorphism.
- Selected programming exercises rather than comprehensive Python mastery.

2. Day 2 — Selenium and Web Elements

- WebDriver methods and compatible Chrome, Firefox and Edge examples.
- Inspect HTML with browser developer tools; use name, text, CSS and XPath locators.
- Parent/child traversal and scoped element selection, without requiring a specific locator browser plugin.
- Extract text and write validation assertions.
- Static dropdowns with Select, dynamic suggestions, checkboxes and radio buttons.
- JavaScript alerts and selected practice-application exercises.

3. Day 3 — Synchronisation and Browser Behaviour

- Develop an end-to-end functional test.
- Implicit versus explicit waits; choose an appropriate strategy without mixing them.
- HTML tables, child tabs/windows and frames.
- Python ActionChains for keyboard/mouse actions, double-click and context-click.
- Selected JavaScript-executor operations and browser options.
- Handle failures and clean up driver sessions.

4. Day 4 – Framework, Test Data and Integrations

- Organise pytest tests, fixtures, assertions and execution results/reports.
- Page-object separation and a small reusable framework.
- Data-driven cases from a supplied XLSX file using an appropriate library such as openpyxl.
- Use Python standard logging rather than Java Log4j.
- Introduce Git/GitHub version control and a supplied Jenkins CI execution/report example.
- Review unit-test versus browser-test responsibilities.
- End-to-end exercise and assessment.