

Why Playwright + Python

- Cross-browser support with modern primitives
- Strong locator model and web-first assertions
- Python ecosystem integrates well with QA workflows

References: <https://playwright.dev/docs/intro> |
<https://docs.pytest.org/en/stable/>

Core Playwright Workflow

- Launch browser/context/page deterministically
- Use resilient locators and explicit assertions
- Collect traces/screenshots on failures

References: <https://playwright.dev/docs/intro> | <https://playwright.dev/docs/test-assertions>

Pytest Integration Patterns

- Use fixtures for setup/teardown reuse
- Parametrize tests to scale coverage
- Use markers to target suites by risk

References: <https://docs.pytest.org/en/stable/>

Data-Driven and Environment-Aware Tests

- Read config from environment variables
- Separate secrets from source code
- Use clean test data lifecycle per run

References: <https://12factor.net/config>

Debugging Failing UI Tests

- Reproduce with headed mode and traces
- Inspect network/console before changing assertions
- Fix product bugs before patching tests

References: <https://playwright.dev/docs/trace-viewer-intro>

