CheatSheet For Automation Testing With Selenium & Python

1. Python FirstBasics

- Variables, data types, and operators – Learn how to declare variables and use basic arithmetic/logical operators.
- Strings, lists, and dictionaries – Master string operations, slicing, list methods, and key-value access in dicts.
- Loops & conditionals – for/while loops, if-elif-else statements with practical usage in automation scripts.
- Functions – Defining reusable blocks of code with parameters and return values.
- File handling – Reading/writing files for data-driven testing.

Intermediate

- Exceptions – Use try/except to handle runtime errors gracefully in tests.
- OOP concepts – Classes, objects, and inheritance for structuring automation frameworks.
- Modules & packages – Organizing reusable test utilities.
- Virtual environments – Keeping project dependencies isolated.
- Working with JSON, CSV, XML – Reading and parsing common test data formats.

2. Testing FoundationTesting Basics

- STLC & SDLC – Understand how testing fits into the software lifecycle.
- Test case writing – How to write effective, reusable, and clear test cases.
- Types of testing – Unit, integration, system, UAT, and automation scope.

Git & Setup

- Git basics – clone, commit, push, and branching strategies.
- IDE setup – Configuring PyCharm or VSCode for Selenium + Python.

3. Selenium UICore Skills

- Setup & first script – Installing Selenium, writing first browser automation.
- Locators – Mastering id, xpath, css selector strategies.
- WebElement actions – Click, send_keys, select dropdowns.
- Waits – Implicit vs explicit waits to handle dynamic elements.
- Navigation & windows – Switching between tabs, alerts, and popups.

Advanced

- ActionChains – Mouse hover, drag-and-drop, right click automation.
- Frames & Shadow DOM – Handling iframes and shadow DOMs.
- Tables handling – Extracting structured data from web tables.

4. FrameworksPytest

- Test functions & assertions – Writing unit-style test cases.
- Fixtures – Setup/teardown reusable blocks for tests.
- Parametrization – Running tests with multiple datasets.
- Markers – Grouping, skipping, or prioritizing tests.

Others

- Unittest basics – Traditional xUnit style testing.
- Robot Framework – Keyword-driven testing approach.

5. Framework DesignArchitecture

- Page Object Model – Structuring page classes for maintainability.
- BasePage & TestPages – Common methods and reusable test steps.
- Utilities – Logging, configuration files, helper functions.
- Test data management – Externalizing test data for flexibility.

Reporting

- Pytest-html / Allure – Generating rich test reports.
- Screenshots & logging – Capturing state during test failures.

6. Beyond SeleniumAPI & Database

- API testing using requests library – GET/POST automation.
- Database testing – Running queries to validate backend.
- CRUD operations – Automating create, read, update, delete validation.

BDD

- Behave & pytest-bdd – Writing tests in Given-When-Then format.
- Gherkin syntax – Business-readable test definitions.

7. CI/CD IntegrationPipelines
- Jenkins & GitHub Actions – Automating test execution in pipelines.
- Test artifacts – Collecting logs, reports, and screenshots.
- Parallel execution – Speeding up test runs.
- Notifications – Slack/email integration for test results.
Grid & Cloud
- Selenium Grid – Running tests in distributed environments.
- BrowserStack / SauceLabs – Cloud-based cross-browser testing.

8. Scaling & AdvancedReliability

- Flaky test handling – Stabilizing inconsistent tests.
- Retry mechanisms – Retrying failed cases automatically.
- Quarantine strategy – Isolating unstable tests.

Enterprise

- Dockerize Selenium – Running tests in containers.
- Kubernetes scaling – Scaling test executions dynamically.
- ReportPortal – Centralized test result tracking.