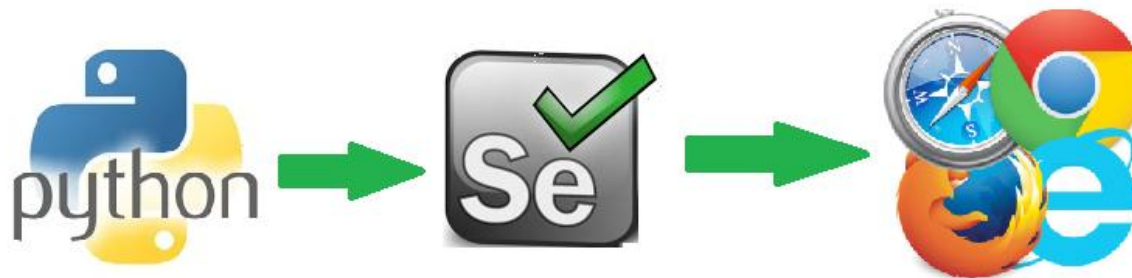




Selenium with Python – Automating the Web



Introduction to Selenium

- Selenium is an open-source tool for automating web browsers.
 - Originally developed for testing web applications.
 - Works with all major browsers and OS.
 - Supports multiple programming languages: Python, Java, C#, Ruby, etc.
-

Use Selenium with Python?

- Python is easy to write, read, and debug.
 - Rich ecosystem and libraries.
 - Great for test automation, scraping, and bots.
-

Selenium Components

- ☐ Selenium WebDriver – Core API to automate browser actions.
 - ☐ Selenium IDE – Chrome/Firefox extension for record and playback.
 - ☐ Selenium Grid – Run tests on multiple machines and browsers.
-

Installing Selenium in Python

Using pip

```
pip install selenium
```

- ❑ Also download appropriate **WebDriver** (ChromeDriver, GeckoDriver, etc.)

First Selenium Script (Example)

```
from selenium import webdriver

driver = webdriver.Chrome()
driver.get("https://www.google.com")
print(driver.title)
driver.quit()
```

Locating Web Elements

- By ID: `driver.find_element(By.ID, "id")`
 - By Name: `driver.find_element(By.NAME, "name")`
 - By Class: `driver.find_element(By.CLASS_NAME, "class")`
 - By Tag: `driver.find_element(By.TAG_NAME, "tag")`
 - By XPath: `driver.find_element(By.XPATH, "//tag[@attr='value']")`
 - By CSS Selector: `driver.find_element(By.CSS_SELECTOR, "css")`
-

Performing Actions

```
element.click()
```

```
element.send_keys("text")
```

```
element.clear()
```

Other actions:

☐ `submit()`

☐ `get_attribute("value")`

Handling Forms and Buttons

```
search_box = driver.find_element(By.NAME, "q")  
search_box.send_keys("Selenium Python")  
search_box.submit()
```

Waits in Selenium

❑ **Implicit Wait:**

```
driver.implicitly_wait(10)
```

❑ **Explicit Wait:**

```
from selenium.webdriver.support.ui import WebDriverWait  
from selenium.webdriver.support import expected_conditions as EC
```

```
WebDriverWait(driver, 10).until(EC.presence_of_element_located((By.ID,  
"some_id")))
```

Handling Alerts

`alert = driver.switch_to.alert`

`alert.accept()`

`alert.dismiss()`

`alert.text`

Handling Frames and Windows

```
driver.switch_to.frame("frame_name")
```

```
driver.switch_to.default_content()
```

```
driver.switch_to.window("window_handle")
```

Taking Screenshots

```
driver.save_screenshot("screenshot.png")
```

Working with Cookies

```
driver.get_cookies()
```

```
driver.add_cookie({"name": "foo", "value": "bar"})
```

```
driver.delete_all_cookies()
```

Closing the Browser

`driver.close()` # Closes the current tab

`driver.quit()` # Quits the entire browser

Use Cases of Selenium

- Web Testing
 - Web Scraping
 - Automation Bots
 - Form Submission
 - Data Extraction
-

Best Practices

- Use explicit waits wisely.
 - Avoid hard sleeps.
 - Use try/except for error handling.
 - Modularize code using functions/classes.
 - Combine with testing frameworks like unittest or pytest.
-

Alternatives to Selenium

- Playwright
 - Puppeteer (Node.js)
 - BeautifulSoup (for static scraping)
 - Scrapy (for scraping only)
-

Demo / Hands-on (Optional)

- Automate Google Search.
- Extract weather data.
- Automate login to a demo site

Resources

- <https://selenium.dev>
- [Selenium with Python Docs](#)
- ChromeDriver:
<https://sites.google.com/chromium.org/driver/>