skillzam

Python Selenium CheatSheet

Selenium is a powerful tool for **automating web browsers**. Selenium is a popular open-source framework used for **automated testing** of web applications across different browsers and platforms. It allows you to write test scripts in various programming languages like:

- Python
- JavaScript
- Java
- C#
- Ruby

Key Components of Selenium:

1. **Selenium WebDriver**:
    - The most widely used component.
    - Allows you to programmatically control a web browser (e.g., Chrome, Firefox, Edge) as if a human were interacting with it.
    - Commonly used for end-to-end testing.
2. **Selenium IDE**:
    - A browser extension (Chrome/Firefox).
    - Provides a simple record-and-playback tool for creating test cases without coding.
    - Good for beginners.
3. **Selenium Grid**:
    - Allows you to run tests on multiple machines and browsers in parallel.
    - Useful for speeding up test execution and cross-browser testing.

Common Use Cases:

- Automated testing of websites and web applications
- Regression testing (to ensure old features still work)
- Scraping web data (though Selenium is overkill for simple scraping)
- Continuous Integration/Continuous Deployment (CI/CD) pipelines

skillzam

Selenium WebDriver operations in Python with examples

1. Setup & Installation

Install Selenium & WebDriver Manager

```
pip install selenium webdriver-manager
```

Import Required Modules

```
from selenium import webdriver
from selenium.webdriver.chrome.service import Service
from selenium.webdriver.chrome.options import Options
from webdriver_manager.chrome import ChromeDriverManager
from selenium.webdriver.common.by import By
from selenium.webdriver.common.keys import Keys
from selenium.webdriver.common.action_chains import ActionChains
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
```

Launch Chrome Browser

```
# Basic setup
driver = webdriver.Chrome(service=Service(ChromeDriverManager().install()))
driver.get("https://example.com")

# With options (maximized, headless)
chrome_options = Options()
chrome_options.add_argument("--start-maximized")
chrome_options.add_argument("--headless")  # Run without GUI
driver = webdriver.Chrome(service=Service(ChromeDriverManager().install()),
options=chrome_options)
```

2. Locating Elements

Locator	Method	Example
ID	`find_element(By.ID, "id")`	`driver.find_element(By.ID, "username")`
Name	`find_element(By.NAME, "name")`	`driver.find_element(By.NAME, "email")`
Class Name	`find_element(By.CLASS_NAME, "class")`	`driver.find_element(By.CLASS_NAME, "btn")`
Tag Name	`find_element(By.TAG_NAME, "tag")`	`driver.find_element(By.TAG_NAME, "input")`
Link Text	`find_element(By.LINK_TEXT, "text")`	`driver.find_element(By.LINK_TEXT, "Sign Up")`
Partial Link Text	`find_element(By.PARTIAL_LINK_TEXT, "partial")`	`driver.find_element(By.PARTIAL_LINK_TEXT, "Sign")`
CSS Selector	`find_element(By.CSS_SELECTOR, "selector")`	`driver.find_element(By.CSS_SELECTOR, "#login .btn")`
XPath	`find_element(By.XPATH, "xpath")`	`driver.find_element(By.XPATH, "//input[@name='username']")`

3. Interacting with Elements

Send Text to Input Field

```
element = driver.find_element(By.ID, "username")
element.send_keys("testuser")
```

Clear Input Field

```
element.clear()
```

Click a Button/Link

```
driver.find_element(By.ID, "submit-btn").click()
```

Check if Element is Displayed/Enabled/Selected

```
element = driver.find_element(By.ID, "checkbox")
print(element.is_displayed())  # True/False
print(element.is_enabled())    # True/False
print(element.is_selected())   # True/False (for checkboxes/radio buttons)
```

Get Element Text & Attributes

```
element = driver.find_element(By.ID, "header")
print(element.text)  # Get visible text
print(element.get_attribute("href"))  # Get attribute value
```

---

4. Dropdowns (Select Class)

```
from selenium.webdriver.support.select import Select

dropdown = Select(driver.find_element(By.ID, "country"))
dropdown.select_by_visible_text("USA")  # Select by text
dropdown.select_by_value("us")          # Select by value
dropdown.select_by_index(1)             # Select by index
```

---

5. Handling Alerts & Popups

```
# Accept alert
alert = driver.switch_to.alert
print(alert.text)  # Get alert text
alert.accept()     # Click OK

# Dismiss alert
alert.dismiss()

# Send text to prompt
alert.send_keys("Yes")
alert.accept()
```

---

6. Waits (Implicit & Explicit)

Implicit Wait (Global Wait for All Elements)

```
driver.implicitly_wait(10)  # Wait up to 10 sec for elements
```

Explicit Wait (Wait for a Specific Condition)

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

element = WebDriverWait(driver, 10).until(
    EC.presence_of_element_located((By.ID, "dynamic-element"))
)
```

Common Expected Conditions:

- `EC.presence_of_element_located` (Element exists in DOM)
- `EC.visibility_of_element_located` (Element is visible)
- `EC.element_to_be_clickable` (Element is clickable)
- `EC.title_contains("Welcome")` (Page title contains text)

---

7. Keyboard & Mouse Actions

Keyboard Actions (Send Special Keys)

```
from selenium.webdriver.common.keys import Keys

driver.find_element(By.ID, "search").send_keys("Selenium" + Keys.ENTER)
```

Mouse Actions (Hover, Drag & Drop, Right-Click)

```
from selenium.webdriver.common.action_chains import ActionChains

element = driver.find_element(By.ID, "menu")
action = ActionChains(driver)
action.move_to_element(element).click().perform()  # Hover & Click
action.context_click(element).perform()            # Right-Click
action.drag_and_drop(source, target).perform()     # Drag & Drop
```

---

8. Handling Frames & Windows

Switch to Frame

```
driver.switch_to.frame("frame-name")  # By name, ID, or index
driver.switch_to.default_content()    # Switch back to main page
```

Switch to New Window/Tab

```
driver.find_element(By.LINK_TEXT, "Open New Tab").click()
driver.switch_to.window(driver.window_handles[1])  # Switch to new tab
driver.close()  # Close current tab
driver.switch_to.window(driver.window_handles[0])  # Switch back
```

---

9. Screenshots & Cookies

Take Screenshot

```
Download
```

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

Manage Cookies

```
# Add cookie
driver.add_cookie({"name": "test", "value": "123"})

# Get cookies
print(driver.get_cookies())

# Delete cookies
driver.delete_all_cookies()
```

---

10. Closing Browser

```
driver.close()      # Close current window
driver.quit()       # Quit entire browser session
```

---

11. Navigation

```
driver.get("https://example.com")

driver.back()     # Go back

driver.forward()  # Go forward

driver.refresh()  # Refresh page
```

Final Notes

- Always use `try-finally` to ensure the browser closes properly.
- Prefer **Explicit Waits** over `time.sleep()` for better efficiency.
- Use **XPath/CSS Selectors** for complex element locating.