QASSERT
Where quality is the core

Cheat sheet for using Selenium with Python

Installation

Before you start using Selenium with Python, you need to install it:

```
pip install selenium
```

You also need to download the appropriate WebDriver for the web browser you intend to use (e.g., Chrome, Firefox, Edge). Make sure the WebDriver executable is in your system's PATH.

Import Selenium

```
from selenium import webdriver
```

Basic Usage

```
# Initialize a web driver
driver = webdriver.Chrome()  # or webdriver.Firefox(), webdriver.Edge(), etc.
```

Navigate to a URL

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

Find elements by various methods

```
element = driver.find_element_by_id("element_id")
elements = driver.find_elements_by_css_selector(".element_class")
element = driver.find_element_by_name("element_name")
element = driver.find_element_by_link_text("Link Text")
element = driver.find_element_by_partial_link_text("Partial Link Text")
element = driver.find_element_by_xpath("//xpath_expression")
element = driver.find_element_by_tag_name("tag_name")
elements = driver.find_elements_by_class_name("class_name")
element = driver.find_element(By.NAME, "element_name")
elements = driver.find_elements(By.PARTIAL_LINK_TEXT, "Partial Link Text")
```

Interact with elements

```
element.click()
element.send_keys("Text to input")
```

QASSERT
Where quality is the core

Perform common browser actions

```
driver.back()
driver.forward()
driver.refresh()
driver.quit()  # Close the browser
```

Waits You should use waits to ensure the page is fully loaded before interacting with elements:

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

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

Handling Frames and Windows

```
#Switch to a frame by name, id, or index
driver.switch_to.frame("frame_name")
driver.switch_to.frame("frame_id")
driver.switch_to.frame(0)  # Switch to the first frame

# Switch back to the main content
driver.switch_to.default_content()

# Open a new window or tab
driver.execute_script("window.open('https://example.com', '_blank')")

# Switch to a new window or tab
driver.switch_to.window(driver.window_handles[-1])
```

Actions For performing complex actions like dragging and dropping:

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

element = driver.find_element_by_id("source_element")
target = driver.find_element_by_id("target_element")

action = ActionChains(driver)
action.drag_and_drop(element, target).perform()
```

Running JavaScript You can execute JavaScript code in the browser:

```
driver.execute_script("document.title = 'New Title';")
```

Taking Screenshots

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

Handling Alerts

```
alert = driver.switch_to.alert
alert.accept()  # Accept the alert
alert.dismiss()  # Dismiss the alert
```

Browser Options You can configure the browser with options:

```
from selenium.webdriver.chrome.options import Options

chrome_options = Options()
chrome_options.add_argument("--headless")  # Run Chrome in headless mode
driver = webdriver.Chrome(chrome_options=chrome_options)
```

Dropdowns and Select For handling dropdown menus:

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

select = Select(driver.find_element_by_id("dropdown_element"))
select.select_by_index(2)  # Select by index
select.select_by_value("option_value")  # Select by value
select.select_by_visible_text("Option Text")  # Select by visible text
```

Cookies Manipulate cookies:

```
# Get all cookies
cookies = driver.get_cookies()

# Add a cookie
driver.add_cookie({"name": "cookie_name", "value": "cookie_value"})

# Delete a cookie by name
driver.delete_cookie("cookie_name")

# Delete all cookies
driver.delete_all_cookies()
```

Page Navigation Navigate between pages:

```
# Navigate forward and backward in history
driver.forward()
driver.back()

Refresh the current page
driver.refresh()
```

Screenshots Capture specific elements:

```
from selenium.webdriver.common.by import By
element = driver.find_element(By.ID, "element_id")
element.screenshot("element_screenshot.png")
```

Handling iframes You can switch to iframes to interact with elements inside them:

```
iframe = driver.find_element(By.ID, "iframe_id")
driver.switch_to.frame(iframe)
# Perform actions within the iframe
driver.switch_to.default_content()  # Switch back to the main content
```

Headless Browsing Run the browser in headless mode (without GUI):

```
chrome_options = Options()
chrome_options.add_argument("--headless")
driver = webdriver.Chrome(options=chrome_options)
```

Browser Maximize and Minimize Control the browser window size:

```
# Maximize the browser window
driver.maximize_window()

# Minimize the browser window
driver.minimize_window()
```

Browser Capabilities Check the browser's capabilities and version:

```
browser_name = driver.capabilities['browserName']
browser_version = driver.capabilities['browserVersion']
```

Event Listeners You can use JavaScript to add event listeners to elements:

```
element = driver.find_element(By.ID, "element_id")
driver.execute_script("arguments[0].addEventListener('click', function(){alert('Element
clicked!');});", element)
element.click()  # Will trigger the event listener
```

Working with Multiple Windows Managing multiple windows or tabs:

```
# Get the handles of all open windows
window_handles = driver.window_handles

# Switch to a specific window
driver.switch_to.window(window_handles[1])
```

Remote WebDriver Connect to a remote WebDriver:

```
from selenium import webdriver
from selenium.webdriver.common.desired_capabilities import DesiredCapabilities

# Specify the remote WebDriver URL
driver = webdriver.Remote(
   command_executor='http://remote-webdriver-url:4444/wd/hub',
   desired_capabilities=DesiredCapabilities.CHROME
)
```

Mobile Emulation Emulate mobile devices:

```
mobile_emulation = {
   "deviceName": "iPhone 6"
}
chrome_options = Options()
chrome_options.add_experimental_option("mobileEmulation", mobile_emulation)
driver = webdriver.Chrome(chrome_options=chrome_options)
```