Opening and Closing Browsers

Python

```python
# Import necessary libraries
from selenium import webdriver
from selenium.webdriver.chrome.service import Service
from webdriver_manager.chrome import ChromeDriverManager
from selenium.webdriver.firefox.service import Service as FirefoxService
from webdriver_manager.firefox import GeckoDriverManager
import time

# Function to open and close Chrome browser
def open_close_chrome():
    # Set up Chrome driver
    driver = webdriver.Chrome(service=Service(ChromeDriverManager().install()))

    # Open Google homepage
    driver.get("https://www.google.com")

    # Wait for 5 seconds
    time.sleep(5)

    # Close the browser
    driver.quit()

# Function to open and close Firefox browser
def open_close_firefox():
    # Set up Firefox driver
    driver = webdriver.Firefox(service=FirefoxService(GeckoDriverManager().install()))

    # Open Google homepage
    driver.get("https://www.google.com")

    # Wait for 5 seconds
    time.sleep(5)

    # Close the browser
    driver.quit()

# Call the functions
open_close_chrome()
open_close_firefox()
```

Element Interaction

```python
from selenium.webdriver.chrome.service import Service
from webdriver_manager.chrome import ChromeDriverManager
from selenium.webdriver.common.by import By
from selenium.webdriver.common.keys import Keys
from selenium.webdriver.support.ui import Select
import time

# Set up Chrome driver
driver = webdriver.Chrome(service=Service(ChromeDriverManager().install()))

# Open Google homepage
driver.get("https://www.google.com")

# Element Interaction Examples

# 1. Clicking a button
search_button = driver.find_element(By.NAME, "btnK")
search_button.click()

# 2. Sending keys to input field
search_input = driver.find_element(By.NAME, "q")
search_input.send_keys("Selenium")

# 3. Selecting dropdown option
# Note: Google doesn't have a dropdown, so this example uses a different webpage
driver.get("https://www.wikipedia.org/")
language_select = Select(driver.find_element(By.ID, "searchLanguage"))
language_select.select_by_visible_text("Deutsch")

# 4. Checking checkbox (Google doesn't have checkboxes, using a different webpage)
driver.get("https://www.w3schools.com/tags/tryit.asp?filename=tryhtml_input_checkbox")
checkbox = driver.find_element(By.XPATH, "//input[@value='vehicle1']")
checkbox.click()

# 5. Clearing input field
driver.get("https://www.google.com")
search_input = driver.find_element(By.NAME, "q")
search_input.send_keys("Selenium")
search_input.clear()

# Wait for 5 seconds
time.sleep(5)

# Close the browser
driver.quit()
```

Navigation

```python
# Import necessary libraries
from selenium import webdriver
from selenium.webdriver.chrome.service import Service
from webdriver_manager.chrome import ChromeDriverManager
import time

# Set up Chrome driver
driver = webdriver.Chrome(service=Service(ChromeDriverManager().install()))

# Navigation Examples

# 1. Get (navigate to a URL)
driver.get("https://www.google.com")

# 2. Current URL
current_url = driver.current_url
print("Current URL:", current_url)

# 3. Title
title = driver.title
print("Title:", title)

# 4. Back
driver.get("https://www.wikipedia.org/")
driver.back()

# 5. Forward
driver.forward()

# 6. Refresh
driver.refresh()

# Navigation with Links

# 1. Clicking a link
driver.get("https://www.google.com")
driver.find_element(By.LINK_TEXT, "About").click()

# 2. Clicking a partial link
driver.get("https://www.wikipedia.org/")
driver.find_element(By.PARTIAL_LINK_TEXT, "Contact").click()

# Wait for 5 seconds
time.sleep(5)

# Close the browser
driver.quit()
```

Window and Frame Handling

```python
# Import necessary libraries
from selenium import webdriver
from selenium.webdriver.chrome.service import Service
from webdriver_manager.chrome import ChromeDriverManager
from selenium.webdriver.common.by import By
import time

# Set up Chrome driver
driver = webdriver.Chrome(service=Service(ChromeDriverManager().install()))

# Window Handling

# 1. Get window handle
driver.get("https://www.google.com")
window_handle = driver.current_window_handle
print("Current Window Handle:", window_handle)

# 2. Switch to new window
driver.get("https://www.wikipedia.org/")
driver.switch_to.new_window('tab')  # or 'window'
driver.get("https://www.github.com/")

# 3. Get window handles
window_handles = driver.window_handles
print("Window Handles:", window_handles)

# 4. Switch to specific window
driver.switch_to.window(window_handles[0])

# Frame Handling

# 1. Switch to frame by ID
driver.get("https://www.w3schools.com/tags/tryit.asp?filename=tryhtml_iframe")
driver.switch_to.frame("iframeResult")

# 2. Switch to frame by index
driver.switch_to.frame(0)

# 3. Switch to frame by WebElement
iframe = driver.find_element(By.XPATH, "//iframe[@id='iframeResult']")
driver.switch_to.frame(iframe)

# 4. Switch to default content
driver.switch_to.default_content()
```

Browser Management

```python
from selenium import webdriver
from selenium.webdriver.chrome.service import Service
from webdriver_manager.chrome import ChromeDriverManager
from selenium.webdriver.common.by import By

# Set up Chrome driver
driver = webdriver.Chrome(service=Service(ChromeDriverManager().install()))

# Navigate to URL
driver.get("https://www.google.com")

# Get current URL and title
print(driver.current_url)
print(driver.title)

# Go back and forward
driver.back()
driver.forward()

# Refresh page
driver.refresh()

# Get window handle
window_handle = driver.current_window_handle
print(window_handle)

# Switch to new window
driver.switch_to.new_window('tab')

# Get cookies
cookies = driver.get_cookies()
print(cookies)

# Save screenshot
driver.save_screenshot("screenshot.png")

# Close browser
driver.quit()
```

Mouse And KeyBoard Actions

Python

```python
from selenium import webdriver
from selenium.webdriver.chrome.service import Service
from webdriver_manager.chrome import ChromeDriverManager
from selenium.webdriver import ActionChains
from selenium.webdriver.common.by import By

# Set up Chrome driver
driver = webdriver.Chrome(service=Service(ChromeDriverManager().install()))

# Navigate to URL
driver.get("https://www.google.com")

# Create ActionChains object
actions = ActionChains(driver)

# Mouse actions
element = driver.find_element(By.ID, "some_id")

actions.move_to_element(element).perform()  # Move to element
actions.click(element).perform()  # Click element
actions.double_click(element).perform()  # Double-click element
actions.context_click(element).perform()  # Right-click element
actions.drag_and_drop(element, target).perform()  # Drag and drop
```

Explicit Wait

~~~python
# Define wait time (10 seconds)
wait_time = 10

# Explicit Wait examples

# 1. Wait for element to be visible
element_visible = WebDriverWait(driver, wait_time).until(
    EC.visibility_of_element_located((By.ID, "some_id"))
)

# 2. Wait for element to be clickable
element_clickable = WebDriverWait(driver, wait_time).until(
    EC.element_to_be_clickable((By.ID, "some_id"))
)

# 3. Wait for element to be present
element_present = WebDriverWait(driver, wait_time).until(
    EC.presence_of_element_located((By.ID, "some_id"))
)

# 4. Wait for title to contain specific text
title_contains = WebDriverWait(driver, wait_time).until(
    EC.title_contains("Google")
)

# 5. Wait for alert to be present
alert_present = WebDriverWait(driver, wait_time).until(
    EC.alert_is_present()
)

# Close browser
driver.quit()
```
~~~

Implicit Wait

~~~~
from selenium import webdriver
from selenium.webdriver.chrome.service import Service
from webdriver_manager.chrome import ChromeDriverManager

# Set up Chrome driver
driver = webdriver.Chrome(service=Service(ChromeDriverManager().install()))

# Set implicit wait time (10 seconds)
driver.implicitly_wait(10)

# Navigate to URL
driver.get("https://www.google.com")

# Search for element
element = driver.find_element(By.ID, "some_id")
```


**Important Considerations:**


*   **Implicit Wait vs. Explicit Wait**: Implicit Wait is global and less fle
*   **Performance Impact**: Implicit Wait can slow down test execution, espec
*   **Compatibility Issues**: Some Selenium versions may have issues with Imp


**Best Practices:**


*   Use Implicit Wait sparingly, as it can mask underlying performance issues
*   Set reasonable wait times to avoid unnecessary delays.
*   Prefer Explicit Wait for more control and flexibility.


**Common Exceptions:**


*   `TimeoutException`: Raised when wait time exceeds.
*   `NoSuchElementException`: Raised when element not found.
~~~~

Alerts

~~~
from selenium import webdriver
from selenium.webdriver.chrome.service import Service
from webdriver_manager.chrome import ChromeDriverManager
from selenium.webdriver.common.alert import Alert
from selenium.webdriver.common.by import By

# Set up Chrome driver
driver = webdriver.Chrome(service=Service(ChromeDriverManager().install()))

# Navigate to URL
driver.get("https://www.example.com")

# Trigger alert
driver.find_element(By.ID, "alert-button").click()

# Switch to alert
alert = Alert(driver)

# Handle alert
alert.accept()  # OK
alert.dismiss()  # Cancel
alert.send_keys("Text")  # Send keys to prompt alert
```"


**Alert Methods:**


*   `accept()`: Accept/OK alert.
*   `dismiss()`: Dismiss/Cancel alert.
*   `send_keys(keys)`: Send keys to prompt alert.
*   `text`: Get alert text.


**Switching to Alert:**


*   `switch_to.alert`: Switch to alert.


**Common Exceptions:**


*   `NoAlertPresentException`: Raised when no alert is present.
~~~

Getting Page Information

4. Retrieve page information:
    - Page Title: `driver.title`
    - Page URL: `driver.current_url`
    - Page Source: `driver.page_source`
    - Page Content: `driver.find_element(By.TAG_NAME, 'body').text`
    - Cookies: `driver.get_cookies()`
    - Window Size: `driver.get_window_size()`
    - Window Position: `driver.get_window_position()`
    - Screen Resolution: `driver.get_window_rect()`
    - Page Loading Status: `driver.execute_script("return document.readyState")`
    - HTTP Response Status Code: `driver.execute_script("return XMLHttpRequest().status")`

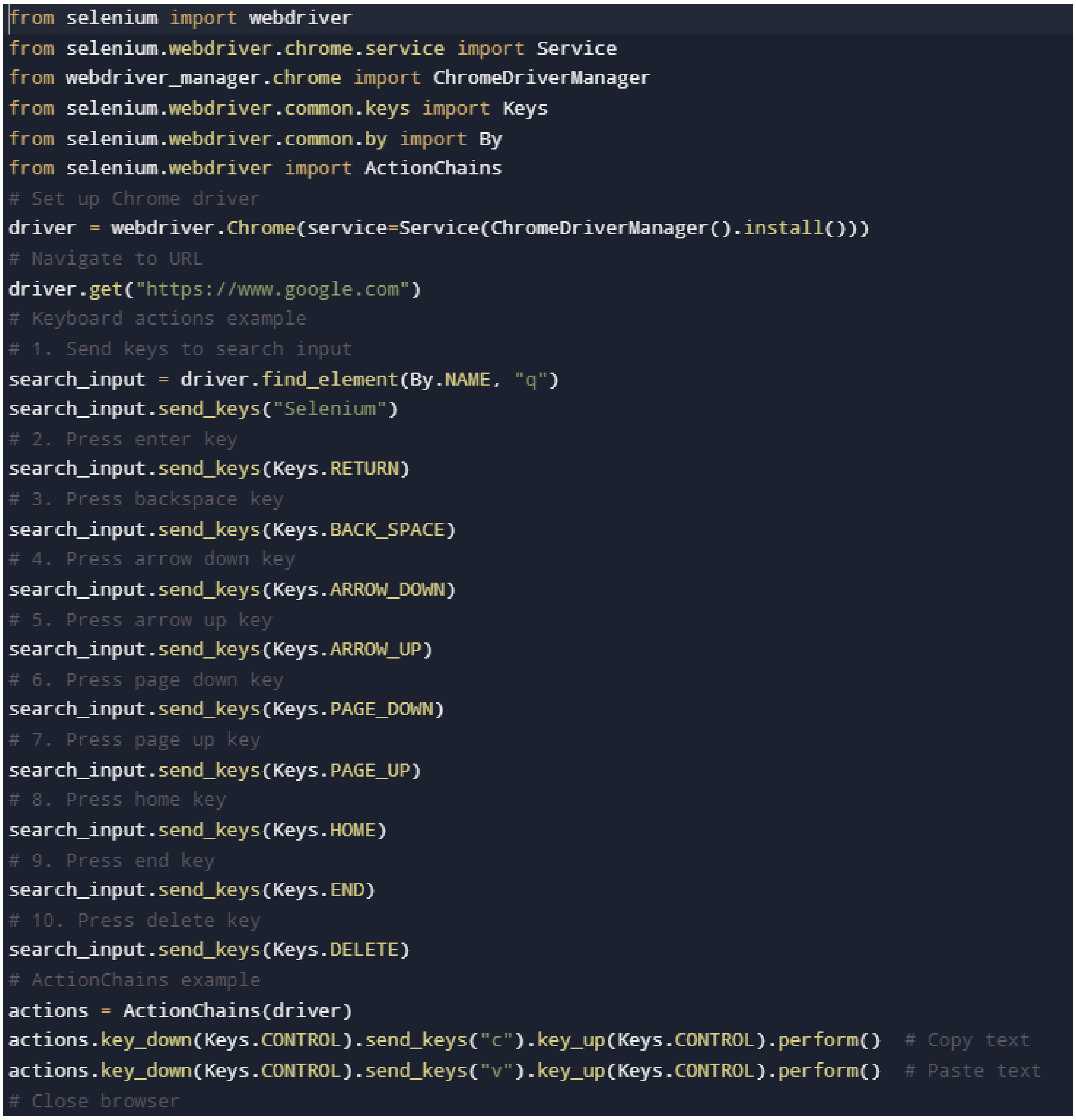
Element information

5. Retrieve element information:
    - Element Tag Name: `element.tag_name`
    - Element Size: `element.size`
    - Element Location: `element.location`
    - Element Text: `element.text`
    - Element Attribute: `element.get_attribute("attribute_name")`
    - Element CSS Property: `element.value_of_css_property("property_name")`
    - Element Rect: `element.rect`
    - Element ID: `element.id`
    - Element Enabled Status: `element.is_enabled()`
    - Element Selected Status: `element.is_selected()`

Menu Handling

```python
from selenium import webdriver
from selenium.webdriver.chrome.service import Service
from webdriver_manager.chrome import ChromeDriverManager
from selenium.webdriver.common.by import By
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC

# Set up Chrome driver
driver = webdriver.Chrome(service=Service(ChromeDriverManager().install()))

# Navigate to URL
driver.get("https://www.example.com")

# Menu handling example

# 1. Click on dropdown menu
dropdown_menu = WebDriverWait(driver, 10).until(
    EC.element_to_be_clickable((By.ID, "dropdown-menu"))
)
dropdown_menu.click()

# 2. Select menu item
menu_item = WebDriverWait(driver, 10).until(
    EC.element_to_be_clickable((By.XPATH, "//a[@id='menu-item']"))
)
menu_item.click()

# 3. Hover over menu item
menu_item_hover = WebDriverWait(driver, 10).until(
    EC.presence_of_element_located((By.XPATH, "//a[@id='menu-item-hover']"))
)
hover_action = webdriver.ActionChains(driver)
hover_action.move_to_element(menu_item_hover).perform()

# 4. Right-click on menu item
menu_item_right_click = WebDriverWait(driver, 10).until(
    EC.presence_of_element_located((By.XPATH, "//a[@id='menu-item-right-click']"))
)
right_click_action = webdriver.ActionChains(driver)
right_click_action.context_click(menu_item_right_click).perform()

# Close browser
driver.quit()
```

Keyboard Actions

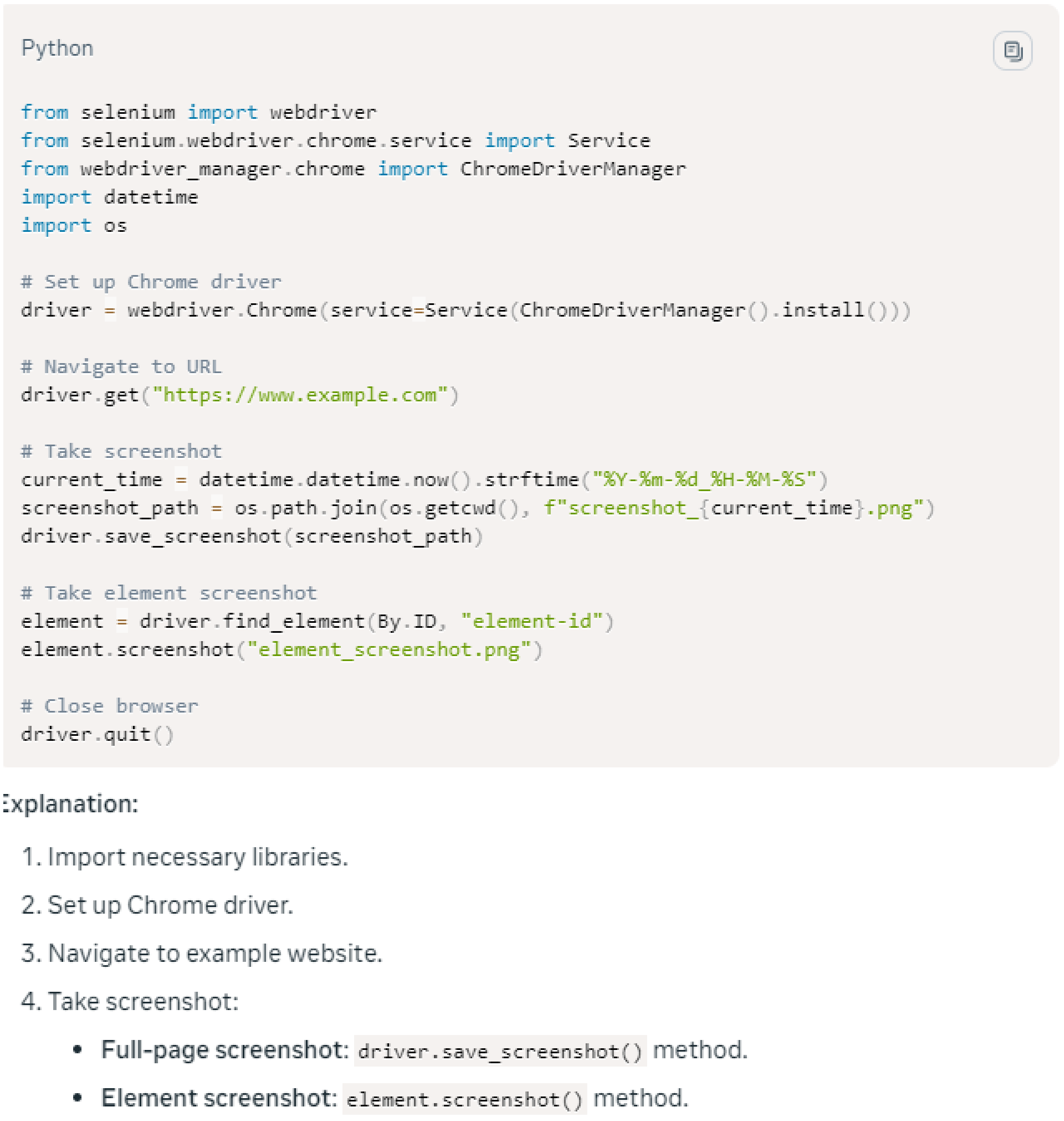


```python
from selenium import webdriver
from selenium.webdriver.chrome.service import Service
from webdriver_manager.chrome import ChromeDriverManager
from selenium.webdriver.common.keys import Keys
from selenium.webdriver.common.by import By
from selenium.webdriver import ActionChains
# Set up Chrome driver
driver = webdriver.Chrome(service=Service(ChromeDriverManager().install()))
# Navigate to URL
driver.get("https://www.google.com")
# Keyboard actions example
# 1. Send keys to search input
search_input = driver.find_element(By.NAME, "q")
search_input.send_keys("Selenium")
# 2. Press enter key
search_input.send_keys(Keys.RETURN)
# 3. Press backspace key
search_input.send_keys(Keys.BACK_SPACE)
# 4. Press arrow down key
search_input.send_keys(Keys.ARROW_DOWN)
# 5. Press arrow up key
search_input.send_keys(Keys.ARROW_UP)
# 6. Press page down key
search_input.send_keys(Keys.PAGE_DOWN)
# 7. Press page up key
search_input.send_keys(Keys.PAGE_UP)
# 8. Press home key
search_input.send_keys(Keys.HOME)
# 9. Press end key
search_input.send_keys(Keys.END)
# 10. Press delete key
search_input.send_keys(Keys.DELETE)
# ActionChains example
actions = ActionChains(driver)
actions.key_down(Keys.CONTROL).send_keys("c").key_up(Keys.CONTROL).perform()  # Copy text
actions.key_down(Keys.CONTROL).send_keys("v").key_up(Keys.CONTROL).perform()  # Paste text
# Close browser
```

Mouse Actions

```python
from selenium import webdriver
from selenium.webdriver.chrome.service import Service
from webdriver_manager.chrome import ChromeDriverManager
from selenium.webdriver.common.by import By
from selenium.webdriver import ActionChains

# Set up Chrome driver
driver = webdriver.Chrome(service=Service(ChromeDriverManager().install()))

# Navigate to URL
driver.get("https://www.example.com")

# Mouse actions example

# 1. Move to element
element = driver.find_element(By.ID, "element-id")
actions = ActionChains(driver)
actions.move_to_element(element).perform()

# 2. Click element
actions.click(element).perform()

# 3. Double-click element
actions.double_click(element).perform()

# 4. Right-click element
actions.context_click(element).perform()

# 5. Drag and drop element
source = driver.find_element(By.ID, "source-id")
target = driver.find_element(By.ID, "target-id")
actions.drag_and_drop(source, target).perform()

# 6. Click and hold element
actions.click_and_hold(element).perform()

# 7. Release element
actions.release(element).perform()

# 8. Move by offset
actions.move_by_offset(10, 20).perform()

# Close browser
driver.quit()
```

Taking ScreenShots


Python

```python
from selenium import webdriver
from selenium.webdriver.chrome.service import Service
from webdriver_manager.chrome import ChromeDriverManager
import datetime
import os

# Set up Chrome driver
driver = webdriver.Chrome(service=Service(ChromeDriverManager().install()))

# Navigate to URL
driver.get("https://www.example.com")

# Take screenshot
current_time = datetime.datetime.now().strftime("%Y-%m-%d_%H-%M-%S")
screenshot_path = os.path.join(os.getcwd(), f"screenshot_{current_time}.png")
driver.save_screenshot(screenshot_path)

# Take element screenshot
element = driver.find_element(By.ID, "element-id")
element.screenshot("element_screenshot.png")

# Close browser
driver.quit()
```

xplanation:

1. Import necessary libraries.
2. Set up Chrome driver.
3. Navigate to example website.
4. Take screenshot:
    - **Full-page screenshot:** `driver.save_screenshot()` method.
    - **Element screenshot:** `element.screenshot()` method.

Switching to IFrame

Python

```python
from selenium import webdriver
from selenium.webdriver.chrome.service import Service
from webdriver_manager.chrome import ChromeDriverManager
from selenium.webdriver.common.by import By

# Set up Chrome driver
driver = webdriver.Chrome(service=Service(ChromeDriverManager().install()))

# Navigate to URL
driver.get("https://www.example.com")

# Switch to iframe
iframe = driver.find_element(By.ID, "iframe-id")
driver.switch_to.frame(iframe)

# Perform actions within iframe
driver.find_element(By.ID, "element-id").click()

# Switch back to default content
driver.switch_to.default_content()

# Close browser
driver.quit()
```

Explanation:

1. Import necessary libraries.
2. Set up Chrome driver.
3. Navigate to example website.
4. Switch to iframe:
    - **Find iframe element:** `driver.find_element()` method.
    - **Switch to iframe:** `driver.switch_to.frame()` method.

Scrolling

```python
from selenium import webdriver
from selenium.webdriver.chrome.service import Service
from webdriver_manager.chrome import ChromeDriverManager
from selenium.webdriver.common.by import By

# Set up Chrome driver
driver = webdriver.Chrome(service=Service(ChromeDriverManager().install()))

# Navigate to URL
driver.get("https://www.example.com")

# Scroll to bottom of page
driver.execute_script("window.scrollTo(0, document.body.scrollHeight);")

# Scroll to top of page
driver.execute_script("window.scrollTo(0, 0);")

# Scroll to specific element
element = driver.find_element(By.ID, "element-id")
driver.execute_script("arguments[0].scrollIntoView();", element)

# Scroll by pixels
driver.execute_script("window.scrollBy(0, 500);")

# Scroll to specific coordinates
driver.execute_script("window.scrollTo(0, 1000);")

# Close browser
driver.quit()
```

Cookie Management

```python
from selenium import webdriver
from selenium.webdriver.chrome.service import Service
from webdriver_manager.chrome import ChromeDriverManager

# Set up Chrome driver
driver = webdriver.Chrome(service=Service(ChromeDriverManager().install()))

# Navigate to URL
driver.get("https://www.example.com")

# Get cookies
cookies = driver.get_cookies()
print(cookies)

# Add cookie
driver.add_cookie({"name": "cookie-name", "value": "cookie-value"})

# Delete cookie
driver.delete_cookie("cookie-name")

# Delete all cookies
driver.delete_all_cookies()

# Close browser
driver.quit()
```

4. Cookie Management

- `Page.GetCookiesAsync`: Get cookies
- `Page.DeleteCookiesAsync`: Delete cookies
- `Page.SetCookiesAsync`: Set cookies

C#

```csharp
var cookies = await page.GetCookiesAsync();
await page.DeleteCookiesAsync();
await page.SetCookiesAsync(new[] { new Cookie { Name = "myCookie", Value = "myValue" }
```

Handling Shadow Dom Elements

8. Handling Shadow DOM Elements

- `Page.QuerySelectorAsync`: Query selector inside shadow root
- `Element.ShadowRootAsync`: Get shadow root element

```C#
var shadowHost = await page.QuerySelectorAsync("#my-shadow-host");
var shadowRoot = await shadowHost.ShadowRootAsync();
var shadowElement = await shadowRoot.QuerySelectorAsync("#my-shadow-element");
```

```python
from selenium import webdriver
from selenium.webdriver.chrome.service import Service
from webdriver_manager.chrome import ChromeDriverManager
from selenium.webdriver.common.by import By

# Set up Chrome driver
driver = webdriver.Chrome(service=Service(ChromeDriverManager().install()))

# Navigate to URL
driver.get("https://www.example.com")

# Get Shadow DOM element
shadow_host = driver.find_element(By.CSS_SELECTOR, "#shadow-host")
shadow_root = driver.execute_script("return arguments[0].shadowRoot", shadow_host)

# Get element within Shadow DOM
element = shadow_root.find_element(By.CSS_SELECTOR, "#shadow-element")

# Perform action on element
element.click()

# Close browser
driver.quit()
```

Explanation:

1. Import necessary libraries.
2. Set up Chrome driver.
3. Navigate to example website.
4. Handling Shadow DOM elements:
   - Get Shadow DOM host: `driver.find_element()` method.
   - Get Shadow DOM root: `driver.execute_script()` method.
   - Get element within Shadow DOM: `shadow_root.find_element()` method.
   - Perform action on element: `element.click()` method.