Selenium with Python - Complete Cheat Sheet

Installation & Setup

```
pip install selenium
pip install webdriver-manager  # Auto-manage drivers
```

WebDriver Initialization

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

# Chrome (Recommended)
driver = webdriver.Chrome()

# With options
from selenium.webdriver.chrome.options import Options
options = Options()
options.add_argument("--headless")  # Run in background
options.add_argument("--no-sandbox")
options.add_argument("--disable-dev-shm-usage")
driver = webdriver.Chrome(options=options)

# Auto-manage driver
driver = webdriver.Chrome(service=Service(ChromeDriverManager().install()))

# Other browsers
driver = webdriver.Firefox()
driver = webdriver.Edge()
driver = webdriver.Safari()
```

Essential Imports

```python
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.common.keys import Keys
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
from selenium.webdriver.support.ui import Select
from selenium.webdriver.common.action_chains import ActionChains
from selenium.common.exceptions import *
```

Element Locators

Find Single Element

```
# By ID (Fastest)
element = driver.find_element(By.ID, "element_id")

# By Name
element = driver.find_element(By.NAME, "username")

# By Class Name
element = driver.find_element(By.CLASS_NAME, "btn-primary")

# By Tag Name
element = driver.find_element(By.TAG_NAME, "input")

# By Link Text
element = driver.find_element(By.LINK_TEXT, "Click Here")

# By Partial Link Text
element = driver.find_element(By.PARTIAL_LINK_TEXT, "Click")

# By XPath
element = driver.find_element(By.XPATH, "//input[@id='username']")

# By CSS Selector
element = driver.find_element(By.CSS_SELECTOR, "#username")
```

Find Multiple Elements

```
elements = driver.find_elements(By.CLASS_NAME, "item")
elements = driver.find_elements(By.XPATH, "//div[@class='product']")
```

Waits (Critical for Stable Tests)

Implicit Wait (Global)

```
driver.implicitly_wait(10)  # Seconds
```

Explicit Wait (Recommended)

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

wait = WebDriverWait(driver, 10)

# Wait for element to be clickable
element = wait.until(EC.element_to_be_clickable((By.ID, "submit_btn")))
```

```python
# Wait for element to be present
element = wait.until(EC.presence_of_element_located((By.ID, "result")))

# Wait for element to be visible
element = wait.until(EC.visibility_of_element_located((By.ID, "popup")))

# Wait for text to be present
wait.until(EC.text_to_be_present_in_element((By.ID, "status"), "Success"))
```

Common Expected Conditions

```python
EC.title_is("Page Title")
EC.title_contains("Partial Title")
EC.presence_of_element_located((By.ID, "myId"))
EC.visibility_of_element_located((By.ID, "myId"))
EC.visibility_of(element)
EC.element_to_be_clickable((By.ID, "myId"))
EC.staleness_of(element)
EC.element_to_be_selected(element)
EC.text_to_be_present_in_element((By.ID, "myId"), "text")
EC.alert_is_present()
```

Element Interactions

Basic Actions

```python
# Click
element.click()

# Send text
element.send_keys("Hello World")
element.send_keys(Keys.ENTER)

# Clear text
element.clear()

# Get text
text = element.text

# Get attribute
value = element.get_attribute("value")
href = element.get_attribute("href")

# Check if element is displayed/enabled/selected
element.is_displayed()
element.is_enabled()
element.is_selected()
```

Special Keys

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

element.send_keys(Keys.ENTER)
element.send_keys(Keys.TAB)
element.send_keys(Keys.SPACE)
element.send_keys(Keys.CONTROL, 'a')  # Ctrl+A
element.send_keys(Keys.CONTROL, 'c')  # Ctrl+C
element.send_keys(Keys.ESCAPE)
element.send_keys(Keys.F5)  # Refresh
```

Advanced Interactions

ActionChains

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

actions = ActionChains(driver)

# Hover over element
actions.move_to_element(element).perform()

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

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

# Drag and drop
actions.drag_and_drop(source_element, target_element).perform()

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

# Chain multiple actions
actions.move_to_element(menu).click().move_to_element(submenu).click().perform()
```

Dropdowns

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

# Standard HTML select dropdown
dropdown = Select(driver.find_element(By.ID, "dropdown_id"))
```

```python
# Select by visible text
dropdown.select_by_visible_text("Option 1")

# Select by value
dropdown.select_by_value("option1")

# Select by index
dropdown.select_by_index(0)

# Get all options
all_options = dropdown.options
for option in all_options:
    print(option.text)

# Get selected option
selected_option = dropdown.first_selected_option
print(selected_option.text)

# Deselect (for multi-select)
dropdown.deselect_by_visible_text("Option 1")
dropdown.deselect_all()
```

Window/Tab Management

```python
# Get current window handle
current_window = driver.current_window_handle

# Get all window handles
all_windows = driver.window_handles

# Switch to new window
for window in all_windows:
    if window != current_window:
        driver.switch_to.window(window)
        break

# Open new tab
driver.execute_script("window.open('');")
driver.switch_to.window(driver.window_handles[-1])

# Close current window
driver.close()

# Switch back to main window
driver.switch_to.window(current_window)
```

Alerts & Popups

```python
# Switch to alert
alert = driver.switch_to.alert

# Get alert text
alert_text = alert.text

# Accept alert (OK button)
alert.accept()

# Dismiss alert (Cancel button)
alert.dismiss()

# Send text to prompt
alert.send_keys("Input text")

# Wait for alert
wait.until(EC.alert_is_present())
```

Frames & iFrames

```python
# Switch to frame by name or id
driver.switch_to.frame("frame_name")

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

# Switch to frame by WebElement
frame_element = driver.find_element(By.TAG_NAME, "iframe")
driver.switch_to.frame(frame_element)

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

# Switch to parent frame
driver.switch_to.parent_frame()
```

Browser Navigation

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

# Get current URL
current_url = driver.current_url
```

```
# Get page title
title = driver.title

# Get page source
page_source = driver.page_source

# Browser navigation
driver.back()
driver.forward()
driver.refresh()

# Set window size
driver.set_window_size(1920, 1080)

# Maximize window
driver.maximize_window()

# Minimize window
driver.minimize_window()

# Fullscreen
driver.fullscreen_window()
```

Screenshots & Debugging

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

# Get screenshot as binary data
screenshot = driver.get_screenshot_as_png()

# Element screenshot
element.screenshot("element.png")

# Execute JavaScript
result = driver.execute_script("return document.title;")
driver.execute_script("window.scrollTo(0, document.body.scrollHeight);")

# Scroll to element
driver.execute_script("arguments[0].scrollIntoView();", element)
```

File Upload

```
# For input type="file"
file_input = driver.find_element(By.XPATH, "//input[@type='file']")
file_input.send_keys("/path/to/file.pdf")
```

```
# Multiple files
file_input.send_keys("/path/file1.pdf\n/path/file2.pdf")
```

Common XPath Patterns

```
# Text contains
"//div[contains(text(), 'Hello')]"

# Attribute contains
"//input[contains(@class, 'form-control')]"

# Following sibling
"//label[text()='Username']/following-sibling::input"

# Parent element
"//input[@id='username']/parent::div"

# Index-based
"//table/tr[2]/td[3]"  # 2nd row, 3rd column

# Multiple conditions
"//div[@class='product' and contains(text(), 'iPhone')]"

# Starts with
"//div[starts-with(@id, 'product_')]"
```

CSS Selector Patterns

```
# ID
"#element_id"

# Class
".class_name"

# Attribute
"input[name='username']"
"div[data-id='123']"

# Contains attribute
"input[class*='form']"

# Starts with
"div[id^='product_']"

# Ends with
```

```
"input[name$='_email']"

# Child combinator
"div > input"

# Descendant combinator
"form input"

# Adjacent sibling
"label + input"

# nth-child
"tr:nth-child(2)"
"td:nth-child(odd)"
```

Exception Handling

```python
from selenium.common.exceptions import *

try:
    element = driver.find_element(By.ID, "element_id")
    element.click()
except NoSuchElementException:
    print("Element not found")
except ElementClickInterceptedException:
    print("Element click intercepted")
except TimeoutException:
    print("Operation timed out")
except WebDriverException as e:
    print(f"WebDriver error: {e}")
finally:
    driver.quit()
```

Page Object Model Pattern

```python
class LoginPage:
    def __init__(self, driver):
        self.driver = driver
        self.username_field = (By.ID, "username")
        self.password_field = (By.ID, "password")
        self.login_button = (By.ID, "login_btn")

    def enter_username(self, username):
        self.driver.find_element(*self.username_field).send_keys(username)

    def enter_password(self, password):
        self.driver.find_element(*self.password_field).send_keys(password)

    def click_login(self):
        self.driver.find_element(*self.login_button).click()

    def login(self, username, password):
        self.enter_username(username)
        self.enter_password(password)
        self.click_login()

# Usage
login_page = LoginPage(driver)
login_page.login("testuser", "password123")
```

Best Practices

1. Always Use Explicit Waits

```
#  Bad
time.sleep(5)

#  Good
wait.until(EC.element_to_be_clickable((By.ID, "submit")))
```

2. Prefer ID and Name Locators

```
#  Best
driver.find_element(By.ID, "username")

#  Good
driver.find_element(By.NAME, "username")

#  Use when necessary
driver.find_element(By.XPATH, "//input[@id='username']")
```

3. Always Close Driver

```
try:
    # Test code here
    pass
finally:
    driver.quit()
```

4. Use Context Manager (Python 3.7+)

```
with webdriver.Chrome() as driver:
    driver.get("https://example.com")
    # Test code here
# Driver automatically quits
```

Quick Test Template

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

class TestExample(unittest.TestCase):
    def setUp(self):
        self.driver = webdriver.Chrome()
        self.driver.maximize_window()
        self.wait = WebDriverWait(self.driver, 10)

    def test_login(self):
        driver = self.driver
        driver.get("https://example.com/login")

        # Login steps
        username = self.wait.until(EC.presence_of_element_located((By.ID, "username")))
        username.send_keys("testuser")

        password = driver.find_element(By.ID, "password")
        password.send_keys("password123")

        login_btn = driver.find_element(By.ID, "login_btn")
        login_btn.click()

        # Verification
        success_msg = self.wait.until(EC.visibility_of_element_located((By.CLASS_NAME, "succ
        self.assertIn("Welcome", success_msg.text)

    def tearDown(self):
        self.driver.quit()

if __name__ == "__main__":
    unittest.main()
```

Mobile Testing (Chrome DevTools)

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

mobile_emulation = {
    "deviceMetrics": {"width": 375, "height": 667, "pixelRatio": 2.0},
    "userAgent": "Mozilla/5.0 (iPhone; CPU iPhone OS 10_3 like Mac OS X) AppleWebKit/602.1.5
}

options = Options()
options.add_experimental_option("mobileEmulation", mobile_emulation)
driver = webdriver.Chrome(options=options)
```

This cheat sheet covers 90% of common Selenium operations. Keep it handy for quick reference during development and testing!

How to Download This Cheat Sheet

To save this as a .md file: 1. Click the **copy icon** ( ) in the top-right corner of this artifact 2. Open any text editor (Notepad, VS Code, Sublime, etc.) 3. Paste the content 4. Save the file as `selenium_python_cheatsheet.md`

Alternative method: - Click the three dots menu ( ) in the top-right corner - Select "Copy" or use your browser's copy function - Save in your preferred markdown editor

You can then view this file in any markdown viewer or IDE that supports markdown preview!