SELENIUM WEBDRIVER クイックリファレンス

ブラウザ自動化、要素操作、待機、アサーション

セットアップ

インストール

```
pip install selenium webdriver-manager
# webdriver-manager がブラウザドライバを自動ダウンロード
```

基本的なドライバセットアップ

```
from selenium import webdriver
from selenium.webdriver.chrome.service import Service
from webdriver_manager.chrome import ChromeDriverManager
driver = webdriver.Chrome(
    service=Service(ChromeDriverManager().install()))
```

ヘッドレスモード

```
options = webdriver.ChromeOptions()
options.add_argument("--headless=new")
options.add_argument("--no-sandbox")
driver = webdriver.Chrome(options=options)
```

対応ブラウザ

`webdriver.Chrome()`	Google Chrome / Chromium
`webdriver.Firefox()`	Mozilla Firefox（GeckoDriver）
`webdriver.Edge()`	Microsoft Edge（Chromium ベース）
`webdriver.Safari()`	Apple Safari（macOS のみ）

ブラウザとナビゲーション

ナビゲーション

```
driver.get("https://example.com")
driver.back()                # ブラウザの戻る
driver.forward()             # ブラウザの進む
driver.refresh()             # ページを再読み込み
```

ブラウザのプロパティ

`driver.title`	現在のページタイトル
`driver.current_url`	現在のページ URL
`driver.page_source`	ページの HTML ソース全体
`driver.get_cookies()`	すべてのクッキーを一覧表示

ウィンドウ管理

```
driver.set_window_size(1920, 1080)
driver.maximize_window()
driver.minimize_window()
driver.quit()                # すべてのウィンドウを閉じてセッション終了
```

要素の検索

ロケータ戦略

```
from selenium.webdriver.common.by import By
driver.find_element(By.ID, "login-btn")
driver.find_element(By.CLASS_NAME, "nav-item")
driver.find_element(By.CSS_SELECTOR, "div.card > h2")
driver.find_element(By.XPATH, "//input[@name='q']")
```

By 戦略

`By.ID`	要素の id 属性でマッチ
`By.NAME`	要素の name 属性でマッチ
`By.CLASS_NAME`	CSS クラスでマッチ（単一クラス）
`By.TAG_NAME`	HTML タグ名でマッチ
`By.CSS_SELECTOR`	CSS セレクタ（最も柔軟）
`By.XPATH`	XPath 式
`By.LINK_TEXT`	アンカーテキストの完全一致
`By.PARTIAL_LINK_TEXT`	アンカーテキストの部分一致

複数要素の検索

```
items = driver.find_elements(By.CSS_SELECTOR, "li.item")
for item in items:
    print(item.text)
# 見つからない場合は空のリストを返す（例外なし）
```

インタラクション

クリックと入力

```
elem = driver.find_element(By.ID, "search")
elem.clear()                 # 既存のテキストをクリア
elem.send_keys("selenium python")
elem.submit()                # 親フォームをサブミット
```

ドロップダウン

```
from selenium.webdriver.support.ui import Select
select = Select(driver.find_element(By.ID, "country"))
select.select_by_visible_text("Canada")
select.select_by_value("ca")
select.select_by_index(2)
```

要素のプロパティ

`.text`	表示テキストの内容
`.get_attribute('href')`	HTML 属性の値
`.is_displayed()`	要素が表示されている場合 True
`.is_enabled()`	要素が操作可能な場合 True
`.is_selected()`	チェックボックス・ラジオが選択されている場合 True
`.tag_name`	HTML タグ（例: 'input'、'div'）
`.value_of_css_property('color')`	計算済み CSS プロパティの値

待機

明示的な待機

```
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
elem = WebDriverWait(driver, 10).until(
    EC.presence_of_element_located((By.ID, "result")))
```

期待される条件

`presence_of_element_located`	要素が DOM に存在する
`visibility_of_element_located`	要素がページに表示されている
`element_to_be_clickable`	要素が表示されており操作可能
`text_to_be_present_in_element`	要素に期待するテキストが含まれる
`alert_is_present`	JavaScript アラートが表示されている
`staleness_of`	要素が DOM から除去された
`title_contains`	ページタイトルにテキストが含まれる

暗黙的な待機

```
driver.implicitly_wait(10)  # 秒単位、グローバルに適用
# 明示的な待機が推奨 — より精密な制御が可能
```

フレームとウィンドウ

フレーム

```
driver.switch_to.frame("frame-name")    # 名前またはIDで
driver.switch_to.frame(0)                # インデックスで
driver.switch_to.frame(elem)             # 要素で
driver.switch_to.default_content()       # メインに戻る
```

ウィンドウとタブ

```
original = driver.current_window_handle
driver.switch_to.new_window("tab")       # 新しいタブを開く
driver.switch_to.window(original)        # 元に戻る
driver.close()                           # 現在のタブを閉じる
```

アラート

```
alert = driver.switch_to.alert
print(alert.text)
alert.accept()    # OK をクリック
alert.dismiss()   # キャンセルをクリック
alert.send_keys("input text")
```

スクリーンショット

スクリーンショットの取得

```
driver.save_screenshot("page.png")       # ページ全体
elem = driver.find_element(By.ID, "chart")
elem.screenshot("chart.png")             # 単一要素
```

Base64 でのスクリーンショット

```
b64 = driver.get_screenshot_as_base64()
png = driver.get_screenshot_as_png()     # バイト列
```

アクション

アクションチェーン

```
from selenium.webdriver.common.action_chains import ActionChains
actions = ActionChains(driver)
actions.move_to_element(menu).click().perform()
```

キーボード操作

```
from selenium.webdriver.common.keys import Keys
elem.send_keys(Keys.ENTER)
elem.send_keys(Keys.CONTROL, "a")        # すべて選択
actions.key_down(Keys.SHIFT).click(elem).perform()
```

マウス操作

`.click(elem)`	要素をクリック
`.double_click(elem)`	要素をダブルクリック
`.context_click(elem)`	要素を右クリック
`.move_to_element(elem)`	要素にホバー
`.drag_and_drop(src, dst)`	ソースをドラッグしてデスティネーションにドロップ
`.click_and_hold(elem)`	マウスボタンを押し続ける
`.release()`	マウスボタンを離す

アサーション

よく使うアサーション（pytest）

```
assert "Dashboard" in driver.title
assert driver.find_element(By.ID, "msg").text == "Done"
assert driver.current_url.endswith("/home")
assert len(driver.find_elements(By.CSS_SELECTOR, "tr")) > 0
```

待機ベースのアサーション

```
WebDriverWait(driver, 5).until(              # 表示を待つ
    EC.visibility_of_element_located((By.ID, "success")))
WebDriverWait(driver, 5).until(              # 非表示を待つ
    EC.invisibility_of_element_located((By.ID, "spinner")))
```

JavaScript の実行

```
result = driver.execute_script("return document.title")
driver.execute_script(
    "arguments[0].scrollIntoView(true);", elem)
```

よく使うパターン

ページオブジェクトパターン

```
class LoginPage:
    URL = "/login"
    user_loc = (By.ID, "username")
    def login(self, drv, user, pwd):
        drv.find_element(*self.user_loc).send_keys(user)
```

コンテキストマネージャ

```
from selenium import webdriver
with webdriver.Chrome() as driver:
    driver.get("https://example.com")
    print(driver.title)
# driver.quit() が自動的に呼ばれる
```

リトライとクリーンアップ

```
try:
    driver.get("https://example.com")
    WebDriverWait(driver, 10).until(
        EC.element_to_be_clickable((By.ID, "btn")))
finally: driver.quit()
```