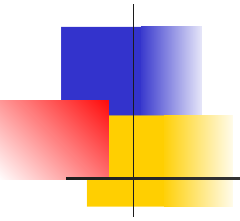


Exercise Using Selenium

"Scrape" URLs from a page of search results

Selenium

Browser automation.

Not just testing.

`https://selenium.dev/`

We will use Selenium WebDriver

- programmatically control a web browser

Selenium Example

Goal:

Use duckduckgo.com to find links to Kasertsart U.

Print the top 10 links.

Requires:

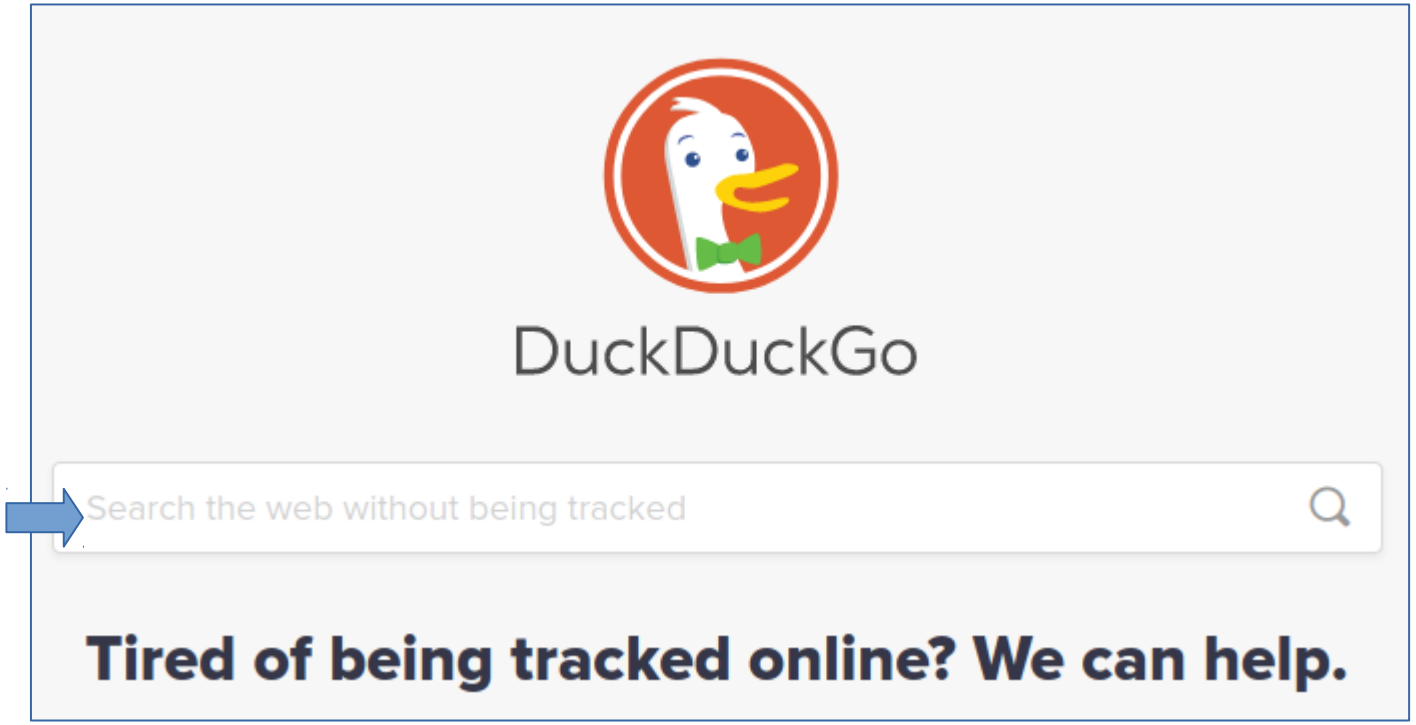
- Selenium WebDriver (`pip install selenium`)
- driver for Firefox browser (called "geckodriver")

  https://github.com/mozilla/geckodriver/releases
- you can use Chrome or Safari instead

Selenium: get a web page

```
from selenium import webdriver
from selenium.webdriver.common.keys import Keys
# browser: a WebDriver object
browser = webdriver.Firefox()
browser.implicitly_wait(5) # seconds


# get the duckduckgo search page
url = "https://duckduckgo.com"
browser.get( url )
```

Get the id of the search input box



Firefox: right-click in search box -> "Inspect Element".

You will see:

<input id='**search_form_input_homepage**' name="**q**"
type="text" ...>

find the id on page & send data

```
# Find the search box on page
# Selenium has many find_by_* commands

field_id = 'search_form_input_homepage'

input_field =
     browser.find_element_by_id(field_id)

input_field.send_keys("Kasetsart Univer")

input_field.send_keys(Keys.ENTER)

# Run It!

# the browser should display results
```

Inspect the Page & Identify Links

We need a way for Selenium to "find" the hyperlinks on the results page.

You can use:

* tag type ('a' tag)

* "id" of an element

* "class" of an element

* CSS selectors, or other identifying data

```
<div class="...">
  <a class="result__url js-result-extras-url"
   href="https://www.usnews.com/education/
   best-global-universities/kasetsart-university-xxx"
   ...>
```

Page Scraping

```
# get links from the results page
# the link URLs have class="result__url"

elements =
  browser.find_elements_by_class_name(
  'result__url')

print(f"Found {len(elements)} matches.")

# Each result is a WebElement object.
# WebElement contains attributes &
# other (child) WebElements.
# Show "href" attribute of first match

url = elements[0].get_attribute('href')
```

Page Scraping (2)

```
# Get the 'href' value

page_url =
  elements[0].get_attribute('href')

print("First result link is", page_url)

# What the heck -- Let's visit the page!

# element must be "clickable" to work

elements[0].click()


input("Press ENTER to go back to results")

browser.back()
```

Exercise: print first 10 URLs

1. Print the URLs of the first 10 matches on the DuckDuckGo search results page.

2. Make this code into a function that you can use to get (and return) the top-10 results for any search!

```
elements =
  browser.find_elements_by_class_name(
   'result__url')

# TODO print the first 10 result URLs
```

Another Way to Find Links

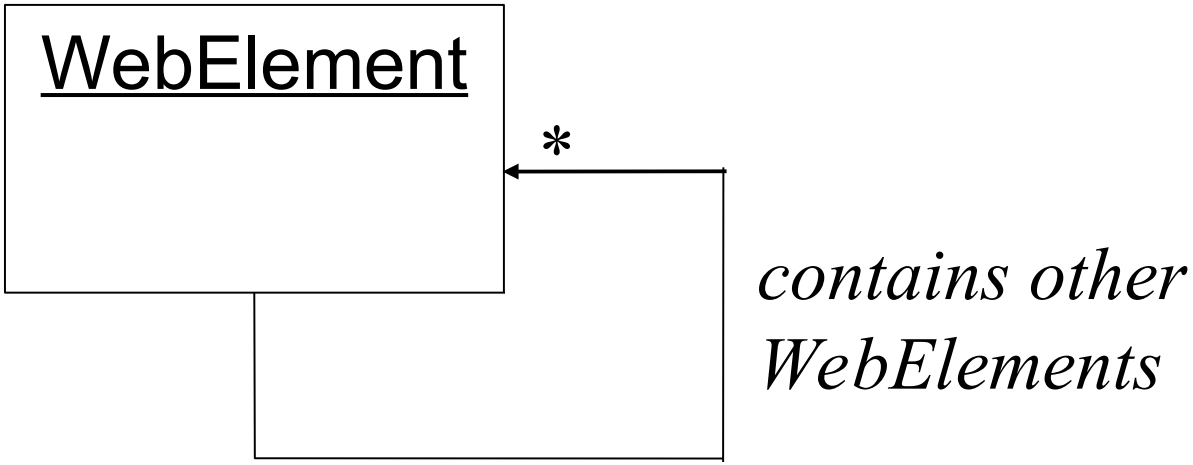
```
# The Hyperlinks use class 'result__a'

links = browser.
  find_elements_by_class_name('result__a')

for link in links:

    if link.tag_name == 'a':

        url = link.get_attribute('href')

        print(url)
```

Composite Design Pattern

`WebElement` may contain other `WebElements`.

`WebElement` is the primary object for interacting with a web page using Selenium.

`WebDriver` contains many of the same methods as WebElement



Headless Browsing

You can run a browser without opening a U.I. window.

This is called headless mode.

May be necessary when running E2E tests on a C.I. server.

It is *faster*, too.

https://developer.mozilla.org/en-US/docs/Mozilla/Firefox/Headless_mode

References

Good Selenium Tutorial in Python

https://blog.testproject.io/2019/07/16/web-ui-testing-python-pytest-selenium-webdriver

The same author has other good testing tutorials:

https://blog.testproject.io/2019/07/16/

Exercise

Write a unit test for this:

When I search DuckDuckGo for Kasetsart University,

then at least 1 of the top-10 search results contains a link to `https://www.ku.ac.th/(anything)`

1. Use `setUp` to create the browser instance.
2. Write a unit test method go perform the test above.

Wrong: testing for exact match of "https://www.ku.ac.th/", because KU's home page could change.

3. Once your test works, add headless mode to setUp and rerun. My intro to Selenium shows how to.