Installation

==========

1. Selenium:

```
pip install selenium
```

CODE:

```
from selenium import webdriver

DRIVER_PATH = '/path/to/chromedriver'
driver = webdriver.Chrome(executable_path=DRIVER_PATH)
driver.get('https://google.com')
```

HEADLESS BROWSER:

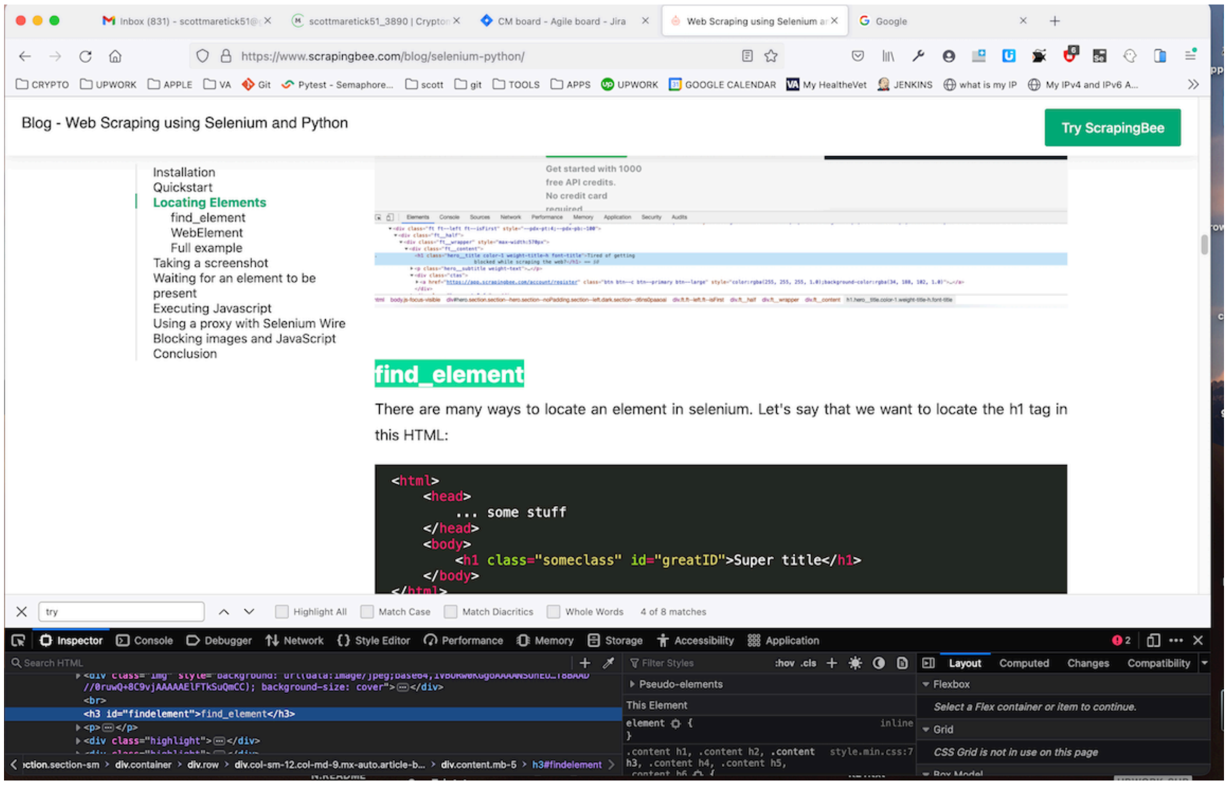
```
from selenium import webdriver
from selenium.webdriver.chrome.options import Options

options = Options()
options.headless = True
options.add_argument("--window-size=1920,1200")

driver = webdriver.Chrome(options=options,
executable_path=DRIVER_PATH)
driver.get("https://www.nintendo.com/")
print(driver.page_source)
driver.quit()
```

LOCATE ELEMENTS:

1. Navigate to the element you want to identify in browser
2. Right click on element & select “Inspect”
3. the detailed HTML code will be in the inspection window at the bottom of the browser (FireFox) & on the right (Chrome)



Blog - Web Scraping using Selenium and Python

Try ScrapingBee

- Installation
- Quickstart
- **Locating Elements**
  - find_element
  - WebElement
  - Full example
- Taking a screenshot
- Waiting for an element to be present
- Executing Javascript
- Using a proxy with Selenium Wire
- Blocking images and JavaScript
- Conclusion

find_element

There are many ways to locate an element in selenium. Let's say that we want to locate the h1 tag in this HTML:

```
<html>
    <head>
        ... some stuff
    </head>
    <body>
        <h1 class="someclass" id="greatID">Super title</h1>
    </body>
</html>
```

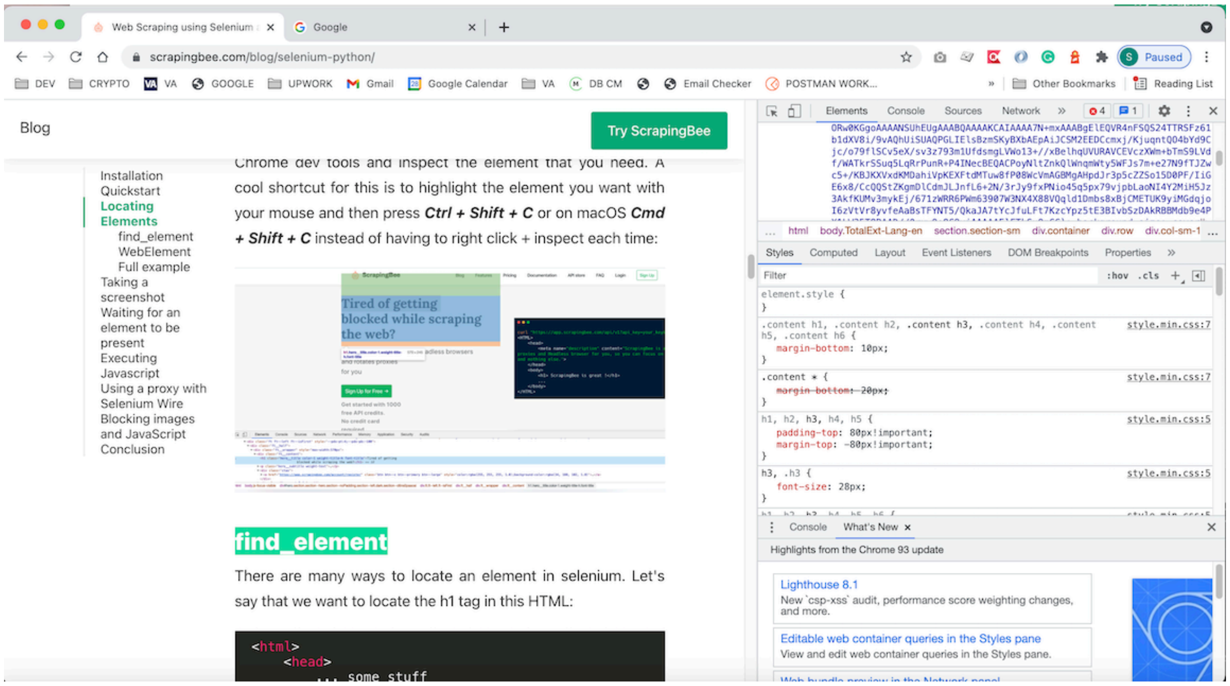
```
try    Highlight All   Match Case   Match Diacritics   Whole Words   4 of 8 matches
```

```
<div class="img" style="background: url(data:image/jpeg;base64,iVBORw0KGgoAAAANSUhEU…T8BAAD
//0ruwQ+8C9vjAAAAAElFTkSuQmCC); background-size: cover">…</div>
<br>
<h3 id="findelement">find_element</h3>
<p>…</p>
<div class="highlight">…</div>
```

```
element {                                  inline
}
.content h1, .content h2, .content        style.min.css:7
h3, .content h4, .content h5,
```

Select a Flex container or item to continue.

CSS Grid is not in use on this page



Blog

Try ScrapingBee

Chrome dev tools and inspect the element that you need. A cool shortcut for this is to highlight the element you want with your mouse and then press ***Ctrl + Shift + C*** or on macOS ***Cmd + Shift + C*** instead of having to right click + inspect each time:

find_element

There are many ways to locate an element in selenium. Let's say that we want to locate the h1 tag in this HTML:

```
<html>
    <head>
        ... some stuff
```

0Rw0KGgoAAAANSUhEUgAAABQAAAAKCAIAAAA7N+mxAAABgElEQVR4nFSQS24TTRSFz61
b1dXV8i/9vAQhUiSUAQPGLIElsBzmSKyBXbAEpAiJCSM2EEDCcmxj/KjuqntQ04bYd9C
jc/o79flSCv5eX/sv3z793m1UfdsmgLVWo13+//xBelhqUVURAVCEVczXWm+bTmS9LVd
f/WATkrSSuq5LqRrPunR+P4INecBEQACPoyNltZnkQlWnqmWty5WFJs7m+e27N9fTJZw
c5+/KBJKXVxdKMDahiVpKEXFtdMTuw8fP08WcVmAGBMgAHpdJr3p5cZZSo15D0PF/IiG
E6x8/CcQQStZKgmDlCdmJLJnfL6+2N/3rJy9fxPNio45q5px79vjpbLaoNI4Y2MiH5Jz
3AkfKUMv3mykEj/671zWRR6PWm63907W3NX4X88VQqld1Dmbs8xBjCMETUK9yiMGdqjo
I6zVtVr8yvfeAaBsTFYNT5/QkaJA7tYcJfuLFt7KzcYpz5tE3BIvbSzDAkRBBMdb9e4P

```
.element.style {
}
.content h1, .content h2, .content h3, .content h4, .content    style.min.css:7
h5, .content h6 {
    margin-bottom: 10px;
}
.content * {                                                    style.min.css:7
    ~~margin-bottom: 20px;~~
}
h1, h2, h3, h4, h5 {                                            style.min.css:5
    padding-top: 80px!important;
    margin-top: -80px!important;
}
h3, .h3 {                                                       style.min.css:5
    font-size: 28px;
}
```

Highlights from the Chrome 93 update

Lighthouse 8.1
New `csp-xss` audit, performance score weighting changes, and more.

Editable web container queries in the Styles pane
View and edit web container queries in the Styles pane.

WebElement

==========

Accessing the text of the element with the property element.text
Clicking on the element with element.click()
Accessing an attribute with element.get_attribute('class')
Sending text to an input with:
element.send_keys('mypassword')

Honeypots are mechanisms used by website owners to detect bots.
For example, if an HTML input has the attribute type=hidden like this:
<input type="hidden" id="custId" name="custId" value="">
In our example, authenticating to Hacker News is not really useful on its own.
However, you could imagine creating a bot to automatically post a link to your latest blog post.

In order to authenticate we need to:
Go to the login page using driver.get()
Select the username input using driver.find_element_by_* and then element.send_keys() to send text to the input
Follow the same process with the password input
Click on the login button using element.click()

CODE:

```
driver.get("https://news.ycombinator.com/login")
login = driver.find_element_by_xpath("//
input").send_keys(USERNAME)
password = driver.find_element_by_xpath("//
input[@type='password']").send_keys(PASSWORD)
submit = driver.find_element_by_xpath("//
input[@value='login']").click()
```

How do we know if we are logged in?

```
try:
    logout_button = driver.find_element_by_id("logout")
    print('Successfully logged in')
except NoSuchElementException:
    print('Incorrect login/password')

try:
    element = WebDriverWait(driver, 5).until(
        EC.presence_of_element_located((By.ID, "mySuperId"))
    )
finally:
    driver.quit()
```

Waiting for an element to be present
=============================

```
try:
    element = WebDriverWait(driver, 5).until(
        EC.presence_of_element_located((By.ID, "mySuperId"))
    )
finally:
    driver.quit()
```

Executing Javascript

```
javaScript = "window.scrollBy(0,1000);"
driver.execute_script(javaScript)
```

Using a proxy with Selenium Wire

```
pip install selenium-wire
```

This code snippet shows you how to quickly use your headless browser behind a proxy:

```
# Install the Python selenium-wire library:
# pip install selenium-wire
from seleniumwire import webdriver
```

```
proxy_username = "USER_NAME"
proxy_password = "PASSWORD"
proxy_url = "http://proxy.scrapingbee.com"
proxy_port = 8886

options = {
    "proxy": {
        "http": f"http://{proxy_username}:{proxy_password}
@{proxy_url}:{proxy_port}",
        "verify_ssl": False,
    },
}

URL = "https://httpbin.org/headers?json"

driver = webdriver.Chrome(
    executable_path="YOUR-CHROME-EXECUTABLE-PATH",
    seleniumwire_options=options,
)
driver.get(URL)
```

Blocking images and JavaScript

With Selenium, by using the correct Chrome options, you can block some requests from being made.

```
from selenium import webdriver

chrome_options = webdriver.ChromeOptions()

### This blocks images and javascript requests
chrome_prefs = {
    "profile.default_content_setting_values": {
        "images": 2,
        "javascript": 2,
    }
}
chrome_options.experimental_options["prefs"] = chrome_prefs
###

driver = webdriver.Chrome(
    executable_path="YOUR-CHROME-EXECUTABLE-PATH",
    chrome_options=chrome_options,
)
driver.get(URL)
```