Section 1: Getting Started with Selenium

What is Selenium?

Selenium is like a robot that controls a web browser for you. You tell it what to do (open a site, click a button, grab text) using Python code. It's great for scraping dynamic websites or automating tasks (e.g., filling forms).

What You Need

1. **Python**: You've got this installed.
2. **Selenium**: Install it with pip install selenium.
3. **WebDriver Manager**: Install with pip install webdriver-manager (makes setup easy).
4. **Chrome Browser**: Already on your computer.

Basic Example: Open a Website

Let's start simple—open Google and grab the page title.

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

# Setup the robot (Chrome browser)
service = Service(ChromeDriverManager().install())
driver = webdriver.Chrome(service=service)

# Tell the robot to visit Google
driver.get("https://www.google.com")

# Get the page title (what's in the browser tab)
title = driver.title
print("Page Title:", title)

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

- **What Happens**:
  - ChromeDriverManager().install(): Sets up the Chrome robot (downloads ChromeDriver if needed).
  - webdriver.Chrome: Opens Chrome.
  - driver.get: Goes to Google.
  - driver.title: Grabs "Google" (the tab title).
  - driver.quit: Closes the browser.
- **Try It**: Run this. You'll see Chrome open, visit Google, and print "Page Title: Google".

Section 2: Finding Stuff on a Page

How Selenium Finds Things

Websites are like maps—Selenium finds "treasures" (text, buttons, links) using locators (like GPS). You've used By.CSS_SELECTOR; let's learn more.

Locators

1. **By.ID**: Finds by id (unique name).
2. **By.CLASS_NAME**: Finds by class (e.g., product_pod).
3. **By.TAG_NAME**: Finds by tag (e.g., div, a).
4. **By.CSS_SELECTOR**: Uses CSS rules (e.g., h3 a).
5. **By.XPATH**: Uses a path (e.g., //h3/a).

Example: Find a Title

Let's scrape a quote from http://quotes.toscrape.com/.

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

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

# Open the quotes site
driver.get("http://quotes.toscrape.com/")

# Find the first quote
quote = driver.find_element(By.CSS_SELECTOR, "div.quote span.text").text
author = driver.find_element(By.CSS_SELECTOR, "div.quote small.author").text
print("Quote:", quote)
print("Author:", author)

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

- **What Happens**:
  - div.quote: Finds the quote box.
  - span.text: Gets the quote text (e.g., "The world as we...").
  - small.author: Gets the author (e.g., "Albert Einstein").
- **Try It**: Run this. You'll see one quote and its author printed.

Multiple Items

Use find_elements to get all quotes on a page:

```
quotes = driver.find_elements(By.CSS_SELECTOR, "div.quote")
for quote in quotes:
    text = quote.find_element(By.CSS_SELECTOR, "span.text").text
    author = quote.find_element(By.CSS_SELECTOR, "small.author").text
    print(f"Quote: {text}, Author: {author}")
```

- **Difference**: find_element gets one; find_elements gets all.

Section 3: Moving Around (Navigation)

Clicking and Pagination

Websites have buttons—like "Next" on books.toscrape.com. Let's click it.

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

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

# Open bookstore
driver.get("https://books.toscrape.com/")

# Click the "Next" button
next_button = driver.find_element(By.CSS_SELECTOR, "li.next a")
next_button.click()
time.sleep(2)  # Wait for the page to load
print("New page title:", driver.title)

# Close
driver.quit()
```

- **What Happens**: Finds the “Next” button and clicks it, moving to page 2.
- **Try It**: Run this. Watch Chrome go to the next page.

Full Pagination Example

Scrape all books across pages:

```
data = []
while True:
    books = driver.find_elements(By.CSS_SELECTOR, "article.product_pod")
    for book in books:
        title = book.find_element(By.CSS_SELECTOR, "h3 a").get_attribute("title")
        data.append({"Title": title})

    try:
        next_button = driver.find_element(By.CSS_SELECTOR, "li.next a")
        next_button.click()
        time.sleep(2)
    except:
        print("Done with all pages!")
        break
print(f"Found {len(data)} books")
```

- **What Happens**: Loops through all pages, collecting titles.

Section 4: Waiting Smartly

Why Wait?

Pages take time to load. time.sleep works but isn't smart—Selenium has better tools.

Explicit Wait

Wait for something specific:

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

# Wait for books to load
books = WebDriverWait(driver, 10).until(
    EC.presence_of_all_elements_located((By.CSS_SELECTOR, "article.product_pod"))
)
print(f"Found {len(books)} books on this page")
```

- **What Happens**: Waits up to 10 seconds for books to appear, then proceeds.
- **Try It**: Replace time.sleep in your script with this.

Implicit Wait

Set a default wait for all finds:

```
driver.implicitly_wait(10)  # Wait up to 10 seconds for any element
books = driver.find_elements(By.CSS_SELECTOR, "article.product_pod")
```

- **Difference**: Applies globally, less flexible than explicit wait.

Section 5: Interacting with Pages

Typing

Fill a search box (if the site had one):

```
search = driver.find_element(By.NAME, "q")  # Example search field
search.send_keys("Fiction")
search.submit()
time.sleep(2)
print("Searched for Fiction!")
```

Scrolling

Scroll to the bottom:

```
driver.execute_script("window.scrollTo(0, document.body.scrollHeight);")
time.sleep(2)
```

Section 6: Intermediate Skills

Handling Errors

Catch missing elements:

```
try:
    title = driver.find_element(By.CSS_SELECTOR, "h3 a").get_attribute("title")
except Exception:
    title = "Not Found"
print("Title:", title)
```

Getting Attributes

Like your title attribute:

```
link = driver.find_element(By.CSS_SELECTOR, "h3 a")
href = link.get_attribute("href")  # Gets the URL
print("Link URL:", href)
```

Section 7: Advanced Selenium

Headless Mode

Run without seeing the browser:

```
options = webdriver.ChromeOptions()
options.add_argument("--headless")
driver = webdriver.Chrome(service=service, options=options)
driver.get("https://books.toscrape.com/")
print("Title (headless):", driver.title)
```

- **Why**: Faster, runs in the background.

Frames and Popups

Switch to a frame (if a site uses them):

```
driver.switch_to.frame("frame_id")
# Do stuff
driver.switch_to.default_content()  # Back to main page
```

Multiple Windows

Handle new tabs:

```
driver.find_element(By.CSS_SELECTOR, "a[target='_blank']").click()
windows = driver.window_handles
driver.switch_to.window(windows[1])  # Switch to new tab
print("New tab title:", driver.title)
```

Section 8: Full Project

Let's scrape quotes.toscrape.com with all pages:

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

# Setup Chrome (headless)
options = webdriver.ChromeOptions()
options.add_argument("--headless")
service = Service(ChromeDriverManager().install())
driver = webdriver.Chrome(service=service, options=options)

# Open quotes site
driver.get("http://quotes.toscrape.com/")
data = []

# Scrape all pages
while True:
    quotes = WebDriverWait(driver, 10).until(
        EC.presence_of_all_elements_located((By.CSS_SELECTOR, "div.quote"))
    )
    for quote in quotes:
        try:
            text = quote.find_element(By.CSS_SELECTOR, "span.text").text
            author = quote.find_element(By.CSS_SELECTOR, "small.author").text
            tags = ", ".join([tag.text for tag in quote.find_elements(By.CSS_SELECTOR, "a.tag")])
        except:
            text, author, tags = "Not Found", "Not Found", "Not Found"
        data.append({"Quote": text, "Author": author, "Tags": tags})

    try:
        next_button = driver.find_element(By.CSS_SELECTOR, "li.next a")
        next_button.click()
    except:
        break

# Save and close
df = pd.DataFrame(data)
df.to_csv("quotes_full.csv", index=False)
print(f"Saved {len(data)} quotes!")
driver.quit()
```

- **What Happens**: Scrapes all quotes, authors, and tags across pages, saves them headlessly.