Name – Aarohan Sarkar Reg No. – 12213072

Selenium WebDriver Lab Assignment 1

This project demonstrates basic Selenium WebDriver functionality using Python, running in a Docker container for consistent environment setup.

Environment Setup

Prerequisites

- Docker Desktop installed

- Git (optional)

Project Structure

```
Dockerfile          # Docker configuration
 requirements.txt     # Python dependencies
 tests/              # Test directory
 test_selenium_basic.py  # Basic Selenium test script
 README.md           # This file
```

Setup Instructions

1. Clone or download this repository to your local machine.

2. Build the Docker image:

```bash
docker build -t selenium-lab .
```

3. Run the tests:

```bash
docker run selenium-lab
```

Technical Details

Dependencies

- Python 3.9
- Selenium 4.18.1
- webdriver-manager 4.0.1
- pytest 8.0.2
- Chrome browser (installed in Docker)

Test Script Details

The test script (`test_selenium_basic.py`) demonstrates:

- Browser initialization with headless Chrome
- Navigation to example.com
- Page title and URL retrieval
- Element location and interaction
- Attribute value extraction
- Graceful browser session closure

Challenges and Solutions

1. **Chrome in Docker**:
   - Challenge: Running Chrome in a containerized environment
   - Solution: Configured Chrome to run in headless mode with appropriate flags (--no-sandbox, --disable-dev-shm-usage)

2. **WebDriver Management**:
   - Challenge: Managing Chrome WebDriver versions
   - Solution: Used selenium-manager for automatic WebDriver management

3. **Container Security**:
   - Challenge: Running browser safely in container
   - Solution: Implemented proper security configurations in Dockerfile

Best Practices Implemented

1. **Code Organization**:
   - Modular test structure
   - Clear separation of concerns
   - Well-documented code with comments

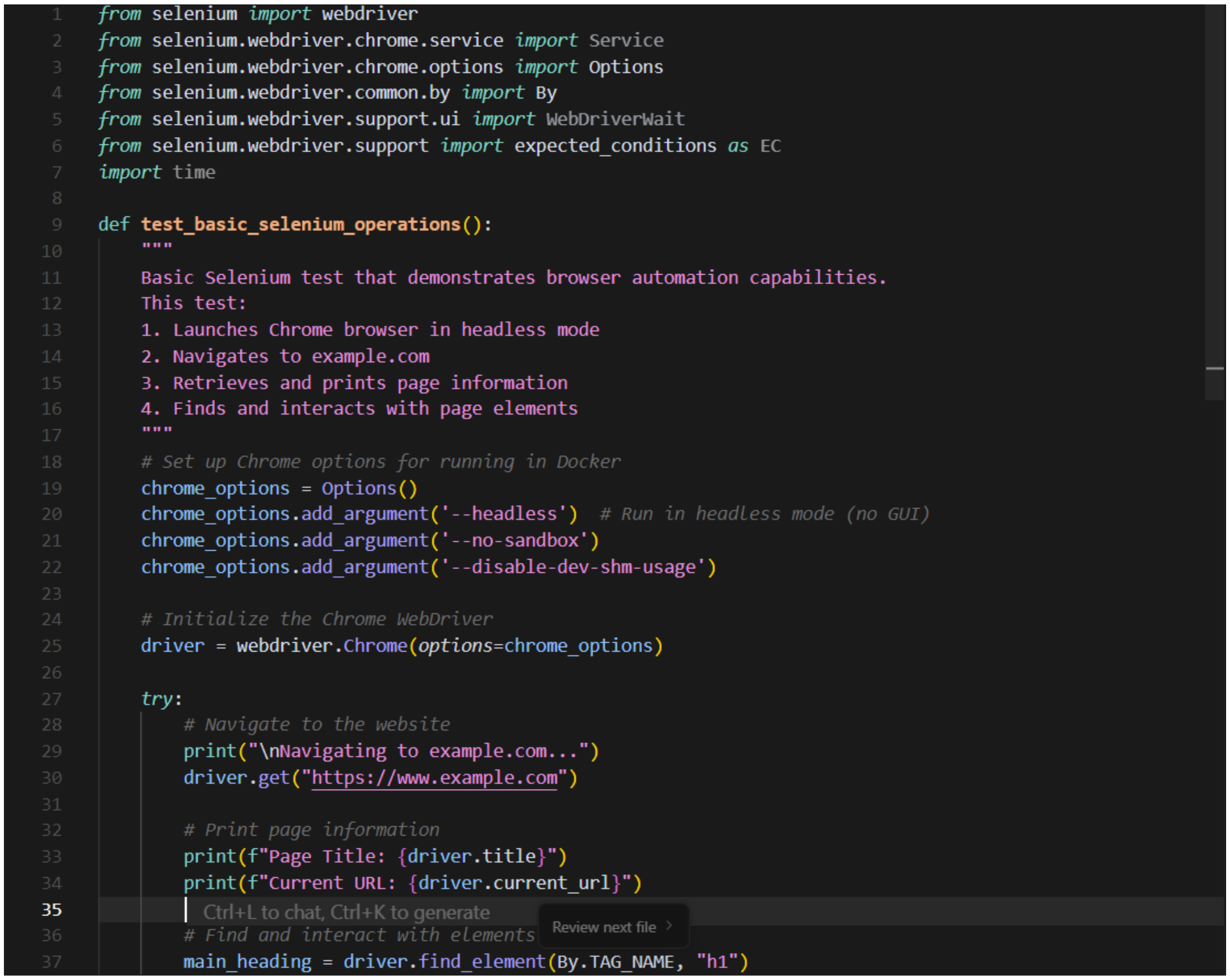
2. **Error Handling**:
   - Try-finally block for proper browser cleanup
   - Graceful session termination

3. **Docker Best Practices**:
   - Multi-stage build
   - Minimal image size
   - Security considerations

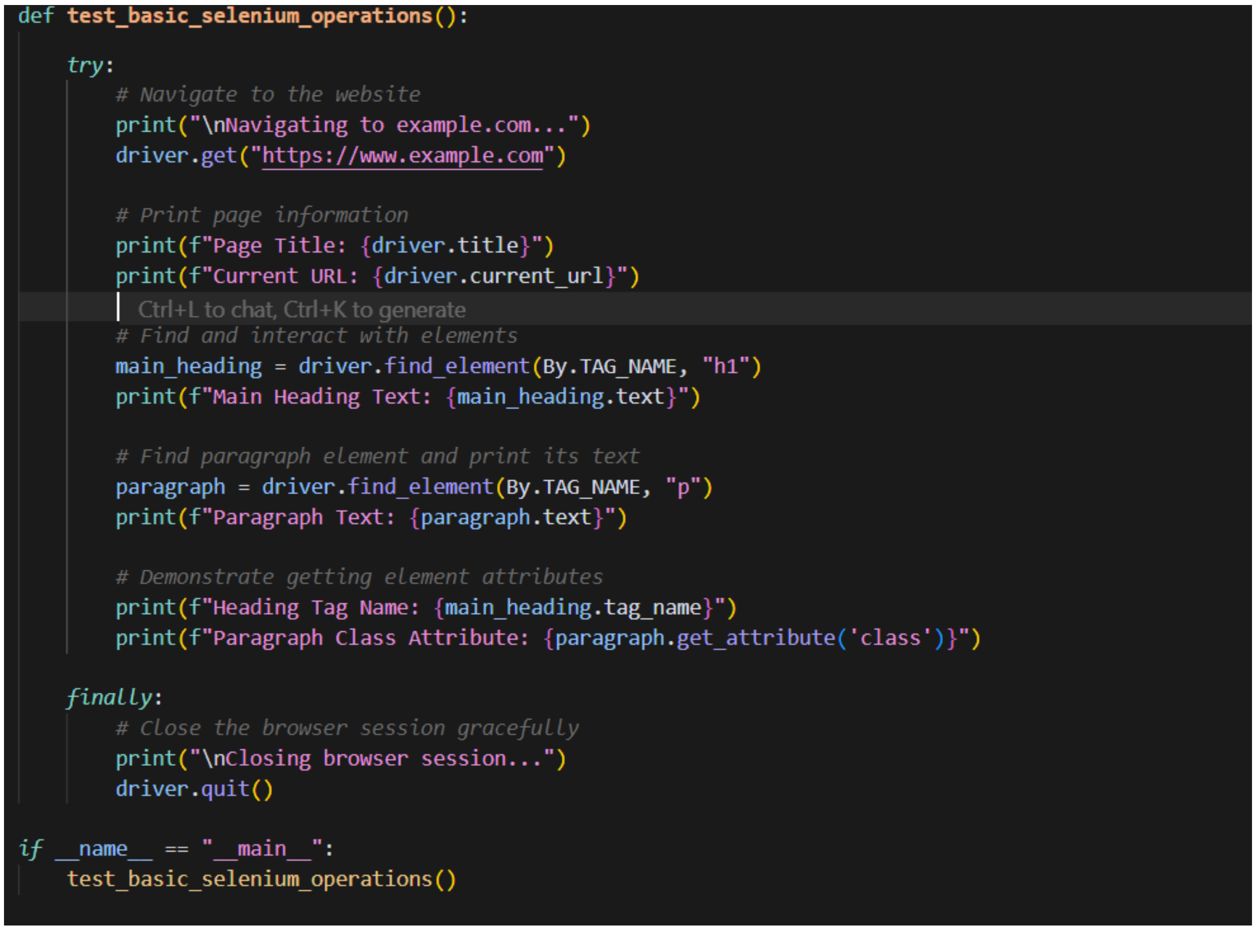
Running Tests Locally (Without Docker)

If you prefer to run tests without Docker:

1. Install Python 3.9
2. Install Chrome browser
3. Create a virtual environment:
   ```bash
   python -m venv venv
   source venv/bin/activate  # On Windows: venv\Scripts\activate
   ```
4. Install dependencies:
   ```bash
   pip install -r requirements.txt
   ```
5. Run tests:
   ```bash
   python -m pytest tests/
   ```

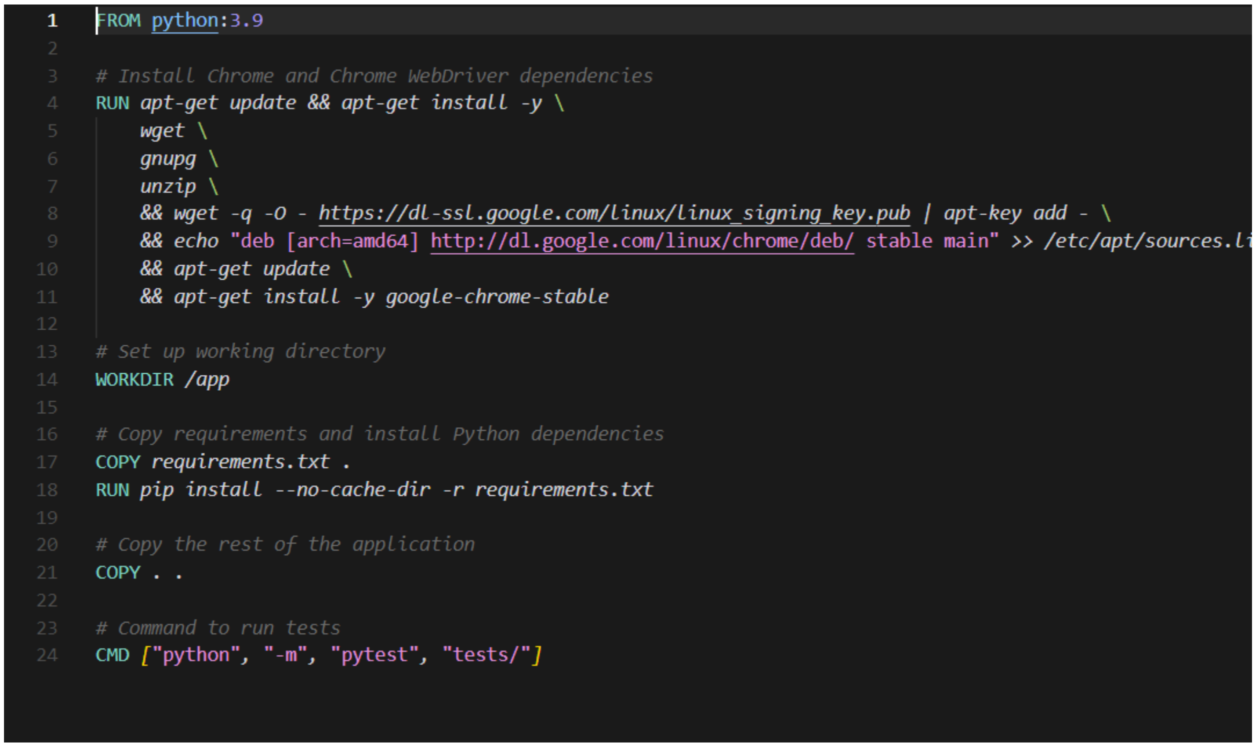


```python
from selenium import webdriver
from selenium.webdriver.chrome.service import Service
from selenium.webdriver.chrome.options import Options
from selenium.webdriver.common.by import By
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
import time

def test_basic_selenium_operations():
    """
    Basic Selenium test that demonstrates browser automation capabilities.
    This test:
    1. Launches Chrome browser in headless mode
    2. Navigates to example.com
    3. Retrieves and prints page information
    4. Finds and interacts with page elements
    """
    # Set up Chrome options for running in Docker
    chrome_options = Options()
    chrome_options.add_argument('--headless')  # Run in headless mode (no GUI)
    chrome_options.add_argument('--no-sandbox')
    chrome_options.add_argument('--disable-dev-shm-usage')

    # Initialize the Chrome WebDriver
    driver = webdriver.Chrome(options=chrome_options)

    try:
        # Navigate to the website
        print("\nNavigating to example.com...")
        driver.get("https://www.example.com")

        # Print page information
        print(f"Page Title: {driver.title}")
        print(f"Current URL: {driver.current_url}")
        
        # Find and interact with elements
        main_heading = driver.find_element(By.TAG_NAME, "h1")
```



```python
def test_basic_selenium_operations():

    try:
        # Navigate to the website
        print("\nNavigating to example.com...")
        driver.get("https://www.example.com")

        # Print page information
        print(f"Page Title: {driver.title}")
        print(f"Current URL: {driver.current_url}")
        | Ctrl+L to chat, Ctrl+K to generate
        # Find and interact with elements
        main_heading = driver.find_element(By.TAG_NAME, "h1")
        print(f"Main Heading Text: {main_heading.text}")

        # Find paragraph element and print its text
        paragraph = driver.find_element(By.TAG_NAME, "p")
        print(f"Paragraph Text: {paragraph.text}")

        # Demonstrate getting element attributes
        print(f"Heading Tag Name: {main_heading.tag_name}")
        print(f"Paragraph Class Attribute: {paragraph.get_attribute('class')}")

    finally:
        # Close the browser session gracefully
        print("\nClosing browser session...")
        driver.quit()

if __name__ == "__main__":
    test_basic_selenium_operations()
```

The following is the docker file –



```
FROM python:3.9

# Install Chrome and Chrome WebDriver dependencies
RUN apt-get update && apt-get install -y \
    wget \
    gnupg \
    unzip \
    && wget -q -O - https://dl-ssl.google.com/linux/linux_signing_key.pub | apt-key add - \
    && echo "deb [arch=amd64] http://dl.google.com/linux/chrome/deb/ stable main" >> /etc/apt/sources.li
    && apt-get update \
    && apt-get install -y google-chrome-stable

# Set up working directory
WORKDIR /app

# Copy requirements and install Python dependencies
COPY requirements.txt .
RUN pip install --no-cache-dir -r requirements.txt

# Copy the rest of the application
COPY . .

# Command to run tests
CMD ["python", "-m", "pytest", "tests/"]
```