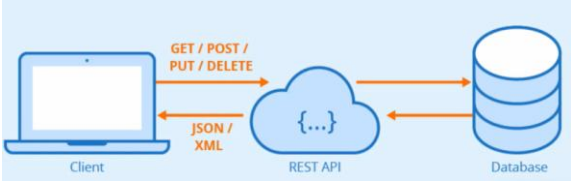
Test Automation using:

- Python
- Playwright
- PyCharm
- PyTest
- GitHub repository

Document highlight:

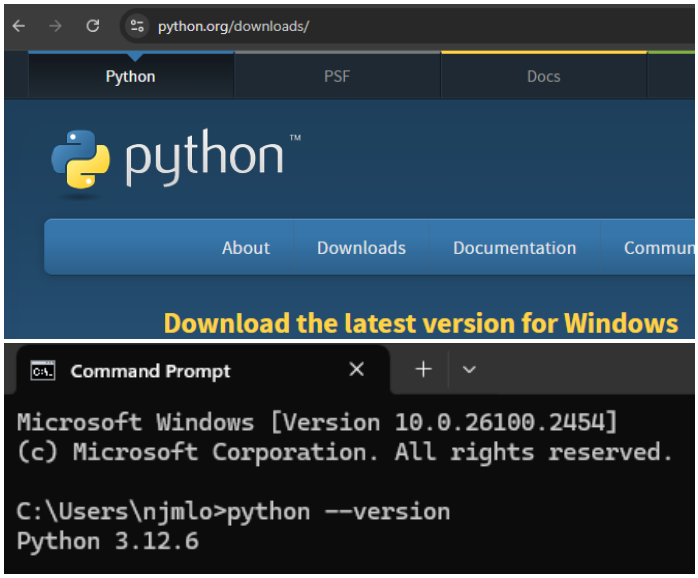
- Tools download and setup
- Working on a web page:
  - Identifying element using css locators
  - Identifying element using XPath locators
  - Selecting values from dropdowns
  - Selecting radio buttons and checkboxes
  - Alerts and dialog box
  - Windows/page handling
  - Cookies and taking page screenshots
  - Page interactions
  - Exception handling and storing elements in lists
  - Ajax handling
  - Page tables
  - Upload and download of files
  - Page desktop video recording and screenshots
  - Web page with Playwright and PyTest
  - Page with fixtures (module and functions)
  - Page with fixtures (class and functions)
  - Parameterizing tests in PyTest with Playwright
  - Test reporting in PyTest with Playwright (PyTest HTML plugin)

1. API Concept (interface between client and server (database), request and response)

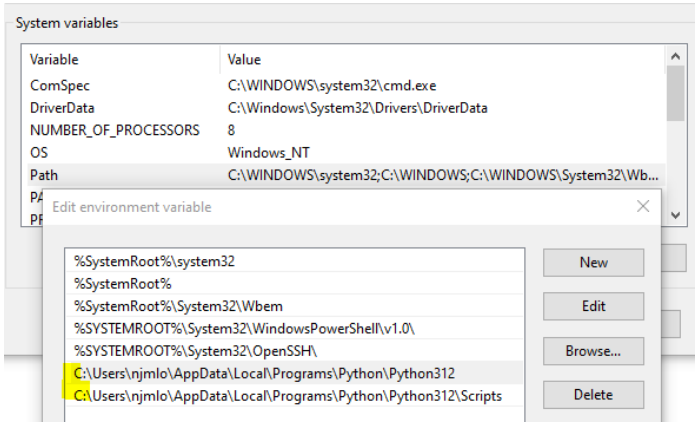


2. Tools download and setup

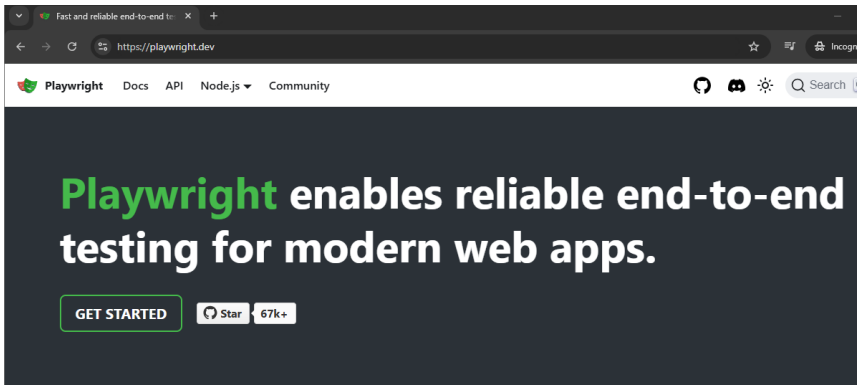
Python download and setup:
https://www.python.org/downloads/



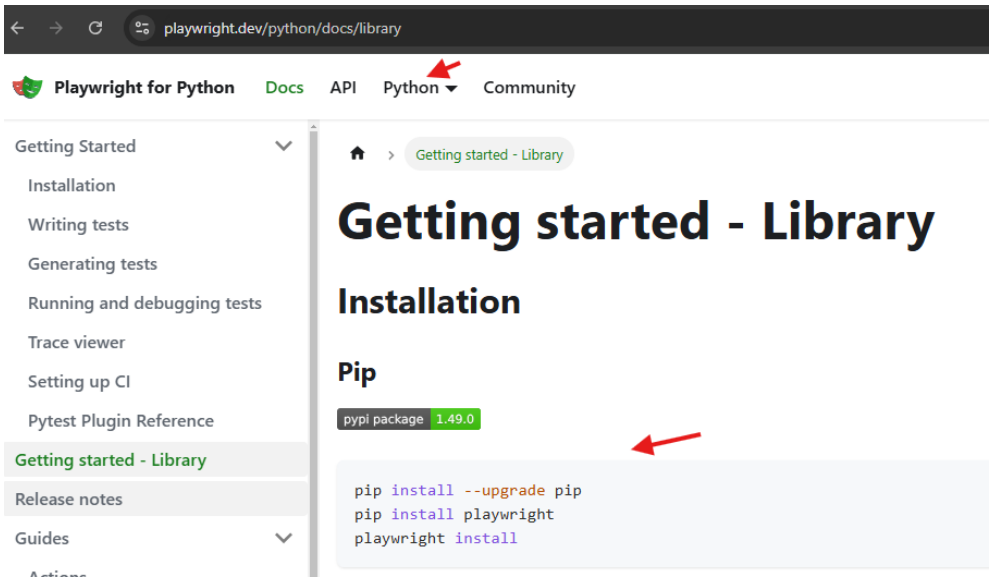
In machine's system variable path:



Playright download and setup:
https://playwright.dev/



Using the command (cmd) line:



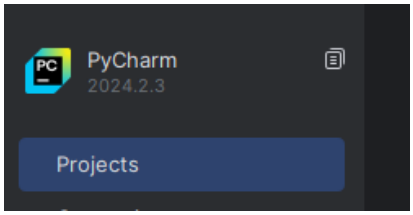

```
PS C:\WINDOWS\system32> pip --version
pip 24.3.1 from C:\Users\njmlo\AppData\Lo
```

```
PS C:\WINDOWS\system32> pip install playwright
Collecting playwright
  Downloading playwright-1.49.0-py3-none-win_amd64.whl.metadata (3.5 kB)
Collecting greenlet==3.1.1 (from playwright)
  Downloading greenlet-3.1.1-cp312-cp312-win_amd64.whl.metadata (3.9 kB)
Collecting pyee==12.0.0 (from playwright)
  Downloading pyee-12.0.0-py3-none-any.whl.metadata (2.8 kB)
Requirement already satisfied: typing-extensions in c:\users\njmlo\appdata\loca
ages (from pyee==12.0.0->playwright) (4.12.2)
Downloading playwright-1.49.0-py3-none-win_amd64.whl (34.0 MB)
   ---------------------------------------- 34.0/34.0 MB 36.0 MB/s eta 0:00:00
Downloading greenlet-3.1.1-cp312-cp312-win_amd64.whl (299 kB)
Downloading pyee-12.0.0-py3-none-any.whl (14 kB)
Installing collected packages: pyee, greenlet, playwright
Successfully installed greenlet-3.1.1 playwright-1.49.0 pyee-12.0.0
PS C:\WINDOWS\system32> playwright --version
Version 1.49.0
```

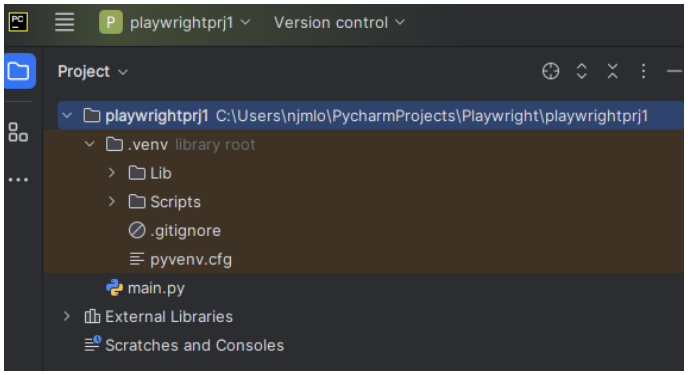
```
PS C:\WINDOWS\system32> playwright install
Downloading Chromium 131.0.6778.33 (playwright build v1148) from https://playwright.azureedge.net/builds/chromium/1148/c
hromium-win64.zip
136.9 MiB [====================] 100% 0.0s
Chromium 131.0.6778.33 (playwright build v1148) downloaded to C:\Users\njmlo\AppData\Local\ms-playwright\chromium-1148
Downloading Chromium Headless Shell 131.0.6778.33 (playwright build v1148) from https://playwright.azureedge.net/builds/
chromium/1148/chromium-headless-shell-win64.zip
87.7 MiB [====================] 100% 0.0s
Chromium Headless Shell 131.0.6778.33 (playwright build v1148) downloaded to C:\Users\njmlo\AppData\Local\ms-playwright\
chromium_headless_shell-1148
Downloading Firefox 132.0 (playwright build v1466) from https://playwright.azureedge.net/builds/firefox/1466/firefox-win
64.zip
85.8 MiB [====================] 100% 0.0s
Firefox 132.0 (playwright build v1466) downloaded to C:\Users\njmlo\AppData\Local\ms-playwright\firefox-1466
Downloading Webkit 18.2 (playwright build v2104) from https://playwright.azureedge.net/builds/webkit/2104/webkit-win64.z
ip
52.7 MiB [====================] 100% 0.0s
Webkit 18.2 (playwright build v2104) downloaded to C:\Users\njmlo\AppData\Local\ms-playwright\webkit-2104
Downloading FFMPEG playwright build v1010 from https://playwright.azureedge.net/builds/ffmpeg/1010/ffmpeg-win64.zip
1.3 MiB [====================] 100% 0.0s
FFMPEG playwright build v1010 downloaded to C:\Users\njmlo\AppData\Local\ms-playwright\ffmpeg-1010
PS C:\WINDOWS\system32>
```

```
C:\Users\njmlo>playwright --version
Version 1.49.0
```

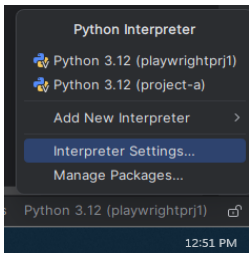
PyCharm download and setup:
https://www.jetbrains.com/pycharm/download

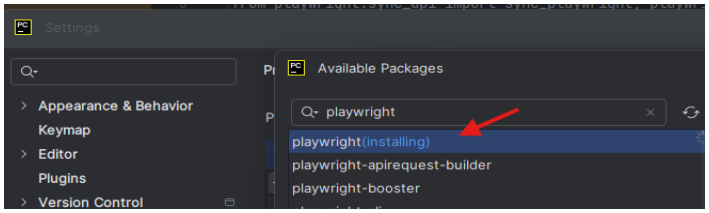


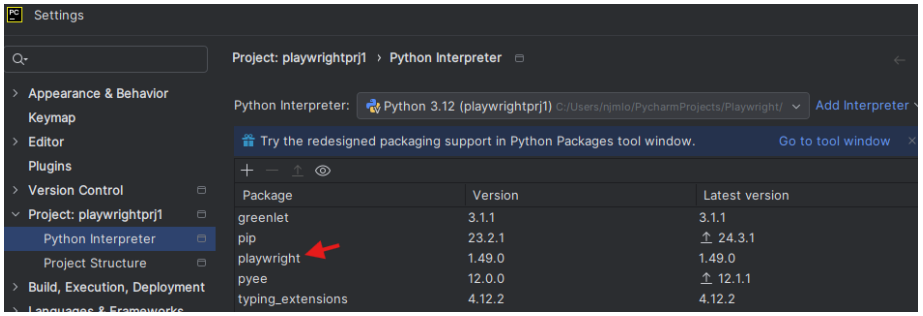
Create a project in PyCharm:



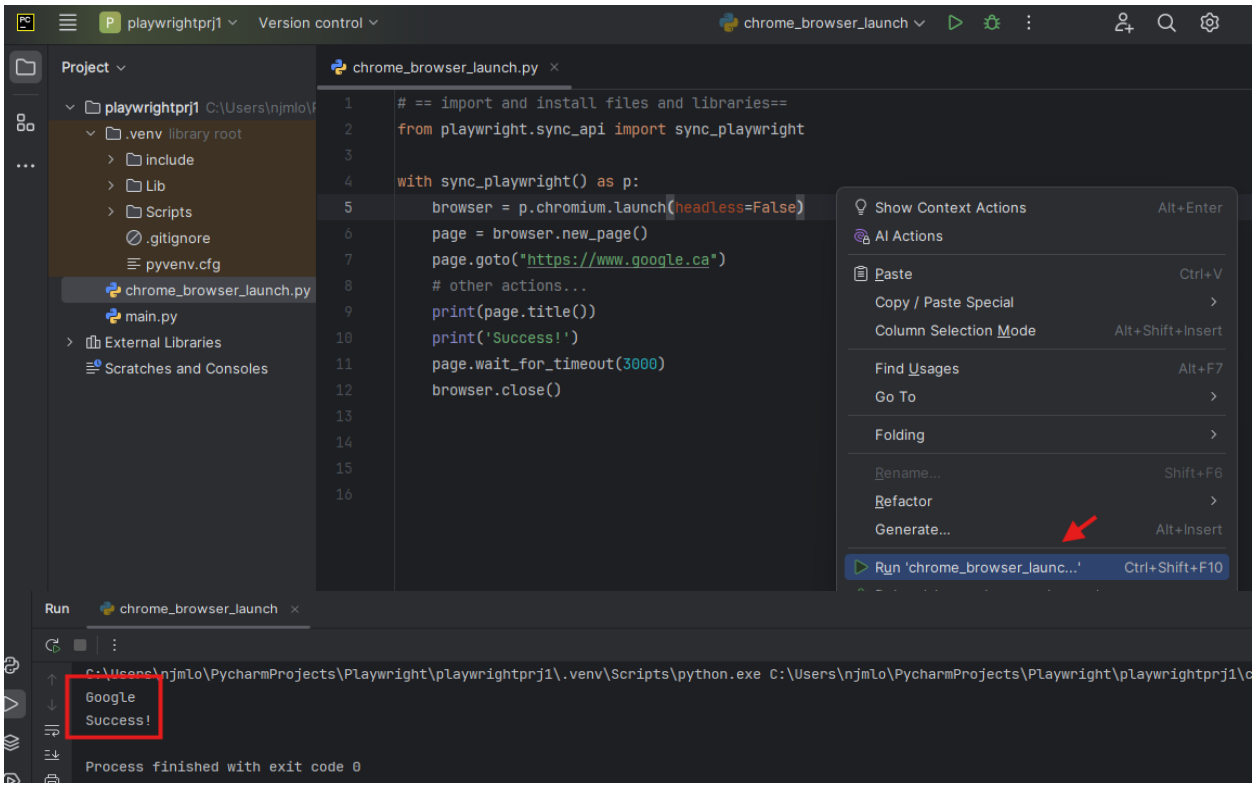
In the interpreter settings, install playwright package:







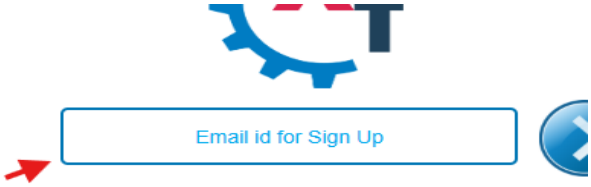
After all the installation and setup, run a quick test:

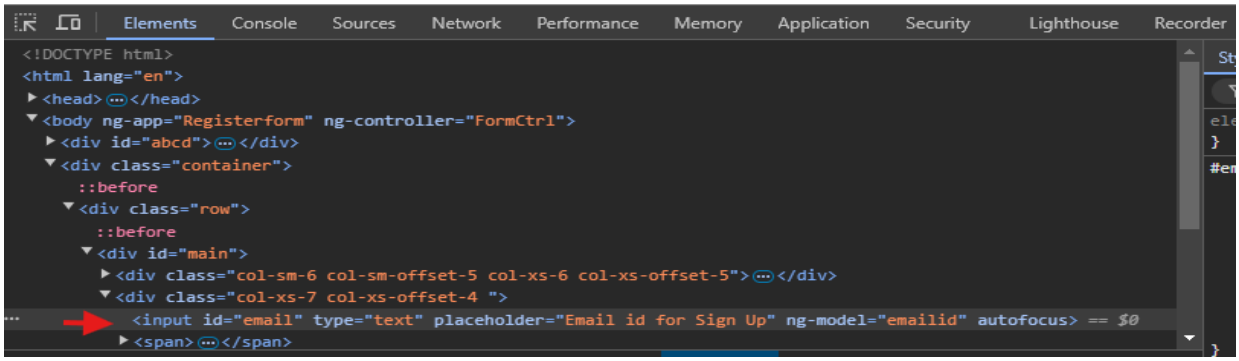


3. Working on a web page identifying element using css locators

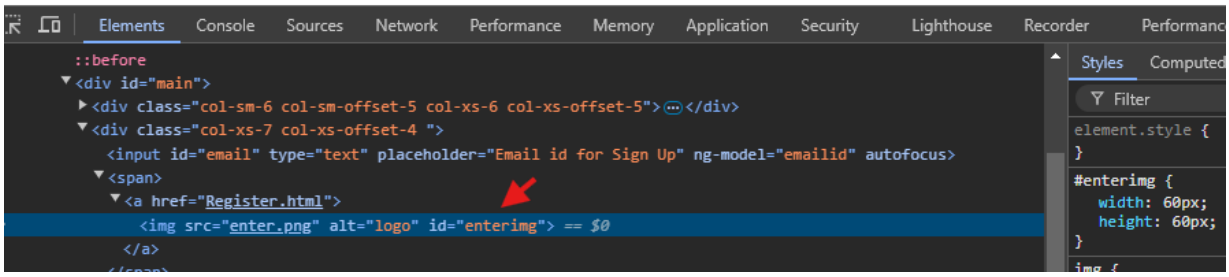
Test page url:
https://demo.automationtesting.in/

Locate (inspect) the element attributes for use as its locator:
Email textbox…





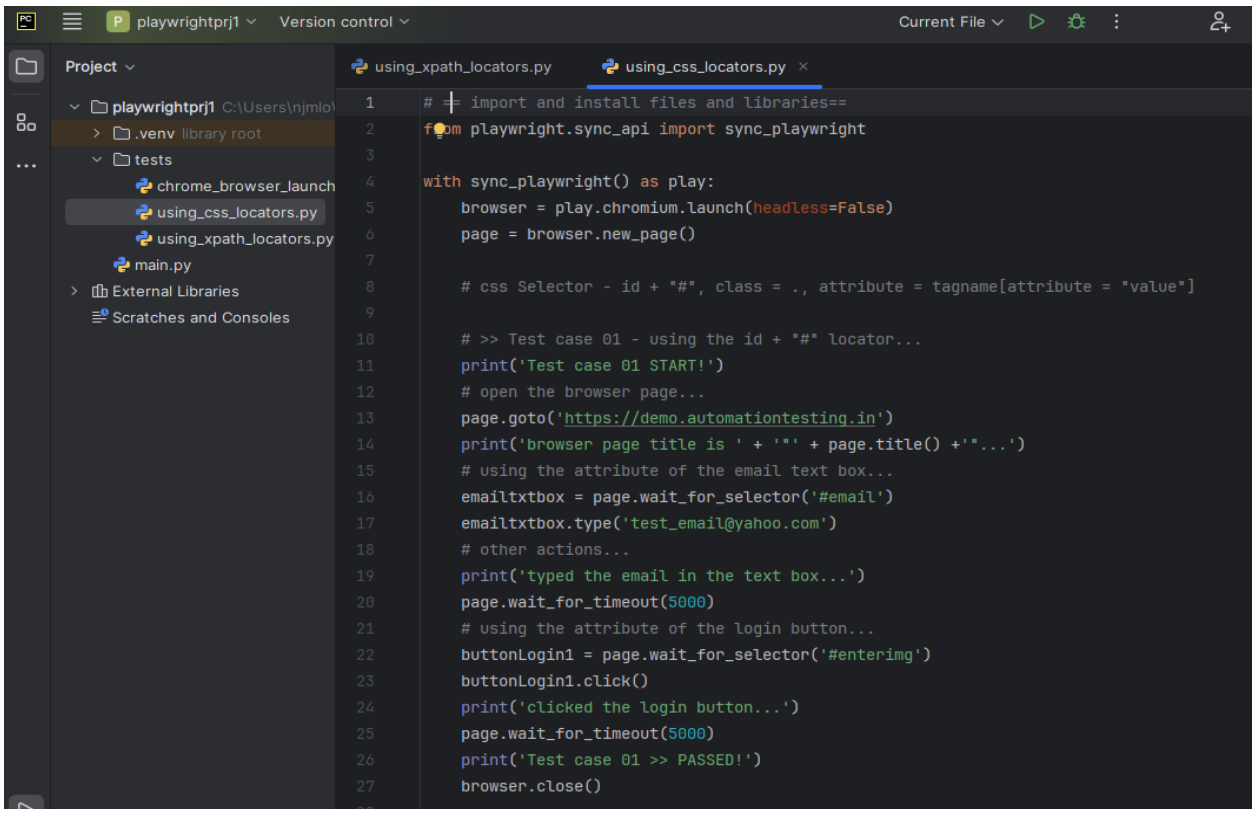
Login button…

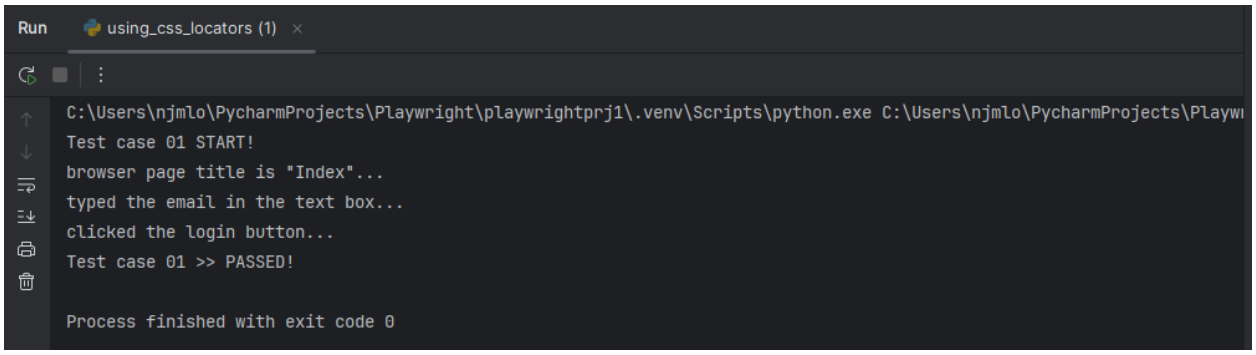


Syntax:

a. (Test case 01) using the id + "#"

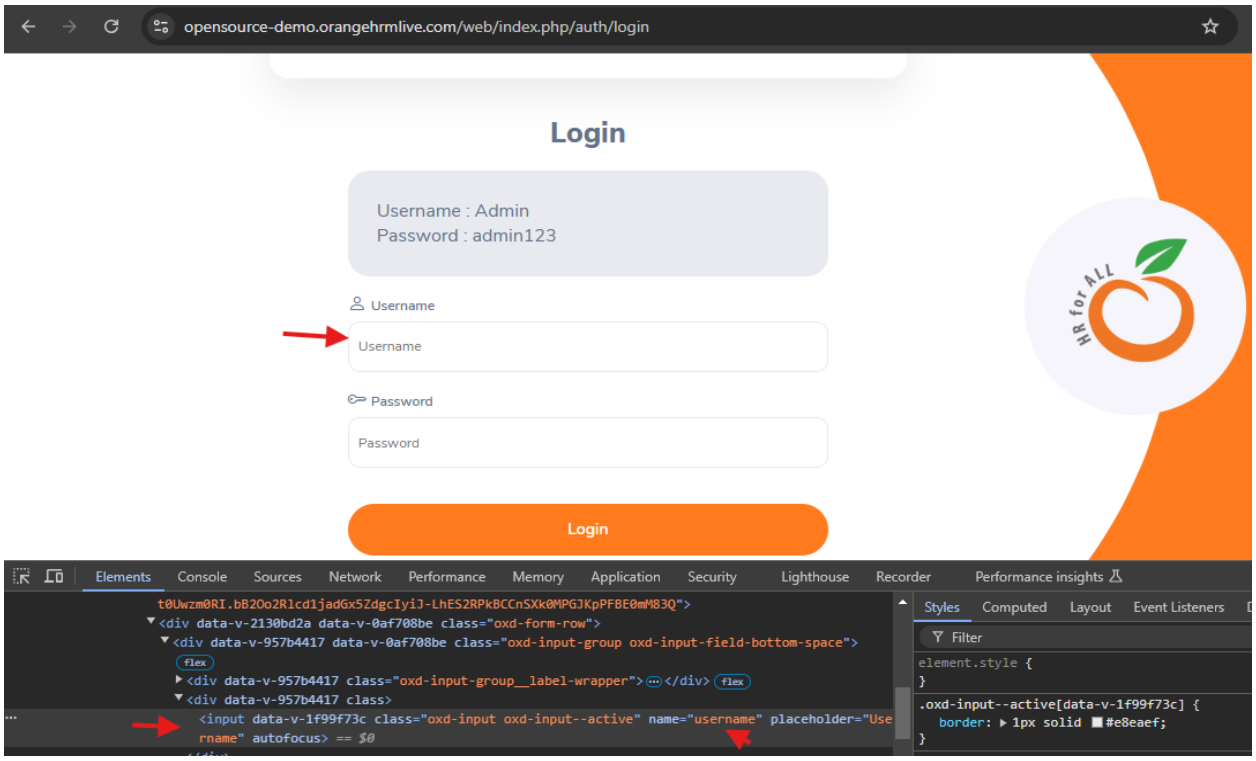
Using the “id” locator, create and run the code:



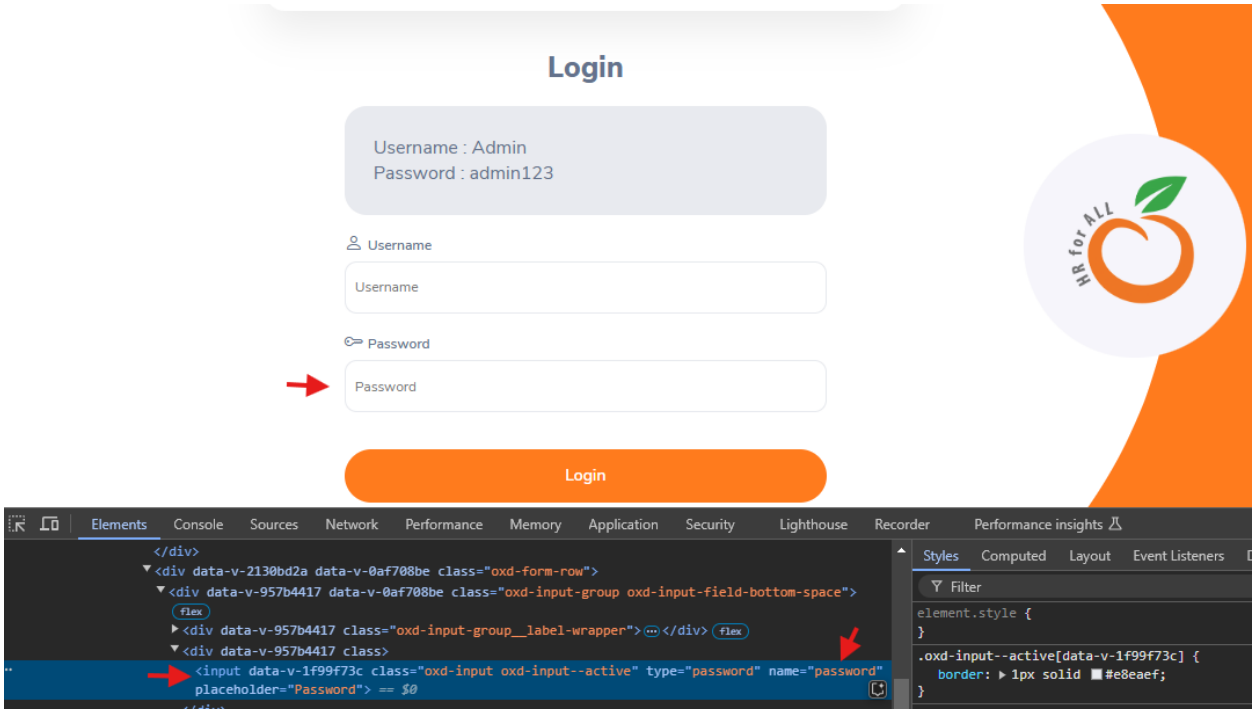


Test page url:
https://opensource-demo.orangehrmlive.com/web/index.php/auth/login

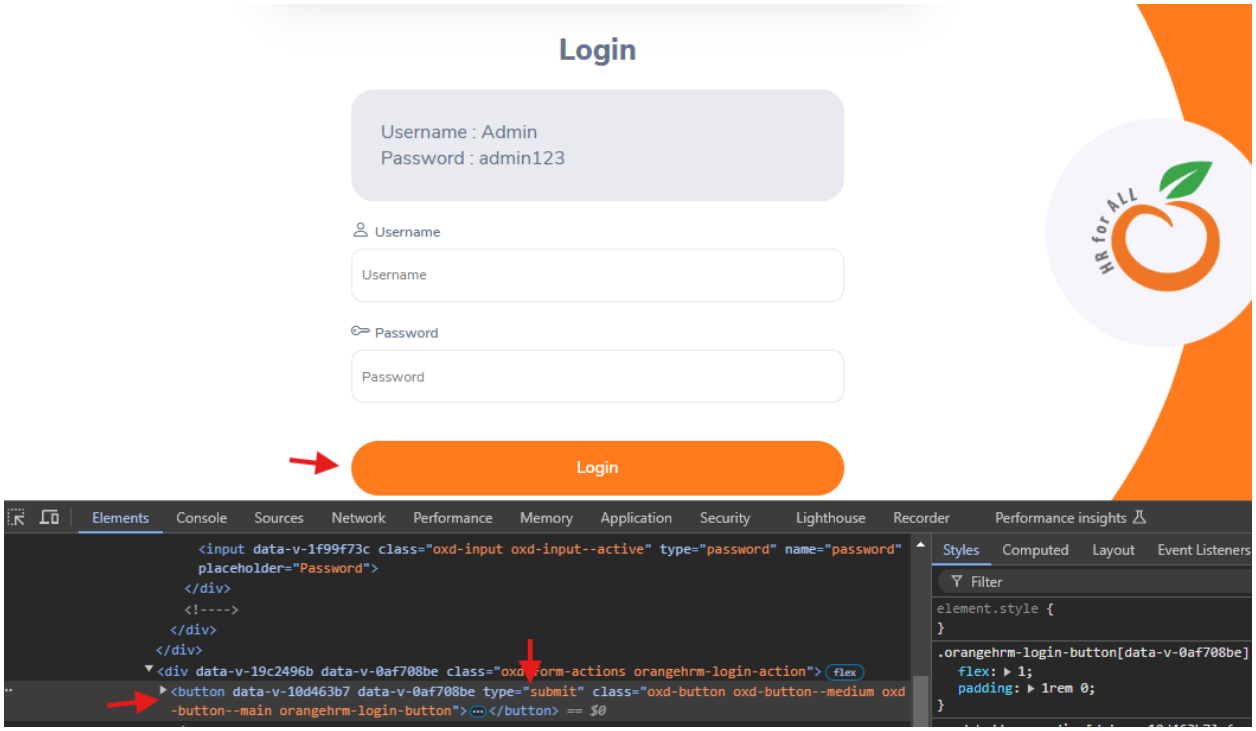
Locate (inspect) the element attributes for use as its locator:
Username text box…



Password text box…



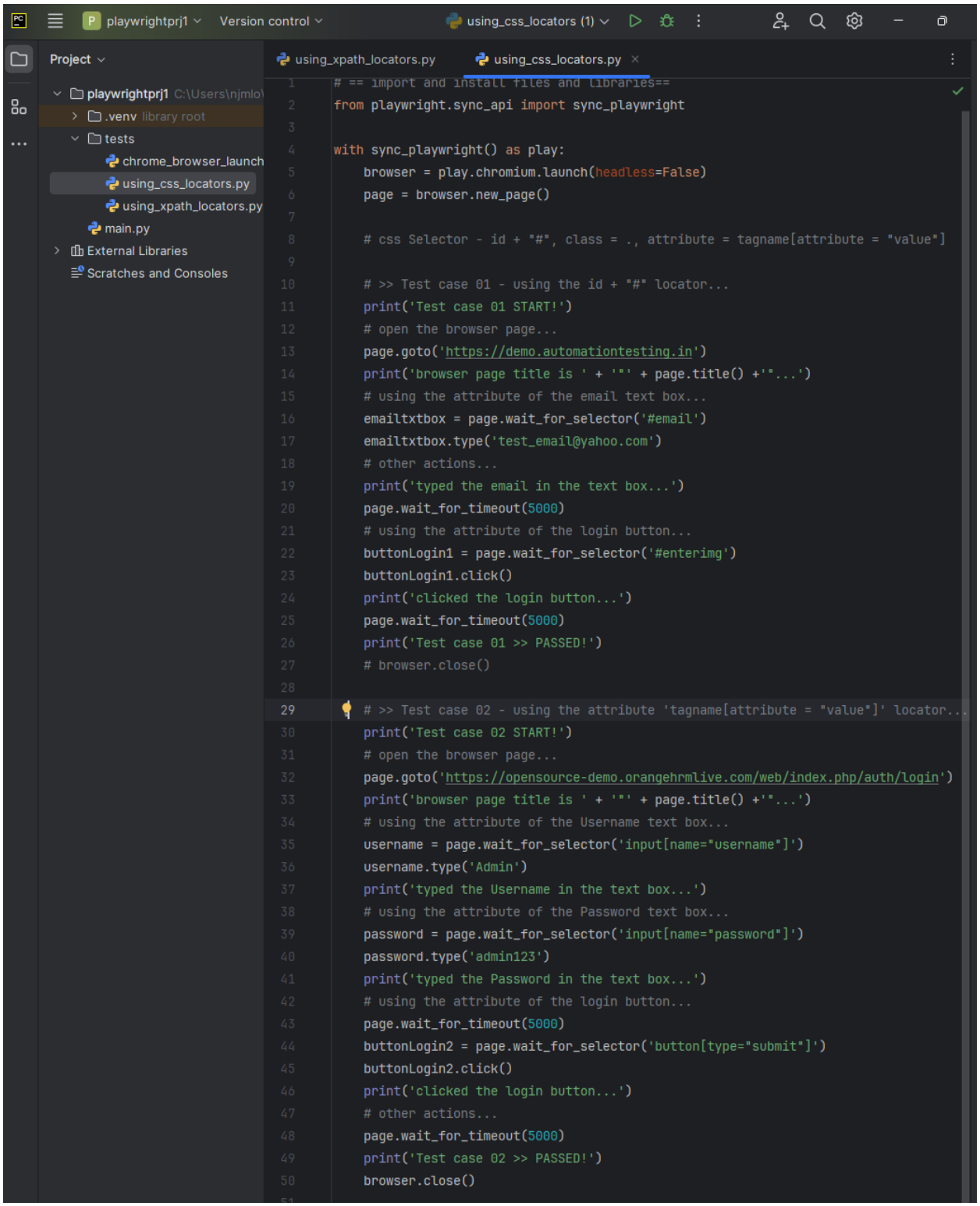
Login button…



Syntax:

b. (Test case 02) use the attribute 'tagname[attribute = "value"]'

Using the “attribute” locator, create and run the code:

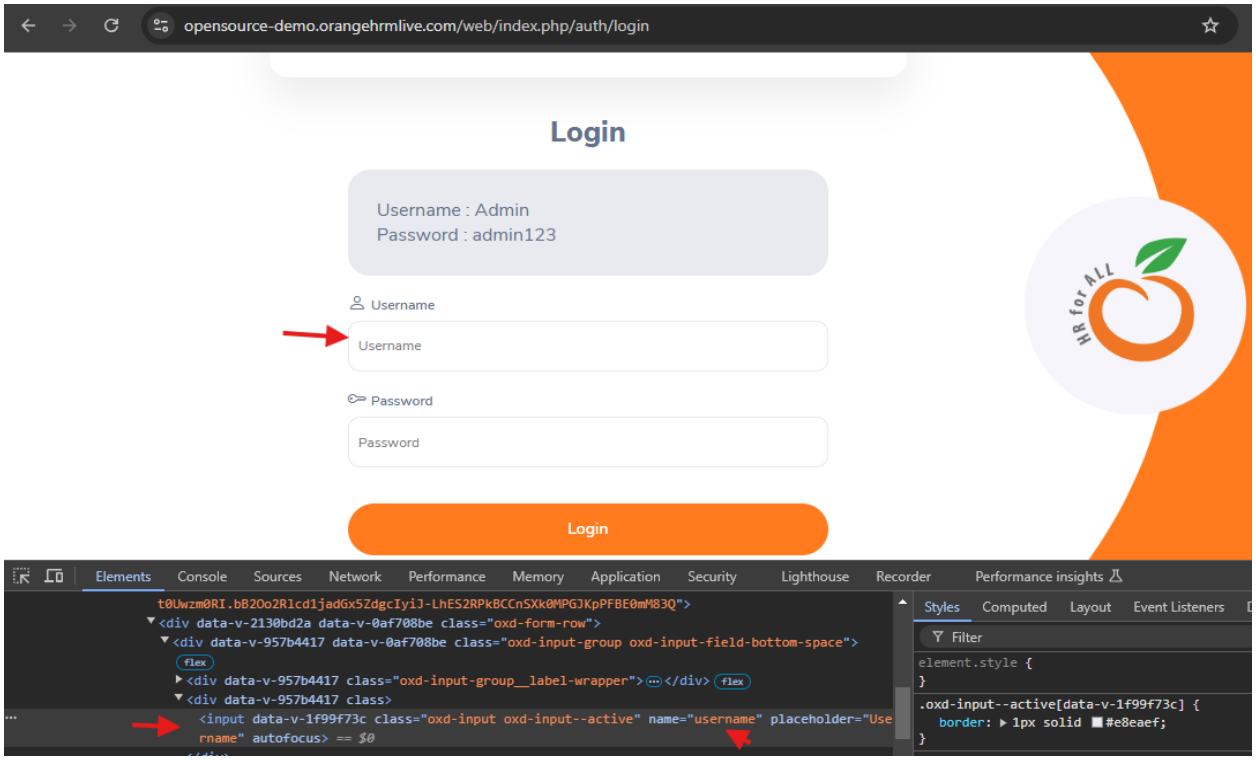


4. Working on a web page identifying element using XPath locators

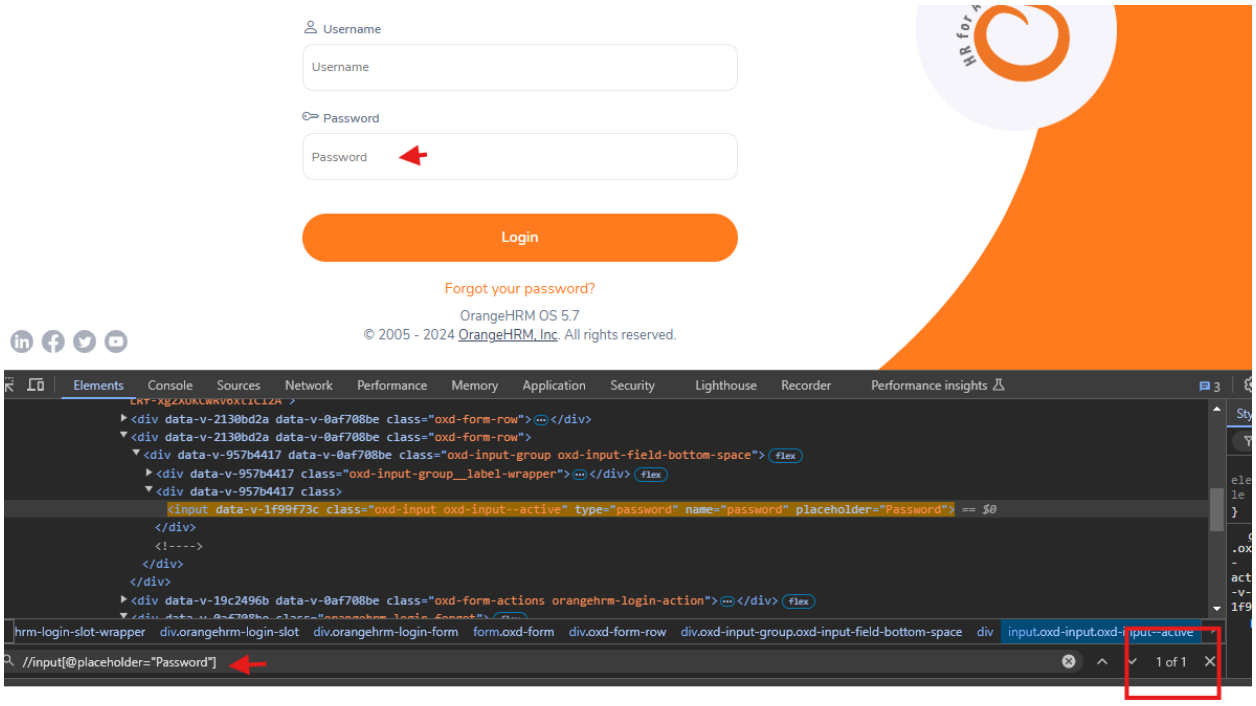
Test page url:
https://opensource-demo.orangehrmlive.com/web/index.php/auth/login

Locate (inspect) the element attributes for use as its XPath locator

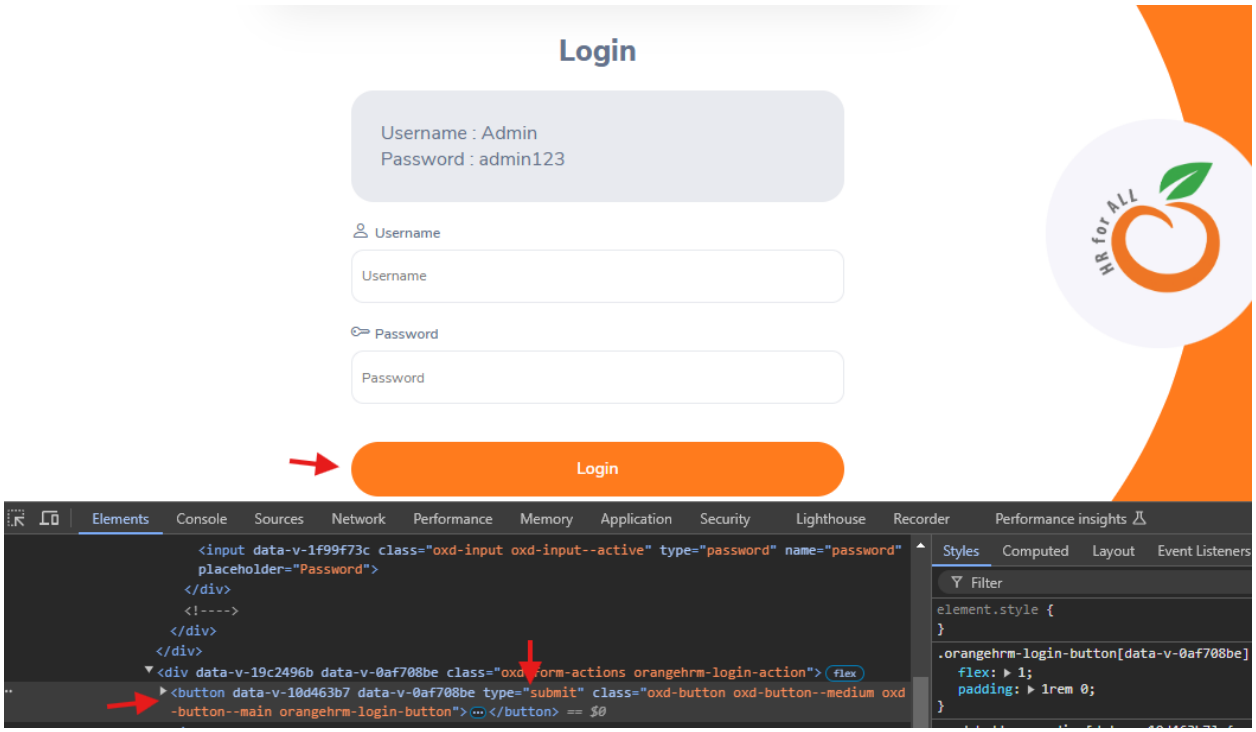
Username text box…



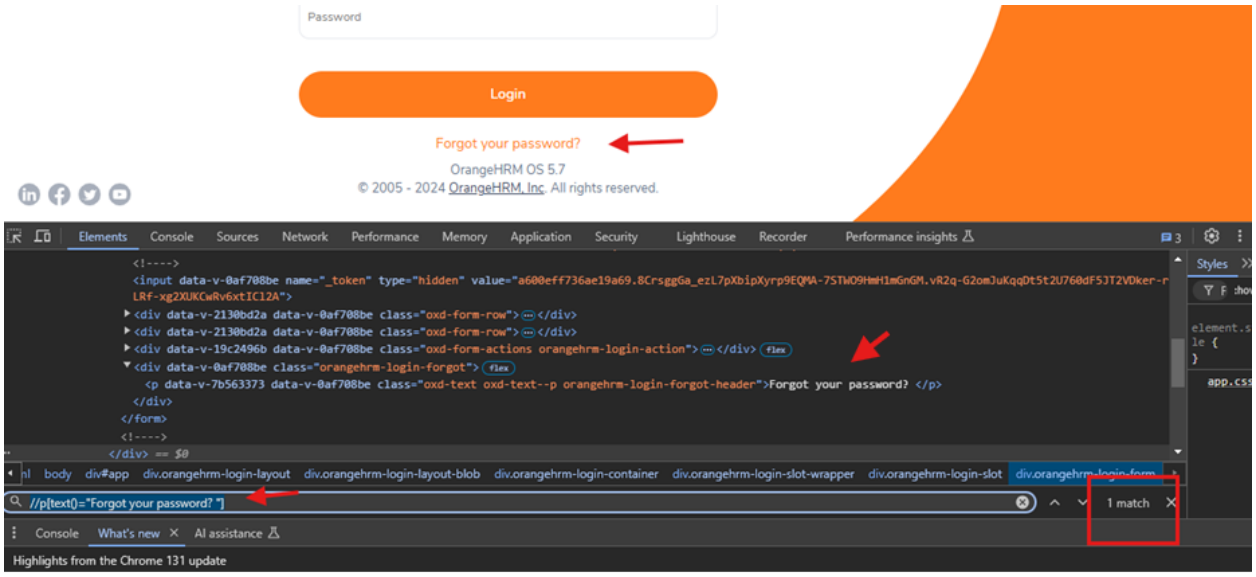
Password text box…



Login button…



Forgot your password link:

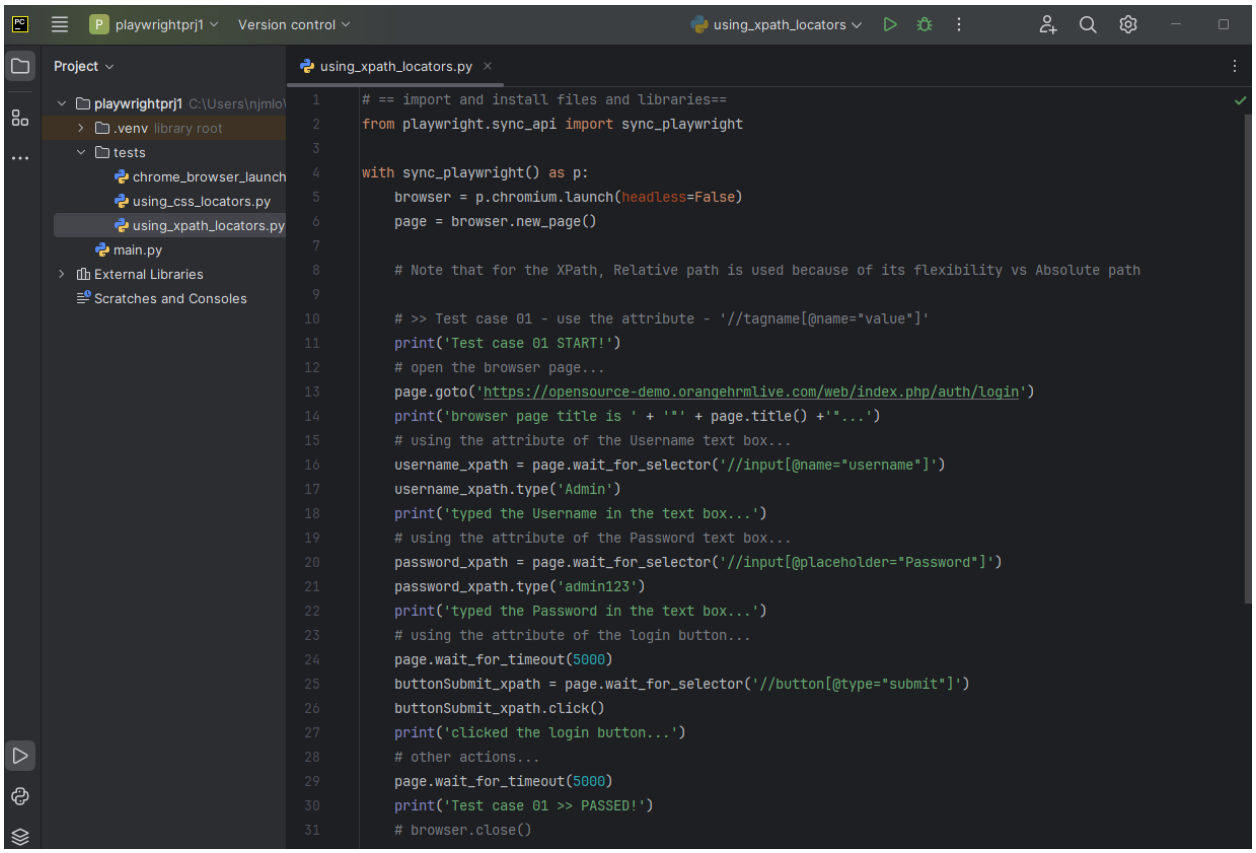


Syntax:

a. (Test case 01) use the attribute - '//tagname[@name="value"]'

b. (Test case 02) use the visible text attribute - '//tagname[text()="text"]'

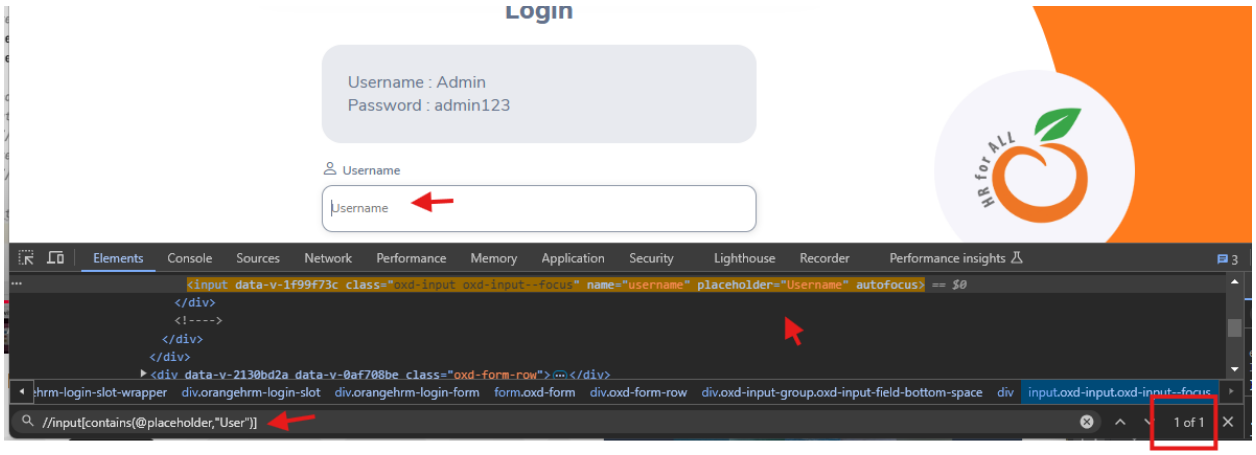
Using the XPath locator, create and run the code:



```python

    # >> Test case 02 - use the visible text attribute - '//tagname[text()="text"]'
    print('Test case 02 START!')
    # open the browser page...
    page = browser.new_page()
    page.goto('https://opensource-demo.orangehrmlive.com/web/index.php/auth/login')
    print('browser page title is ' + '"' + page.title() + '"...')
    # using the attribute of the Username text box...
    page.wait_for_timeout(5000)
    page.wait_for_selector('//p[text()="Forgot your password? "]').click()
    print('clicked the "Forgot your password?" link...')
    # other actions...
    page.wait_for_timeout(5000)
    print('Test case 02 >> PASSED!')
    browser.close()
```

```
Run  using_xpath_locators

C:\Users\njmlo\PycharmProjects\Playwright\playwrightprj1\.venv\Scripts\python.exe C:\Users\njmlo\PycharmProjects\Playw
Test case 01 START!
browser page title is "OrangeHRM"...
typed the Username in the text box...
typed the Password in the text box...
clicked the login button...
Test case 01 >> PASSED!
Test case 02 START!
browser page title is "OrangeHRM"...
clicked the "Forgot your password?" link...
Test case 02 >> PASSED!

Process finished with exit code 0
```

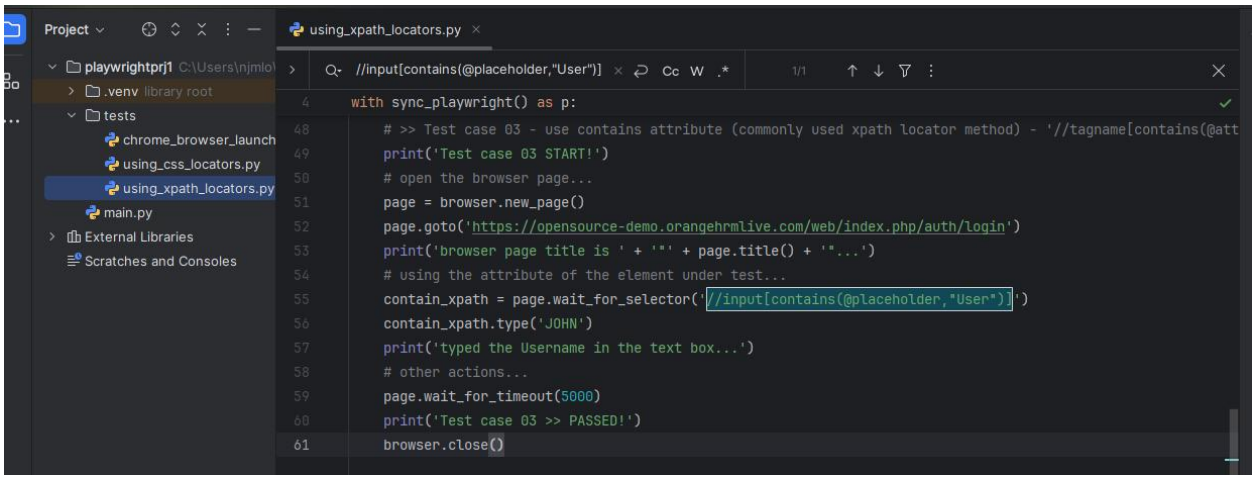
Username text box...



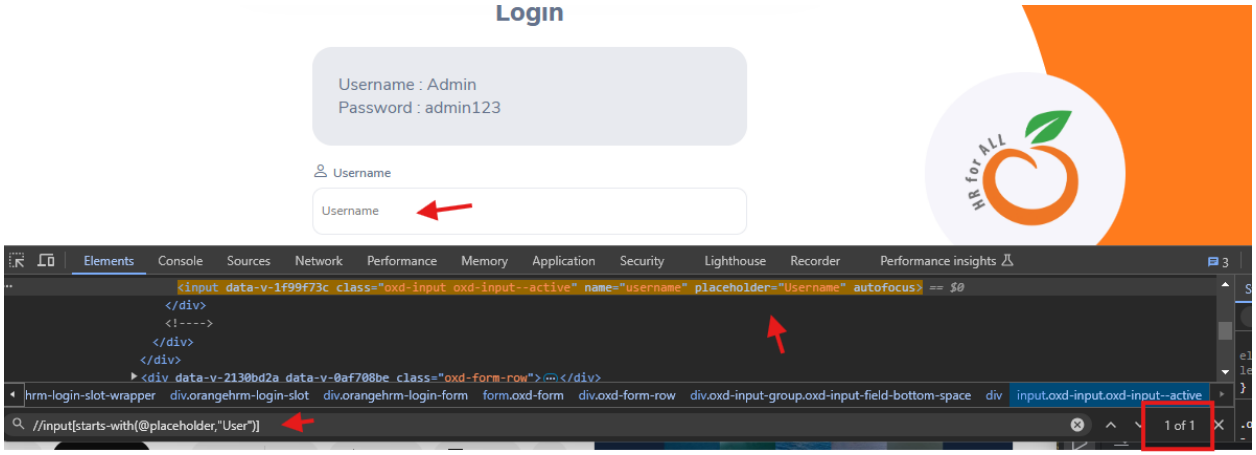
Syntax:

c. (Test case 03) use contains attribute (commonly used xpath locator method) - '//tagname[contains(@attribute, "value")]'

Using the XPath locator, create and run the code:



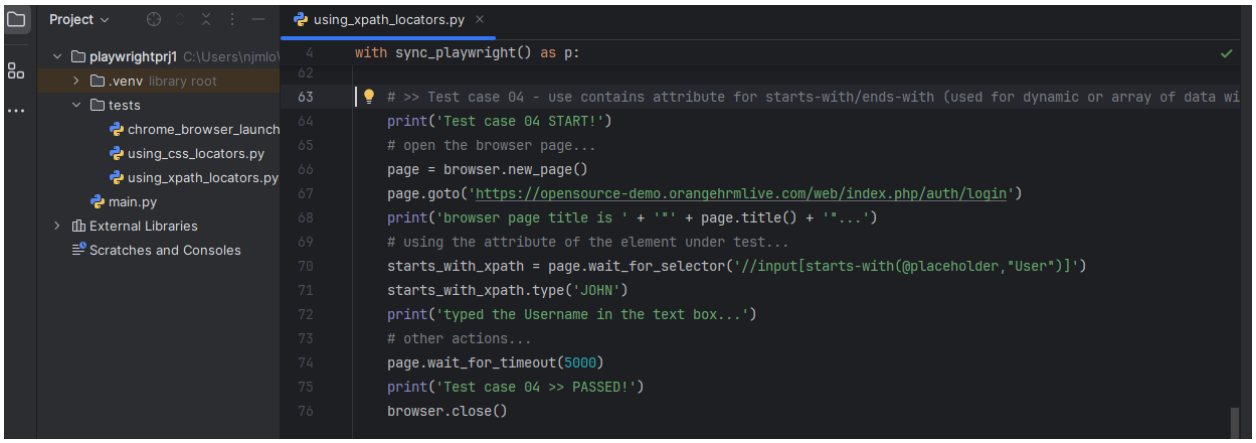
Username text box…



Syntax:

c. (Test case 04) use contains attribute for starts-with/ends-with (used for dynamic or array of data with common value) - //tagname[starts-with(@attribute, "value")]

Using the XPath locator, create and run the code:



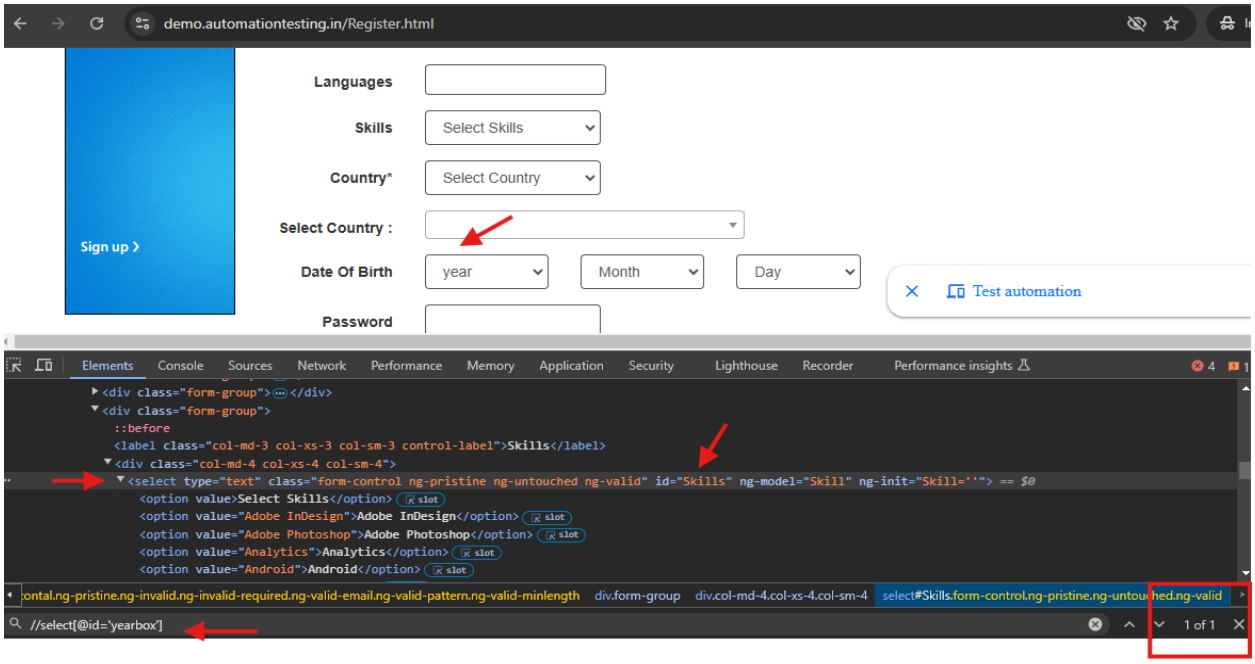
Samples of common XPath locators:

- `/bookstore/book` - Selects all book elements in the root bookstore.
- `//title[text()='XPath']` - Selects title elements with text 'XPath' anywhere in the document.
- `//*[@id='myId']` - Selects elements with the attribute id equal to 'myId'.
- `/bookstore/book[position()=1]` - Selects the first book element.
- `//div[@class='highlight']//p` - Selects p elements within div with class 'highlight'.
- `//a[contains(@href, 'example.com')]` - Selects a elements with 'example.com' in the href attribute.

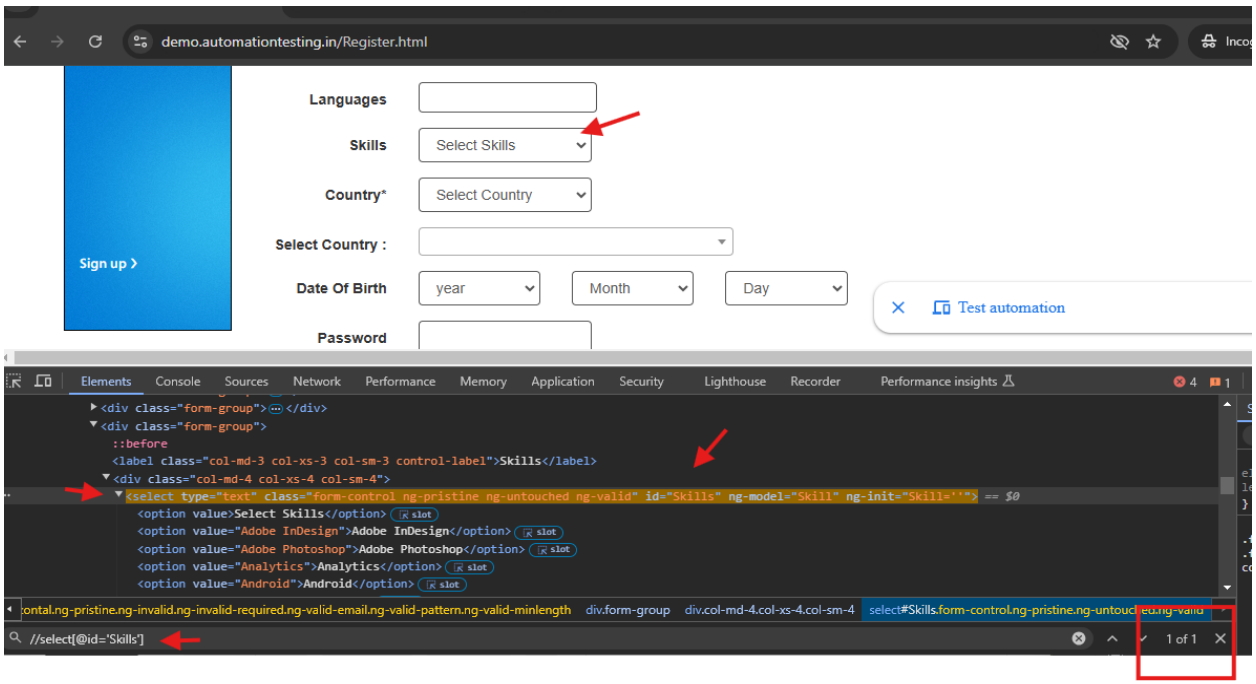
5. Working on a web page in selecting values from dropdowns

Test page url:
https://demo.automationtesting.in/

Locate (inspect) the element attributes for use as its locator:
Skills dropdown…



DOB dropdown…

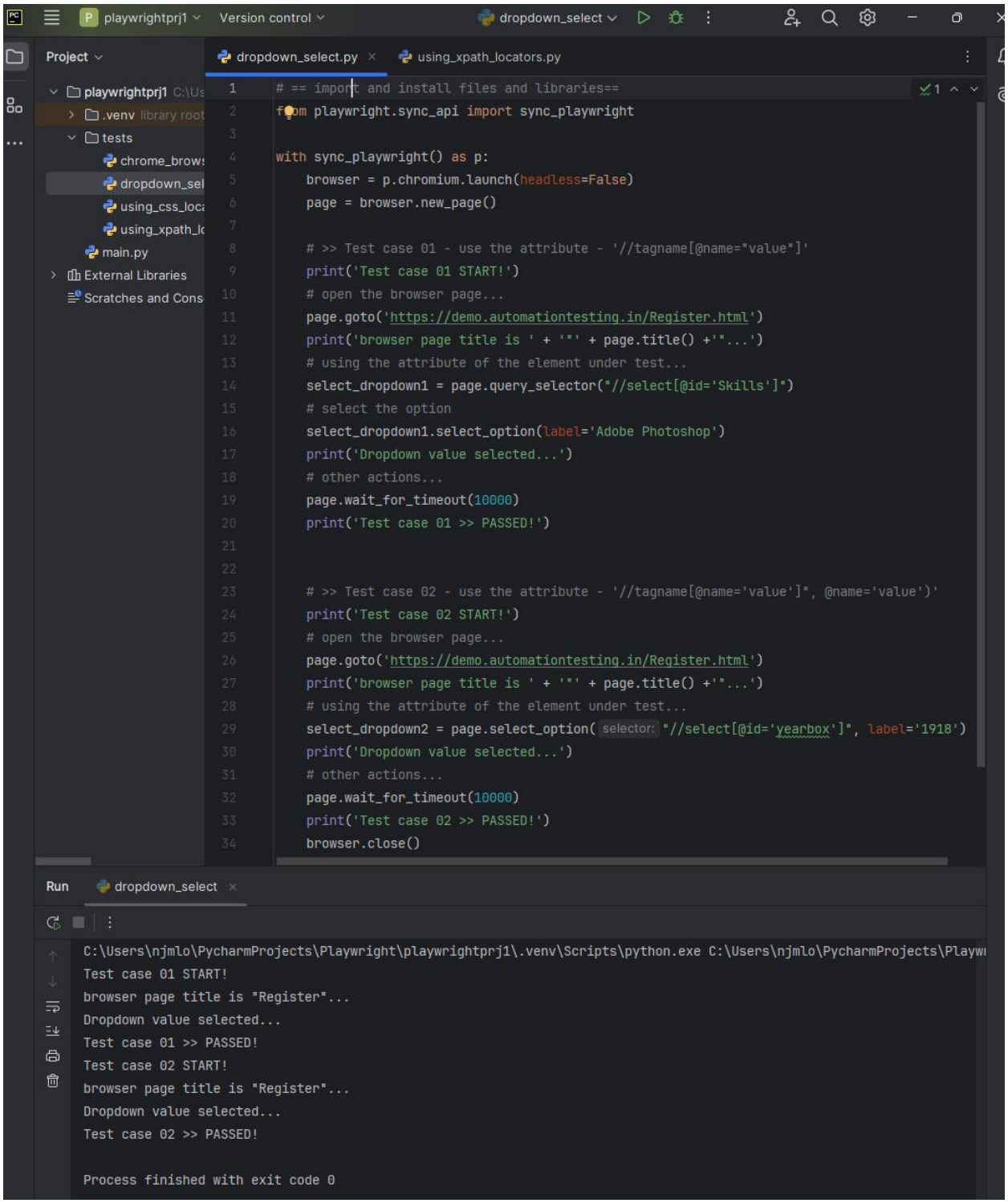


Syntax:

a. Test case 01 - use the attribute - '//tagname[@name="value"]'

b. Test case 02 - use the attribute - '//tagname[@name='value']", @name='value')'

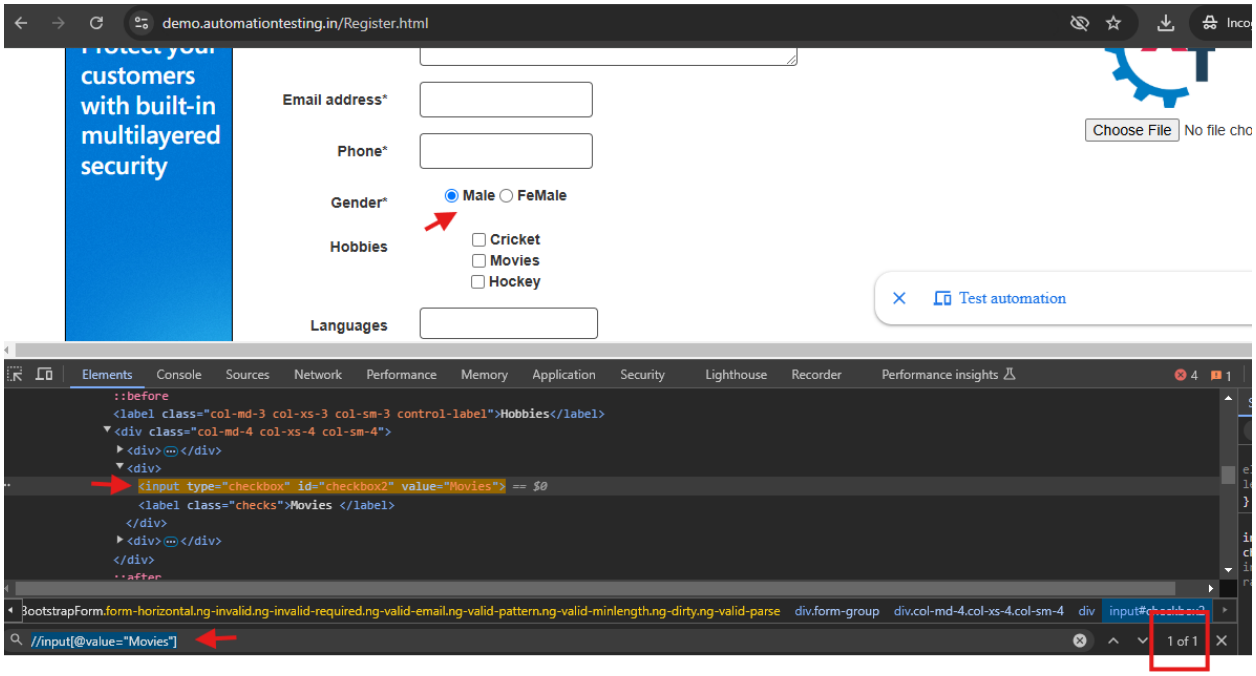
Using the XPath locator, create and run the code:



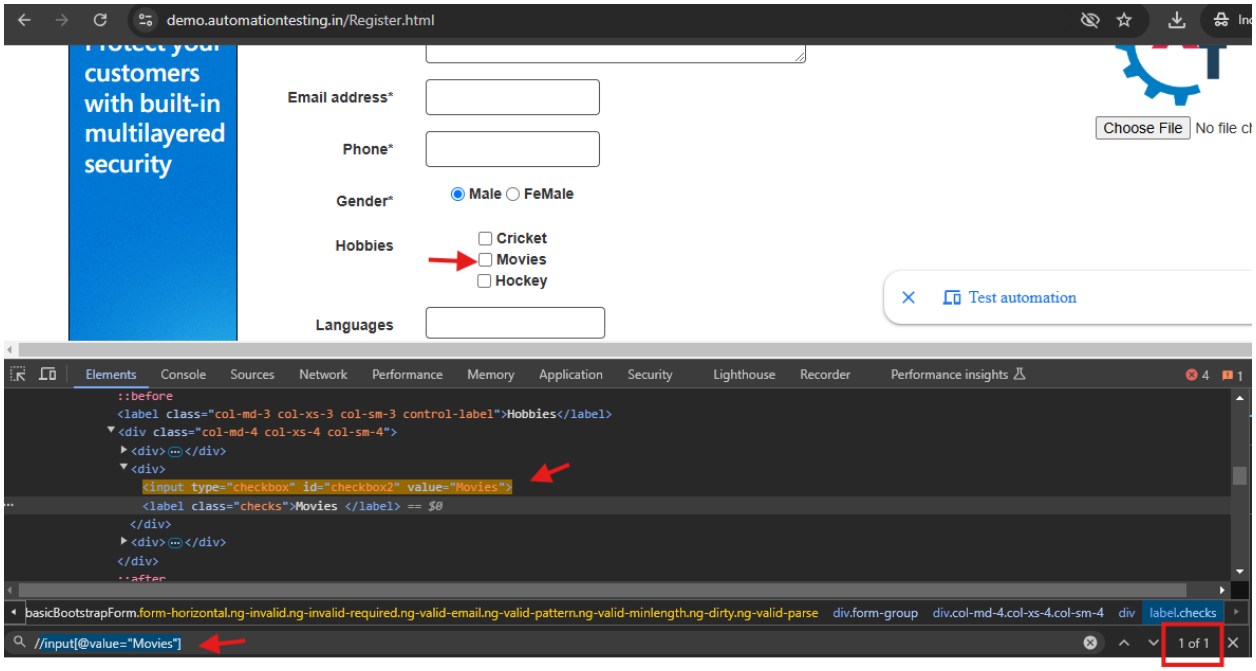
6. Working on a web page in selecting radio buttons and checkboxes

Test page url:
https://demo.automationtesting.in/Register.html

Locate (inspect) the element attributes for use as its locator:
Gender radio button…



Hobbies checkbox…

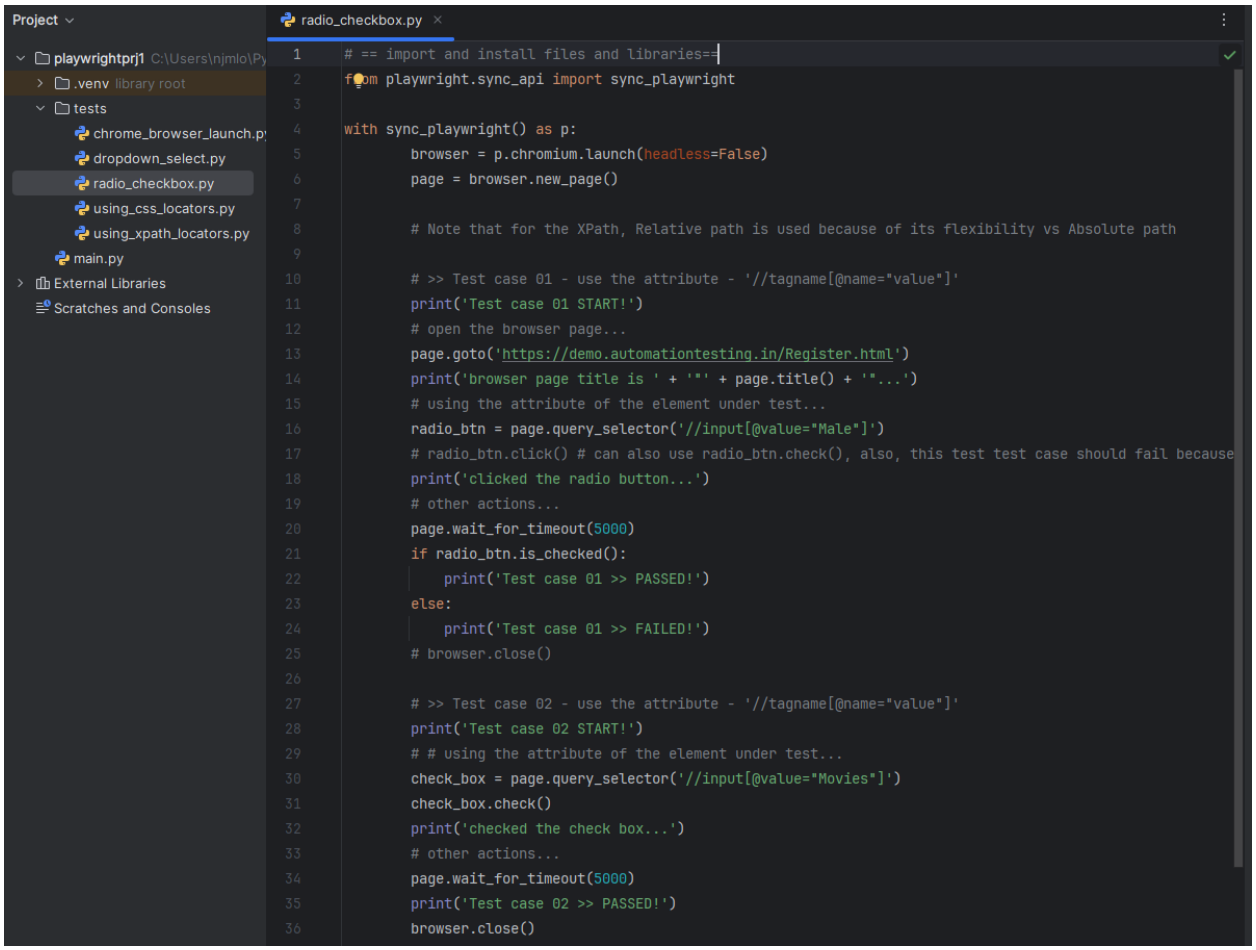


Syntax:

a. Test case 01 - use the attribute - '//tagname[@name="value"]'

b. Test case 02 - use the attribute - '//tagname[@name="value"]'

Using the XPath locator, create and run the code:

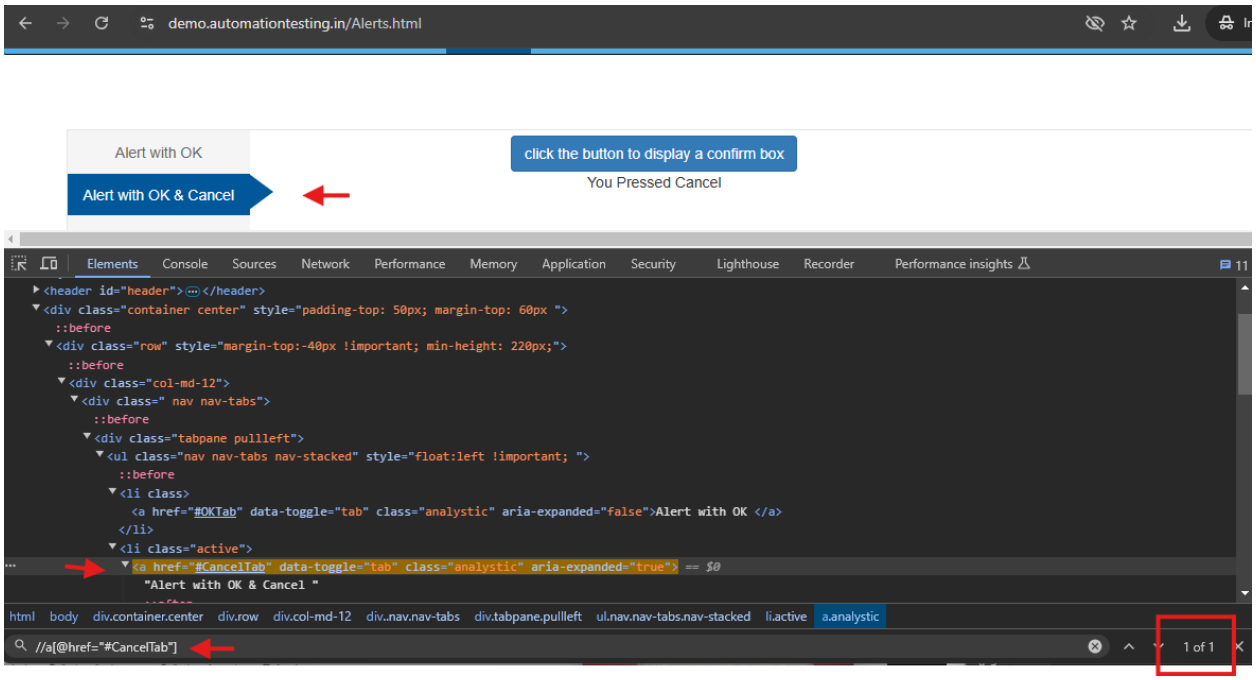


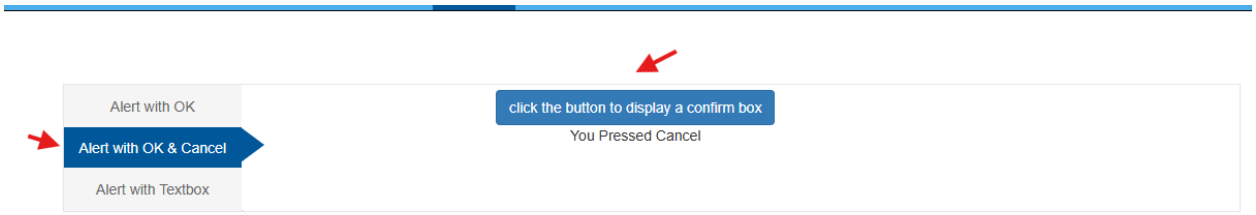
7. Working on a web page on alerts and dialog box

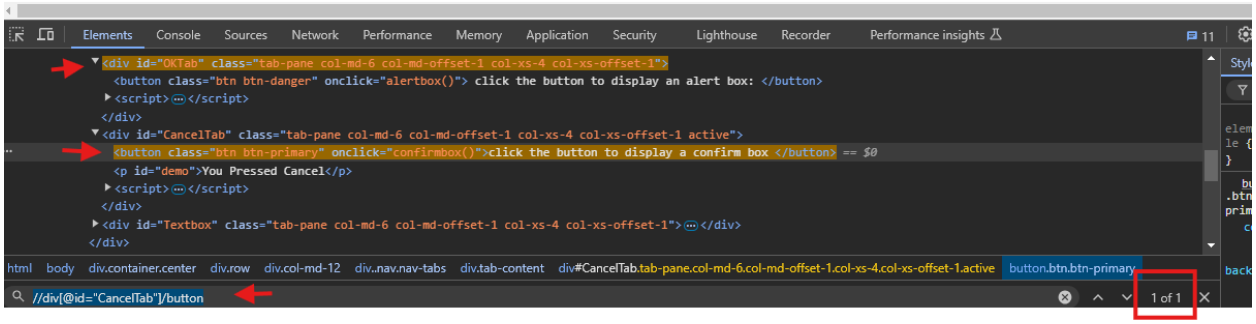
Test page url:
https://demo.automationtesting.in/Alerts.html

Locate (inspect) the element attributes for use as its locator:

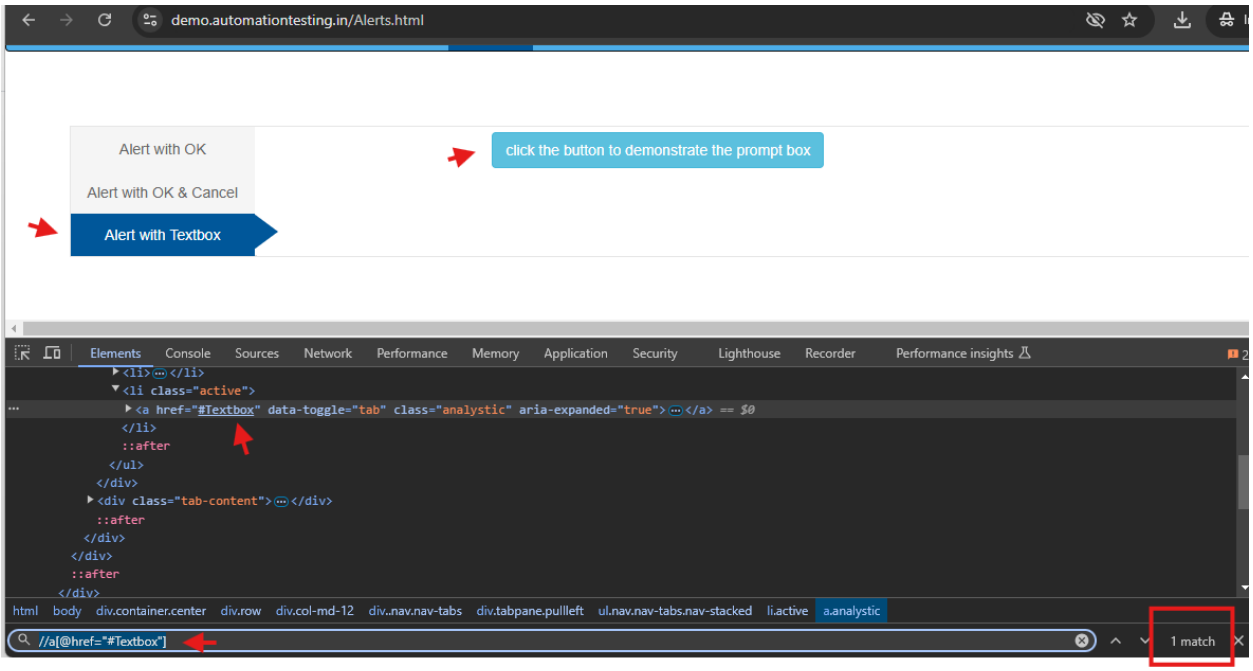
Alert screen options for the Alerts with OK & Cancel dialog popup window…

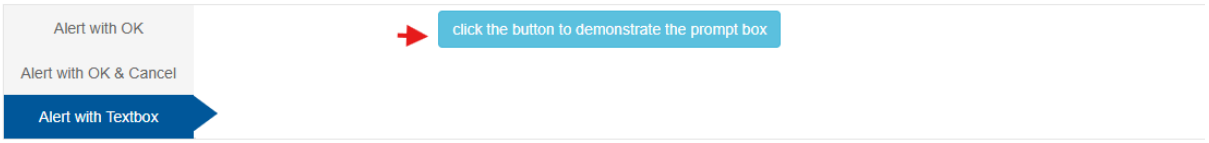


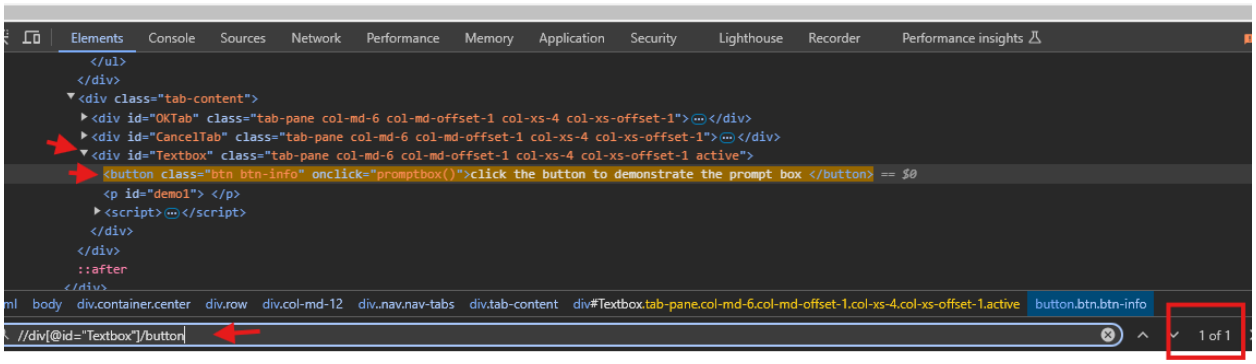




Alert screen options for the Alerts with OK & Cancel dialog popup window…





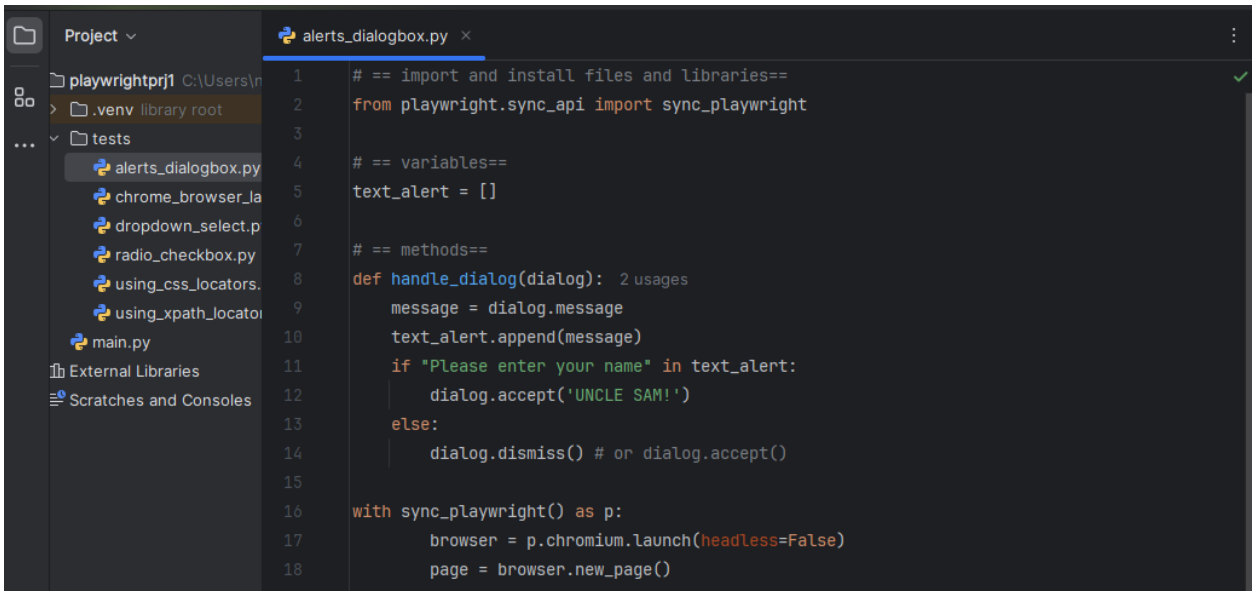


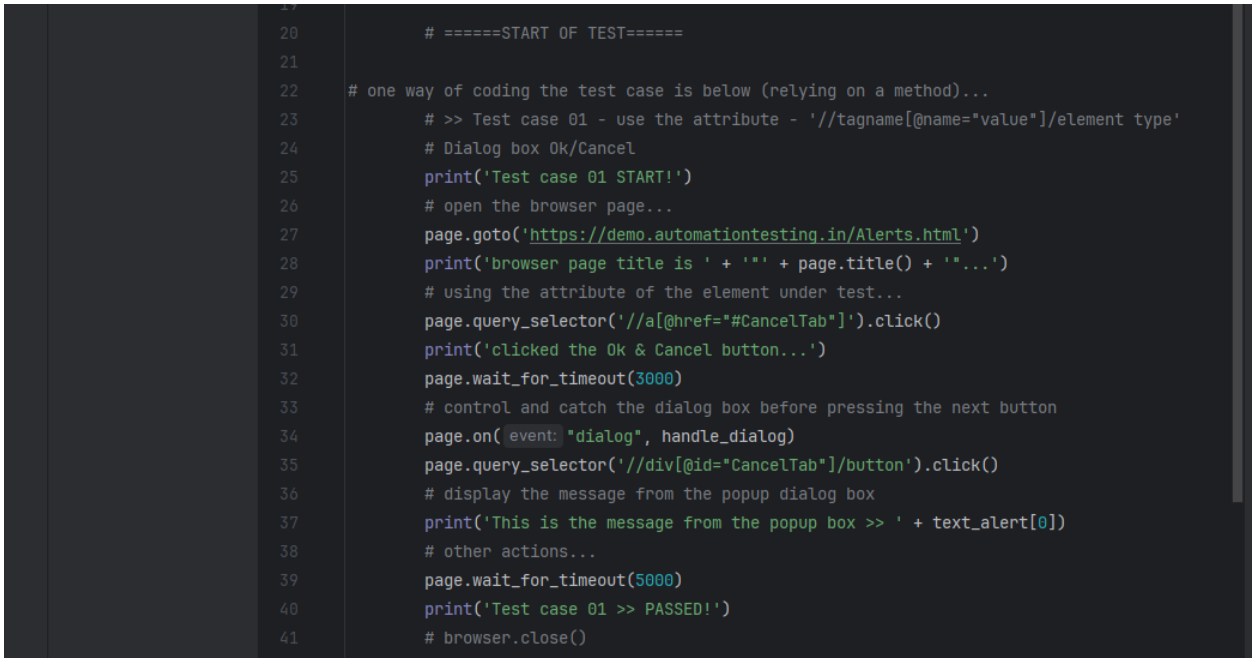
Syntax:

a. Test case 01 - use the attribute - '//tagname[@name="value"]/child tagname'', dialog box Ok/Cancel

a. Test case 02 - use the attribute - '//tagname[@name="value"]/child tagname'', dialog box with Textbox

Using the XPath locator, create and run the code:





```python
        # >> Test case 02 - use the attribute - '//tagname[@name="value"]/element type'
        # Dialog box with Textbox
        print('Test case 02 START!')
        # open the browser page...
        page.goto('https://demo.automationtesting.in/Alerts.html')
        print('browser page title is ' + '"' + page.title() + '"...')
        # using the attribute of the element under test...
        page.query_selector('//a[@href="#Textbox"]').click()
        print('clicked the Alerts with Textbox button...')
        page.wait_for_timeout(3000)
        # control and catch the dialog box before pressing the next button
        page.on( event: "dialog", handle_dialog)
        page.query_selector('//div[@id="Textbox"]/button').click()
        # display the message from the popup dialog box
        print('This is the message from the popup box >> ' + text_alert[1])
        # other actions...
        page.wait_for_timeout(5000)
        print('Test case 02 >> PASSED!')
        browser.close()
```

Run alerts_dialogbox

```
C:\Users\njmlo\PycharmProjects\Playwright\playwrightprj1\.venv\Scripts\python.exe C:\Users\njmlo\PycharmProjects\Playwr
Test case 01 START!
browser page title is "Alerts"...
clicked the Ok & Cancel button...
This is the message from the popup box >> Press a Button !
Test case 01 >> PASSED!
Test case 02 START!
browser page title is "Alerts"...
clicked the Alerts with Textbox button...
This is the message from the popup box >> Please enter your name
Test case 02 >> PASSED!

Process finished with exit code 0
```

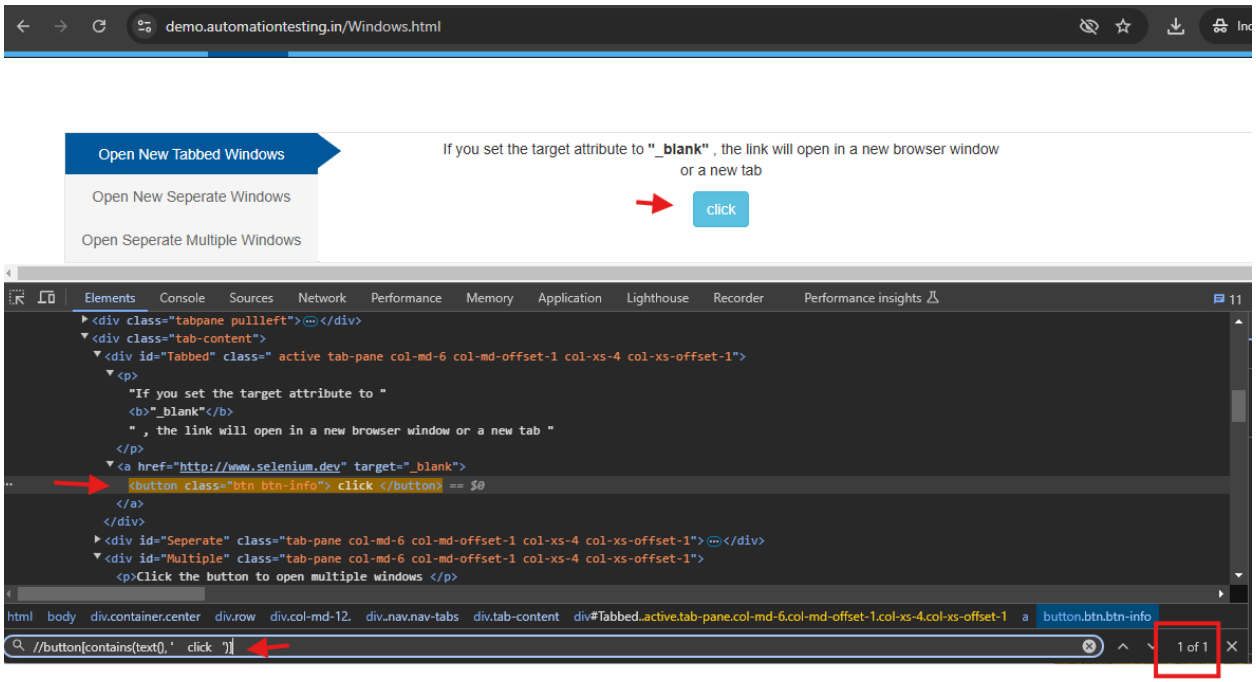
aywrightprj1 › tests › alerts_dialogbox.py 65:5 CRLF UTF-8 4 spaces Python 3.12 (playwrightprj1)

8. Working on a web page windows/page handling

Test page url:
https://demo.automationtesting.in/Windows.html

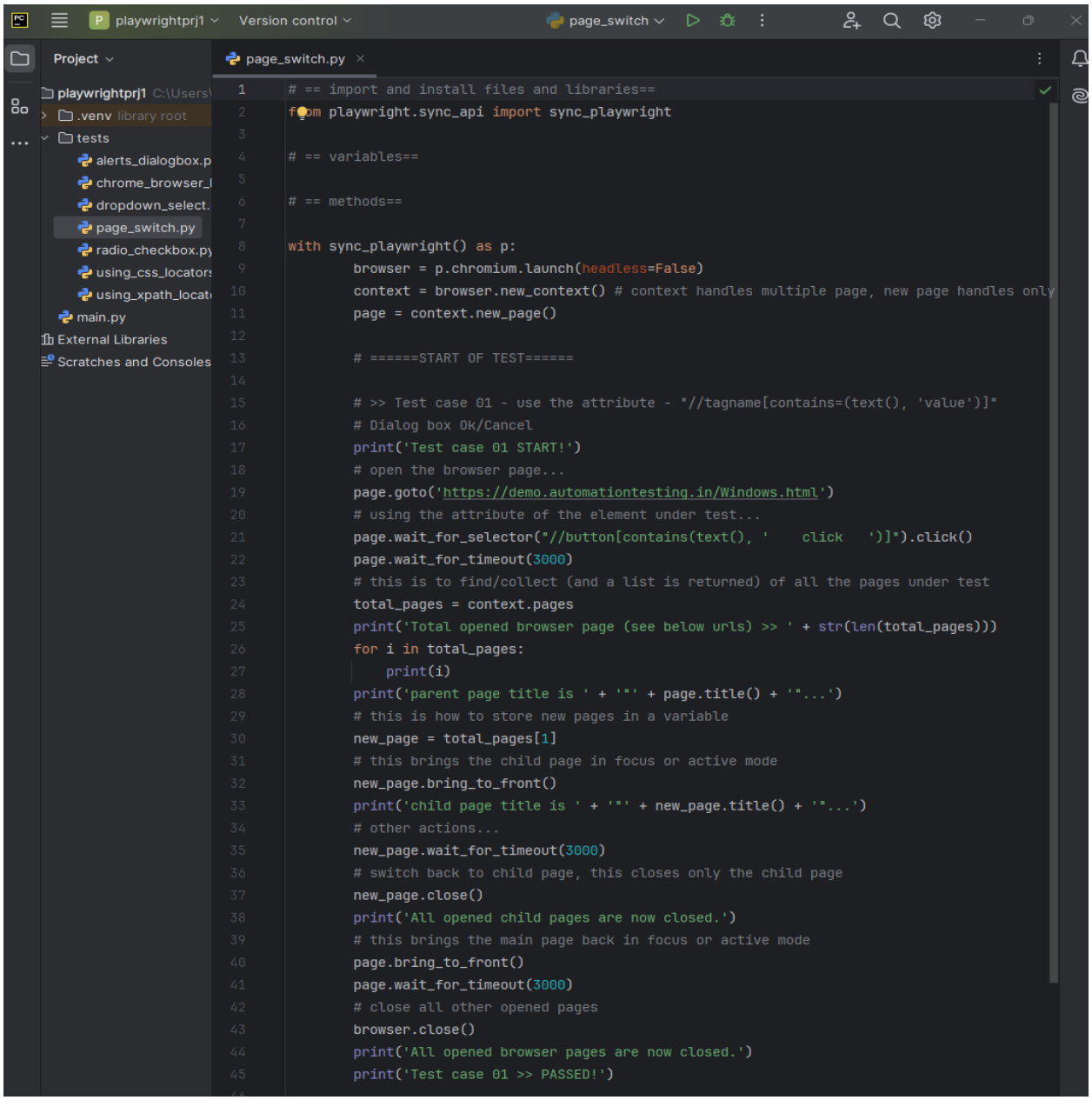
Locate (inspect) the element attributes for use as its locator:

Open new tabbed windows click button…



Syntax:
a. Test case 01 - use the attribute - "//tagname[contains=(text(), 'value')]"

Using the XPath locator, create and run the code:

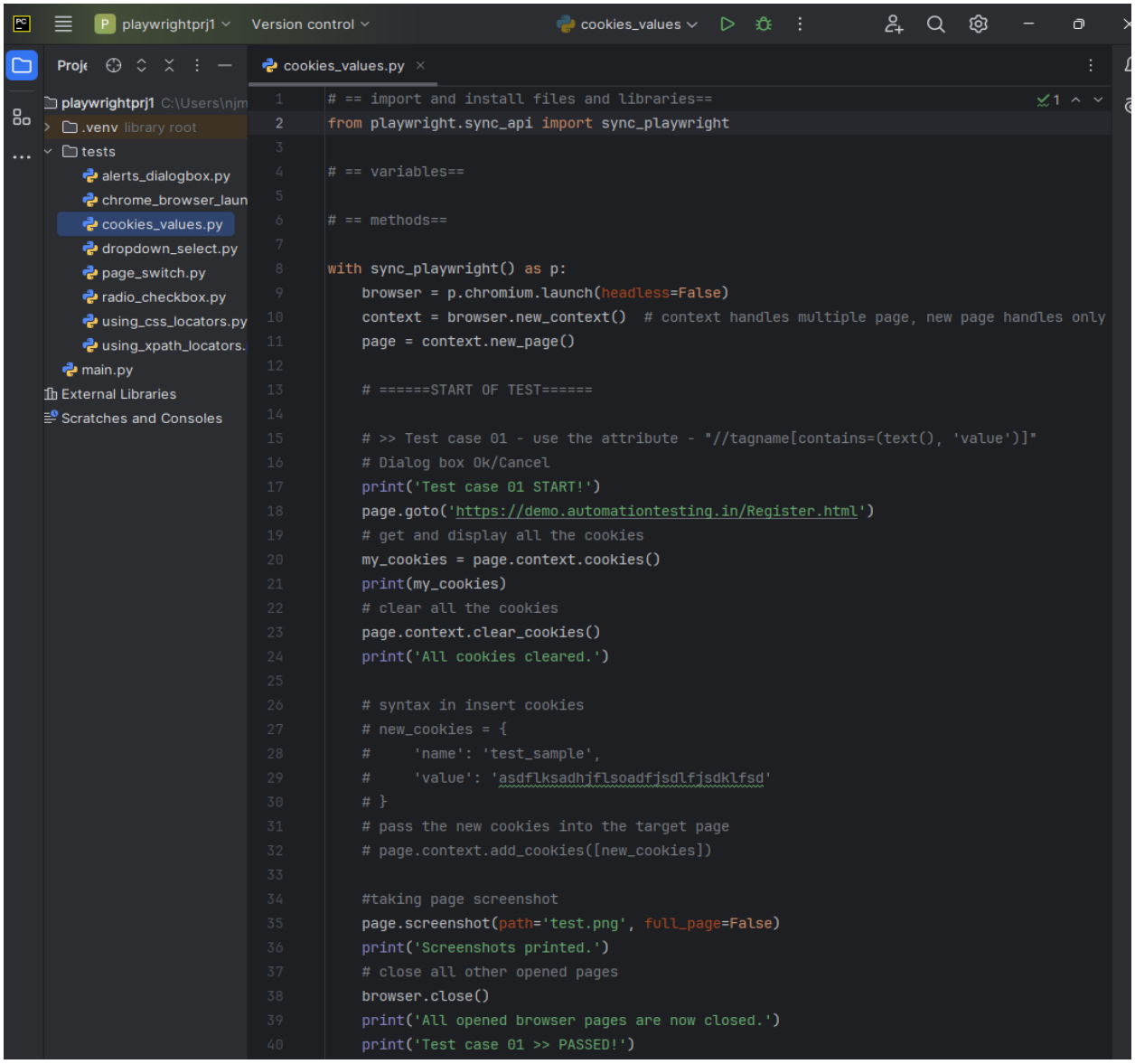


```
# == import and install files and libraries==
from playwright.sync_api import sync_playwright

# == variables==

# == methods==

with sync_playwright() as p:
        browser = p.chromium.launch(headless=False)
        context = browser.new_context() # context handles multiple page, new page handles only
        page = context.new_page()

        # ======START OF TEST======

        # >> Test case 01 - use the attribute - "//tagname[contains=(text(), 'value')]"
        # Dialog box Ok/Cancel
        print('Test case 01 START!')
        # open the browser page...
        page.goto('https://demo.automationtesting.in/Windows.html')
        # using the attribute of the element under test...
        page.wait_for_selector("//button[contains(text(), '    click   ')]").click()
        page.wait_for_timeout(3000)
        # this is to find/collect (and a list is returned) of all the pages under test
        total_pages = context.pages
        print('Total opened browser page (see below urls) >> ' + str(len(total_pages)))
        for i in total_pages:
            print(i)
        print('parent page title is ' + '"' + page.title() + '"...')
        # this is how to store new pages in a variable
        new_page = total_pages[1]
        # this brings the child page in focus or active mode
        new_page.bring_to_front()
        print('child page title is ' + '"' + new_page.title() + '"...')
        # other actions...
        new_page.wait_for_timeout(3000)
        # switch back to child page, this closes only the child page
        new_page.close()
        print('All opened child pages are now closed.')
        # this brings the main page back in focus or active mode
        page.bring_to_front()
        page.wait_for_timeout(3000)
        # close all other opened pages
        browser.close()
        print('All opened browser pages are now closed.')
        print('Test case 01 >> PASSED!')
```

```
C:\Users\njmlo\PycharmProjects\Playwright\playwrightprj1\.venv\Scripts\python.exe C:\Users\njmlo\PycharmProjects\Playwr
Test case 01 START!
Total opened browser page (see below urls) >> 2
<Page url='https://demo.automationtesting.in/Windows.html'>
<Page url='https://www.selenium.dev/'>
parent page title is "Frames & windows"...
child page title is "Selenium"...
All opened child pages are now closed.
All opened browser pages are now closed.
Test case 01 >> PASSED!

Process finished with exit code 0
```

9. Working on a web page cookies and taking page screenshots

Test page url:
https://demo.automationtesting.in/Register.html

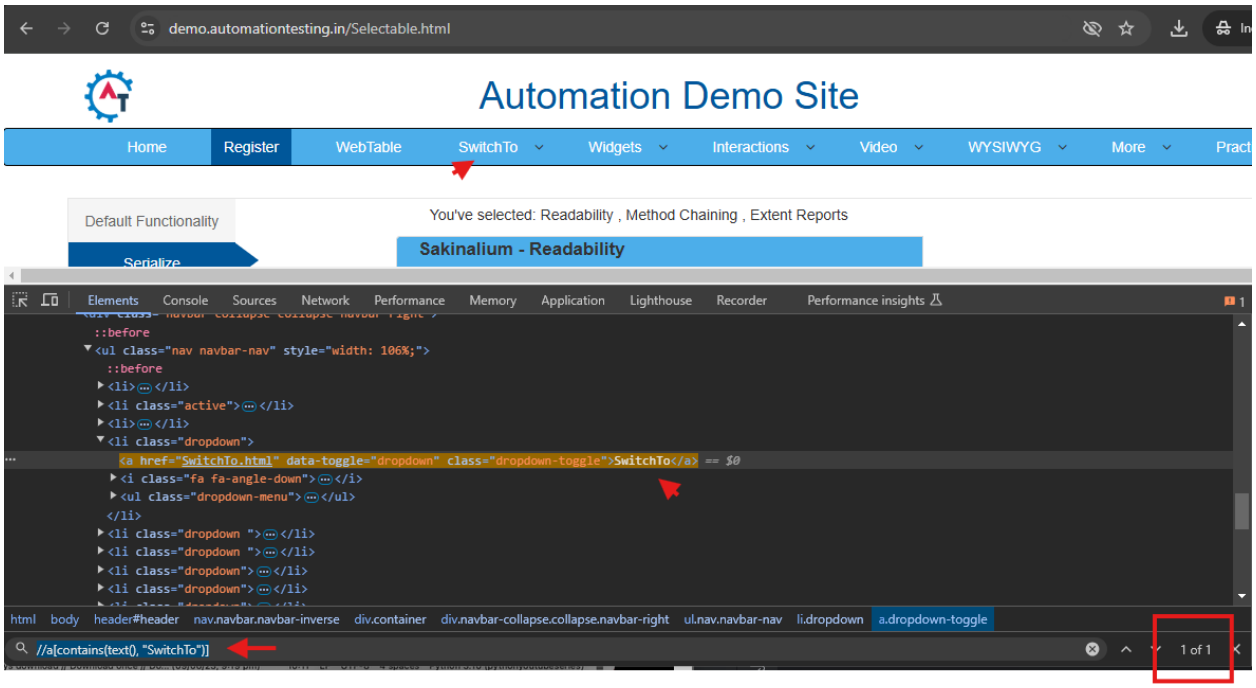


10. Working on web page interactions

Test page url:
https://demo.automationtesting.in/Selectable.html

Locate (inspect) the element attributes for use as its locator:

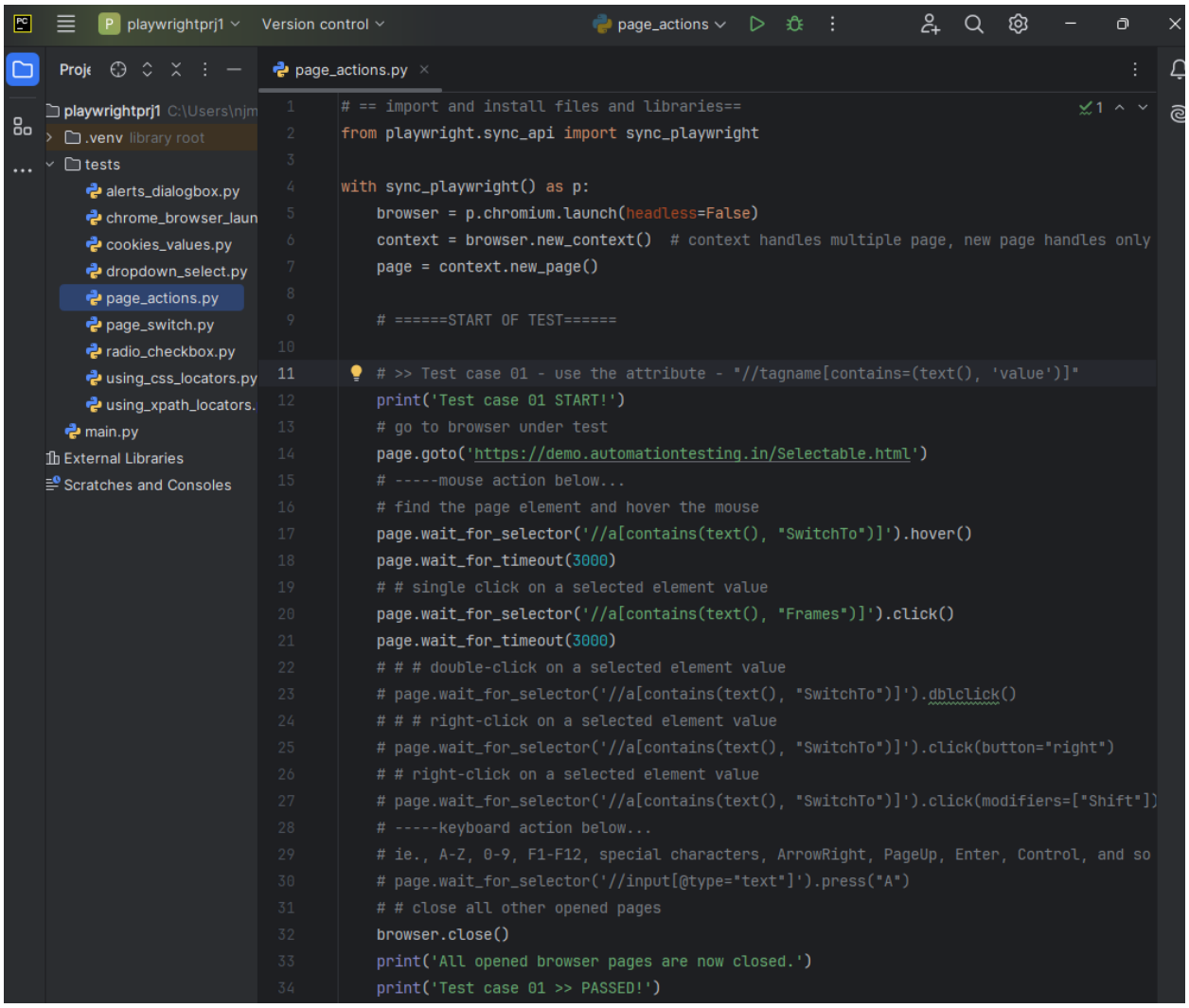
Switch To list men …

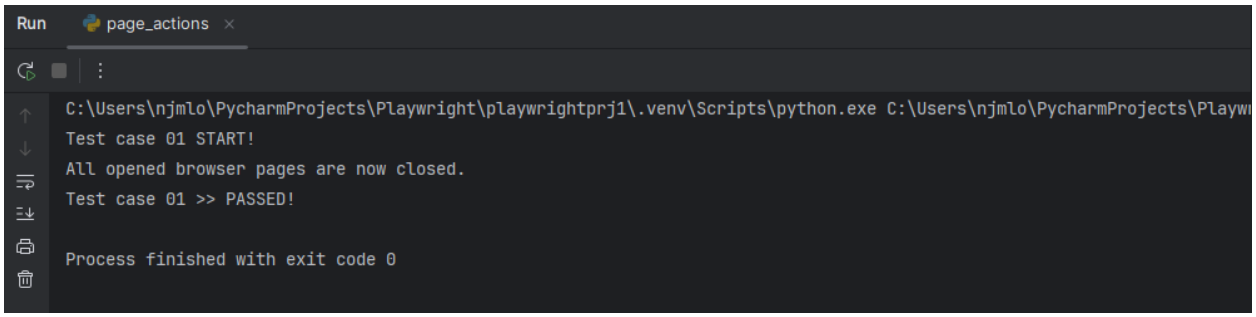


Syntax:

a. Test case 01 - use the attribute - "//tagname[contains=(text(), 'value')]"

Using the XPath locator, create and run the code:





11. Working on web page exception handling and storing elements in lists

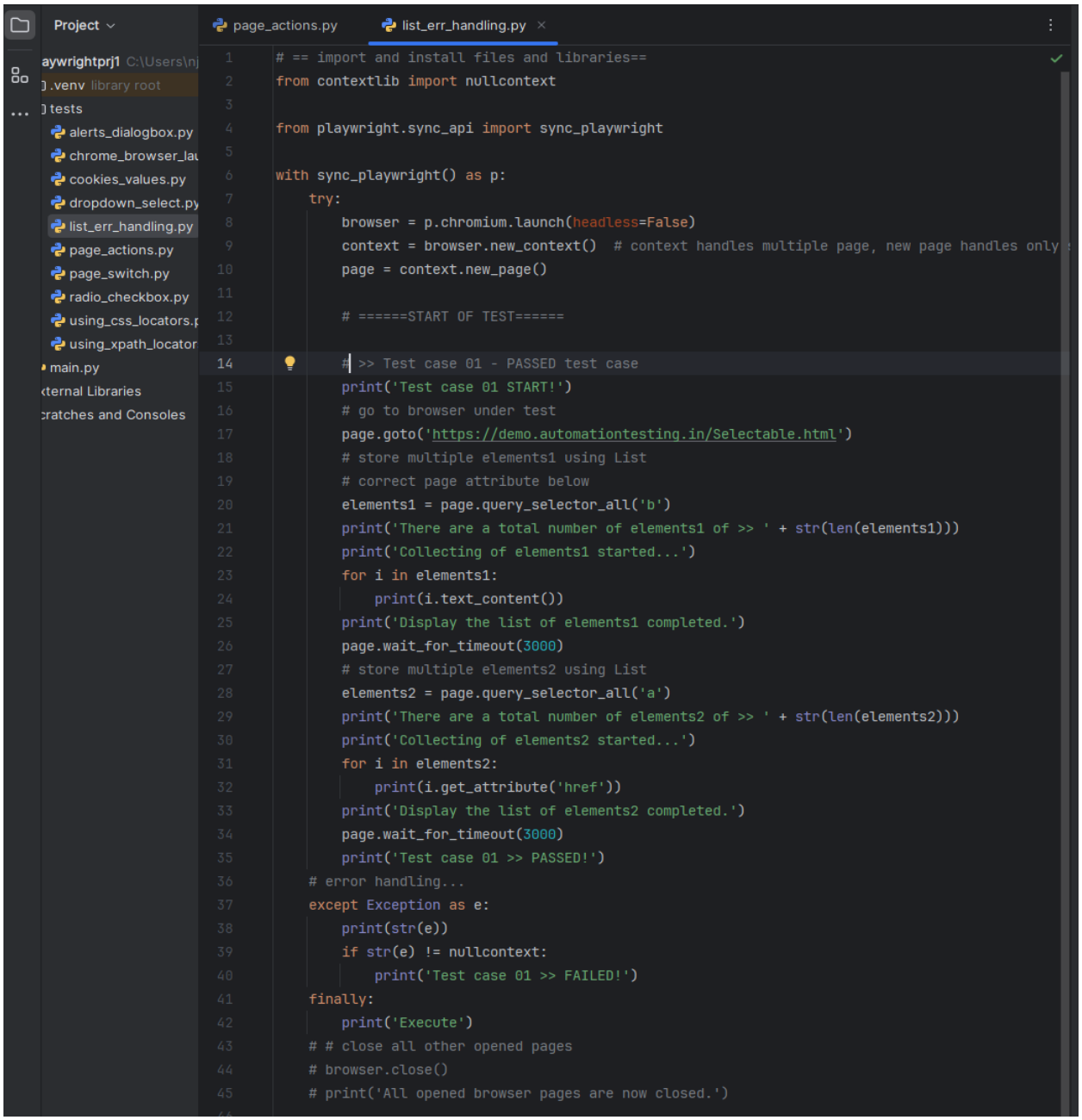
Test page url:
https://demo.automationtesting.in/Selectable.html

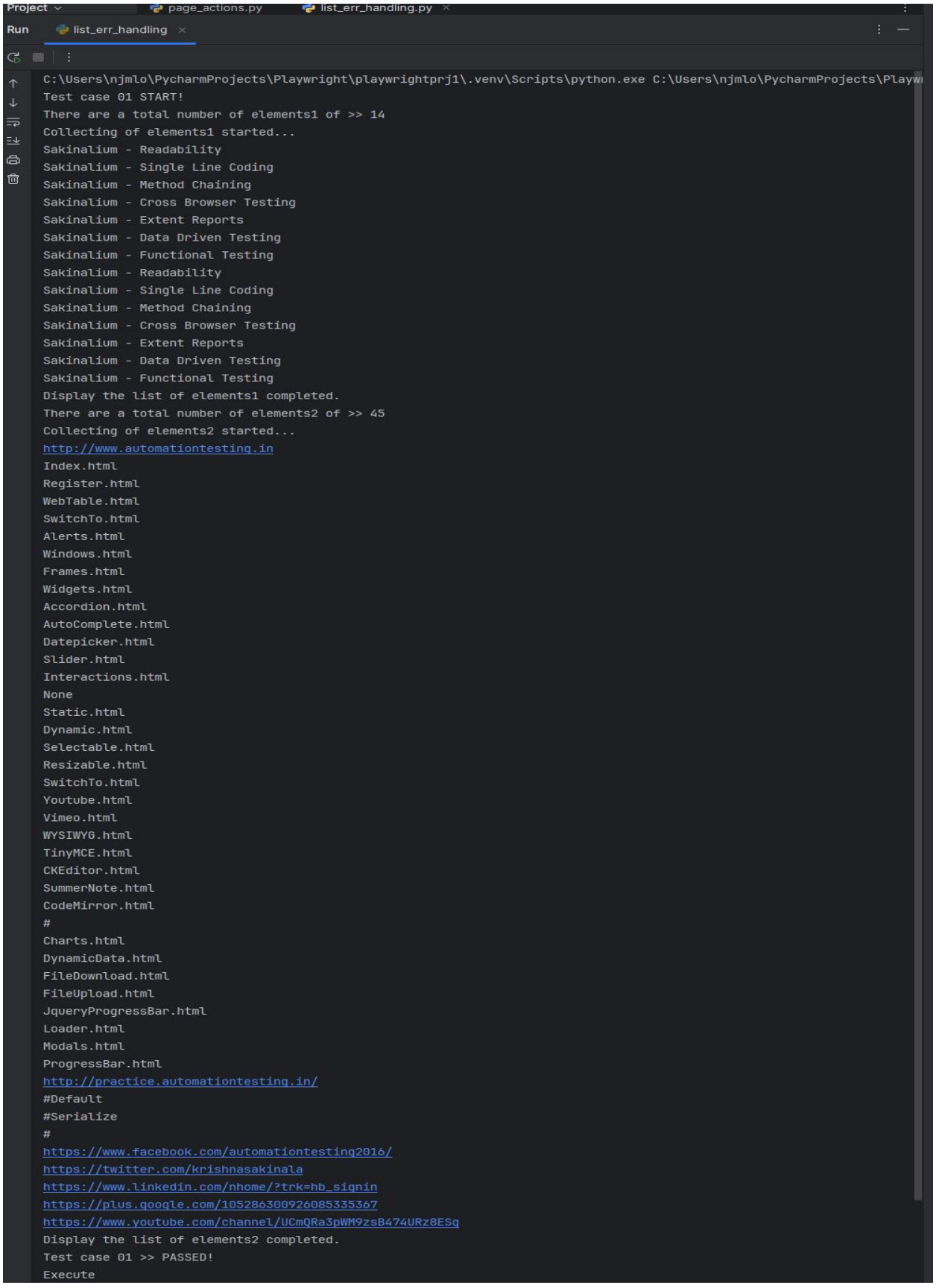
Syntax:

a. Test case 01 – PASSED test case

b. Test case 02 – FAILED test case (to test error handling syntax)

Using the locator, create and run the code:




Project
page_actions.py
list_err_handling.py
Run
list_err_handling
C:\Users\njmlo\PycharmProjects\Playwright\playwrightprj1\.venv\Scripts\python.exe C:\Users\njmlo\PycharmProjects\Playwr
Test case 01 START!
There are a total number of elements1 of >> 14
Collecting of elements1 started...
Sakinalium - Readability
Sakinalium - Single Line Coding
Sakinalium - Method Chaining
Sakinalium - Cross Browser Testing
Sakinalium - Extent Reports
Sakinalium - Data Driven Testing
Sakinalium - Functional Testing
Sakinalium - Readability
Sakinalium - Single Line Coding
Sakinalium - Method Chaining
Sakinalium - Cross Browser Testing
Sakinalium - Extent Reports
Sakinalium - Data Driven Testing
Sakinalium - Functional Testing
Display the list of elements1 completed.
There are a total number of elements2 of >> 45
Collecting of elements2 started...
http://www.automationtesting.in
Index.html
Register.html
WebTable.html
SwitchTo.html
Alerts.html
Windows.html
Frames.html
Widgets.html
Accordion.html
AutoComplete.html
Datepicker.html
Slider.html
Interactions.html
None
Static.html
Dynamic.html
Selectable.html
Resizable.html
SwitchTo.html
Youtube.html
Vimeo.html
WYSIWYG.html
TinyMCE.html
CKEditor.html
SummerNote.html
CodeMirror.html
#
Charts.html
DynamicData.html
FileDownload.html
FileUpload.html
JqueryProgressBar.html
Loader.html
Modals.html
ProgressBar.html
http://practice.automationtesting.in/
#Default
#Serialize
#
https://www.facebook.com/automationtesting2016/
https://twitter.com/krishnasakinala
https://www.linkedin.com/nhome/?trk=hb_signin
https://plus.google.com/105286300926085335367
https://www.youtube.com/channel/UCmQRa3pWM9zsB474URz8ESg
Display the list of elements2 completed.
Test case 01 >> PASSED!
Execute

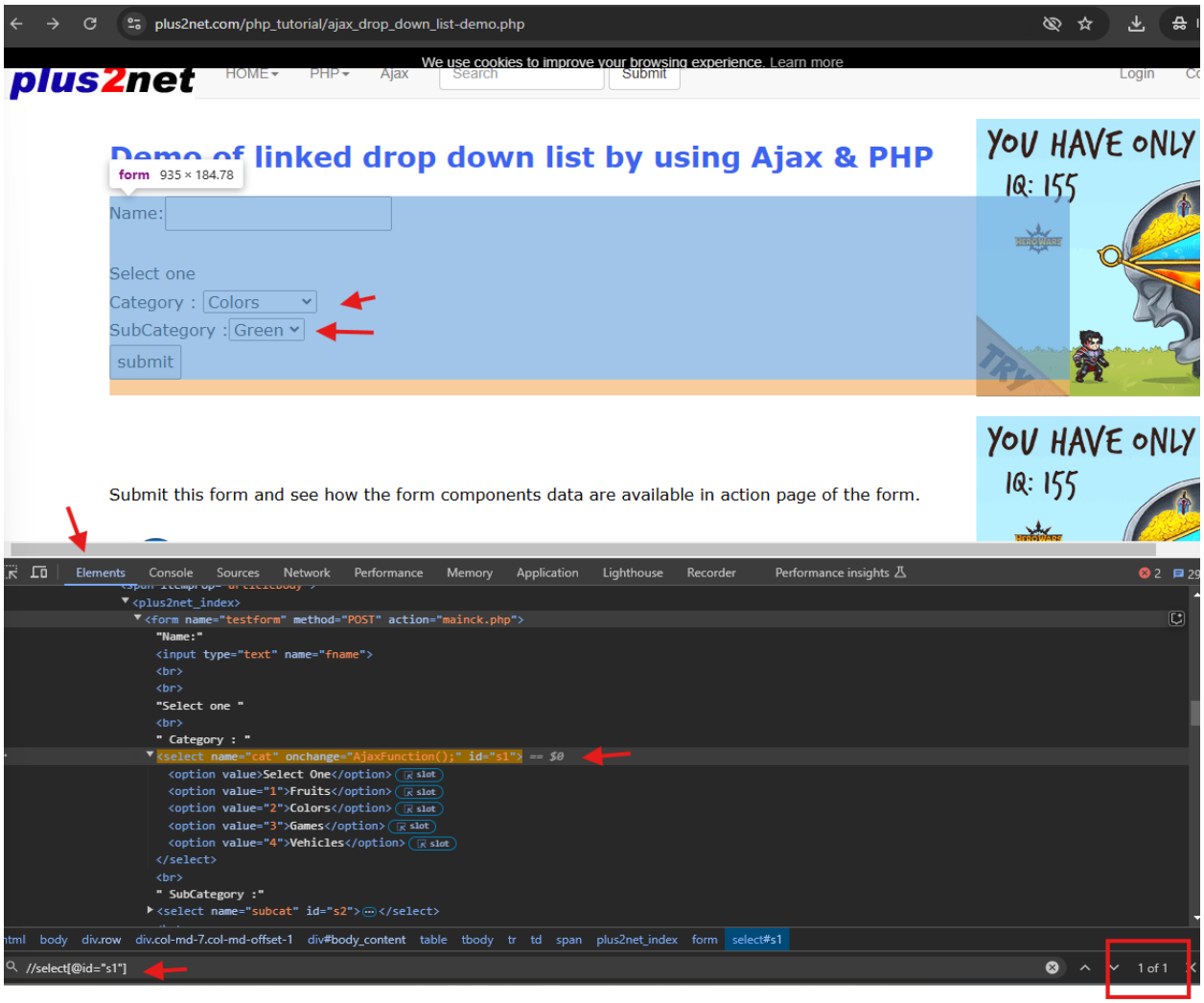
```python
    try:
        # >> Test case 02 - FAILED test case
        print('Test case 02 START!')
        # go to browser under test
        page.goto('https://demo.automationtesting.in/Selectable.html')
        # store multiple elements1 using List
        # incorrect page attribute below (to test the error handling syntax)
        page.query_selector('d//[sf="wrong"]').click()
        # correct page attribute below
        elements1 = page.query_selector_all('b')
        print('There are a total number of elements1 of >> ' + str(len(elements1)))
        print('Collecting of elements1 started...')
        for i in elements1:
            print(i.text_content())
        print('Display the list of elements1 completed.')
        page.wait_for_timeout(3000)
        # store multiple elements2 using List
        elements2 = page.query_selector_all('a')
        print('There are a total number of elements2 of >> ' + str(len(elements2)))
        print('Collecting of elements2 started...')
        for i in elements2:
            print(i.get_attribute('href'))
        print('Display the list of elements2 completed.')
        page.wait_for_timeout(3000)
        print('Test case 02 >> PASSED!')
    # error handling...
    except Exception as e:
        print(str(e))
        if str(e) != nullcontext:
            print('Test case 02 >> FAILED!')
    finally:
        print('Execute')
    # # close all other opened pages
    browser.close()
    print('All opened browser pages are now closed.')
```

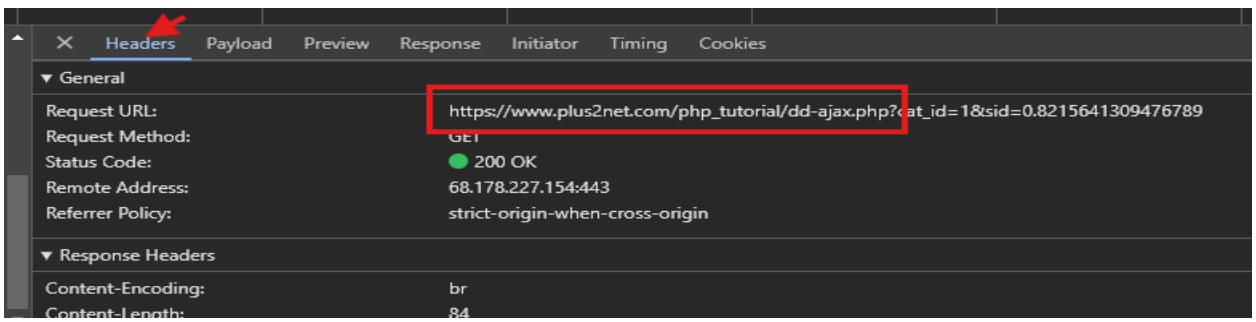
```
Test case 02 START!
Page.query_selector: Unexpected token "/" while parsing selector "d//[sf="wrong"]"
Test case 02 >> FAILED!
Execute
All opened browser pages are now closed.

Process finished with exit code 0
```

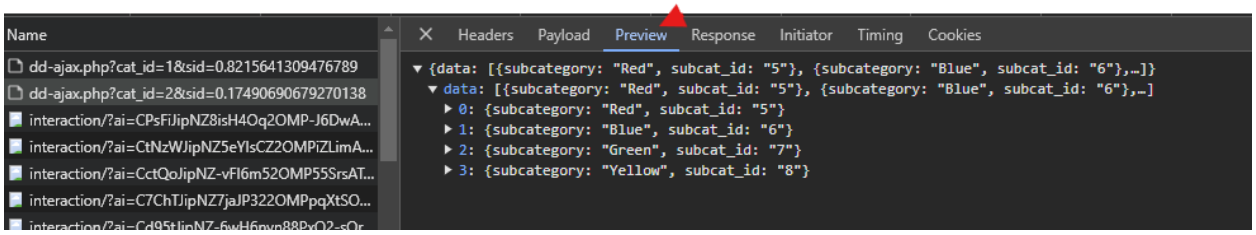
12. Working on web page Ajax handling

Test page url:
https://www.plus2net.com/php_tutorial/ajax_drop_down_list-demo.php

Locate (inspect) the element attributes for use as its locator:
Main dropdown…
Sub dropdown…







Syntax:

a. Test case 01 - AJAX handling

Using the XPath locator, create and run the code:

```
# == import and install files and libraries==
from playwright.sync_api import sync_playwright

# == methods==
def handle_ajax(response):  1 usage
    if 'https://www.plus2net.com/php_tutorial/dd-ajax.php?' in response.url:
        status = response.status
        data = response.text()
        print(f'status:{status}, data:{data}')

with sync_playwright() as p:
    browser = p.chromium.launch(headless=False)
    context = browser.new_context()
    page = context.new_page()

    # ======START OF TEST======

    # >> Test case 01 - AJAX handling
    print('Test case 01 START!')
    page.goto('https://www.plus2net.com/php_tutorial/ajax_drop_down_list-demo.php')
    print('browser page title is ' + '"' + page.title() +'"...')
    select = page.wait_for_selector('//select[@id="s1"]')
    # setup a "listener" to display the list values of selected category
    page.on( event: 'response', lambda response : handle_ajax(response))
    select.select_option('3')
    print('Dropdown value selected...')
    # other actions...
    print('Test case 01 >> PASSED!')
    page.wait_for_timeout(5000)
    browser.close()
```

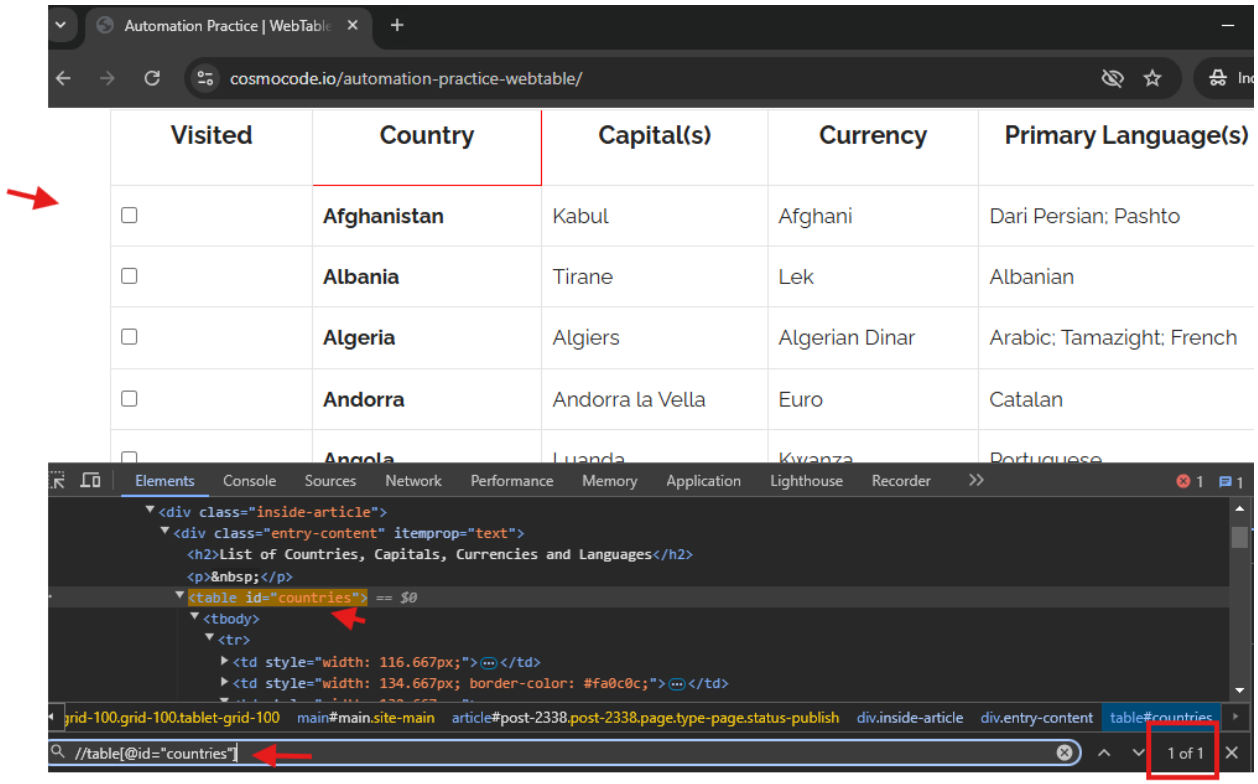
```
C:\Users\njmlo\PycharmProjects\Playwright\playwrightprj1\.venv\Scripts\python.exe C:\Users\njmlo\PycharmProjects\Playwr
Test case 01 START!
browser page title is "Demo of two Dependent drop down list using Ajax and PHP"...
Dropdown value selected...
Test case 01 >> PASSED!
status:200, data:{"data":[{"subcategory":"Cricket","subcat_id":"9"},{"subcategory":"Football","subcat_id":"10"},{"subca

Process finished with exit code 0
```

13. Working on web page Web Tables

Test page url:
https://cosmocode.io/automation-practice-webtable/

Locate (inspect) the element attributes for use as its locator:

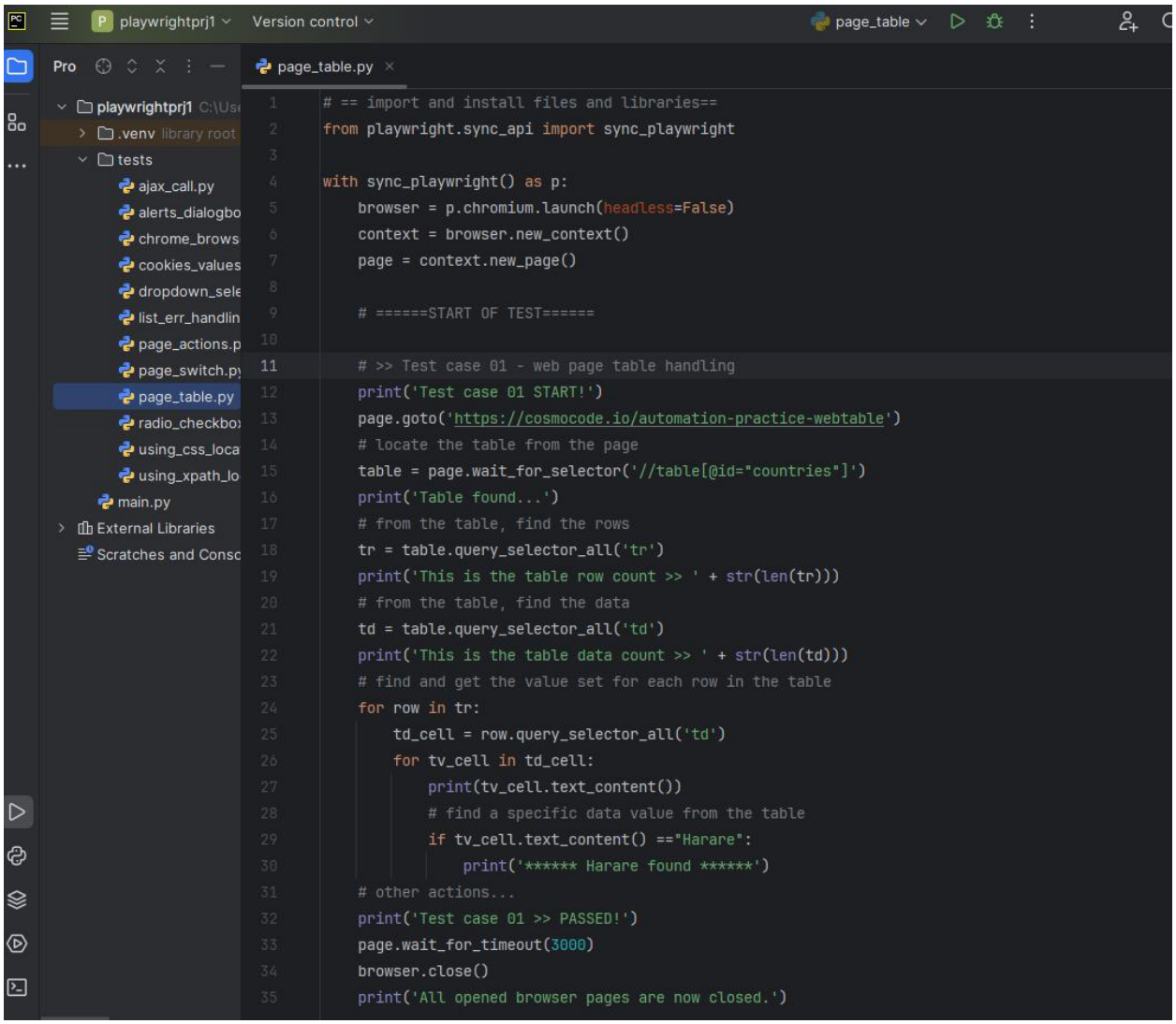
Demo table…

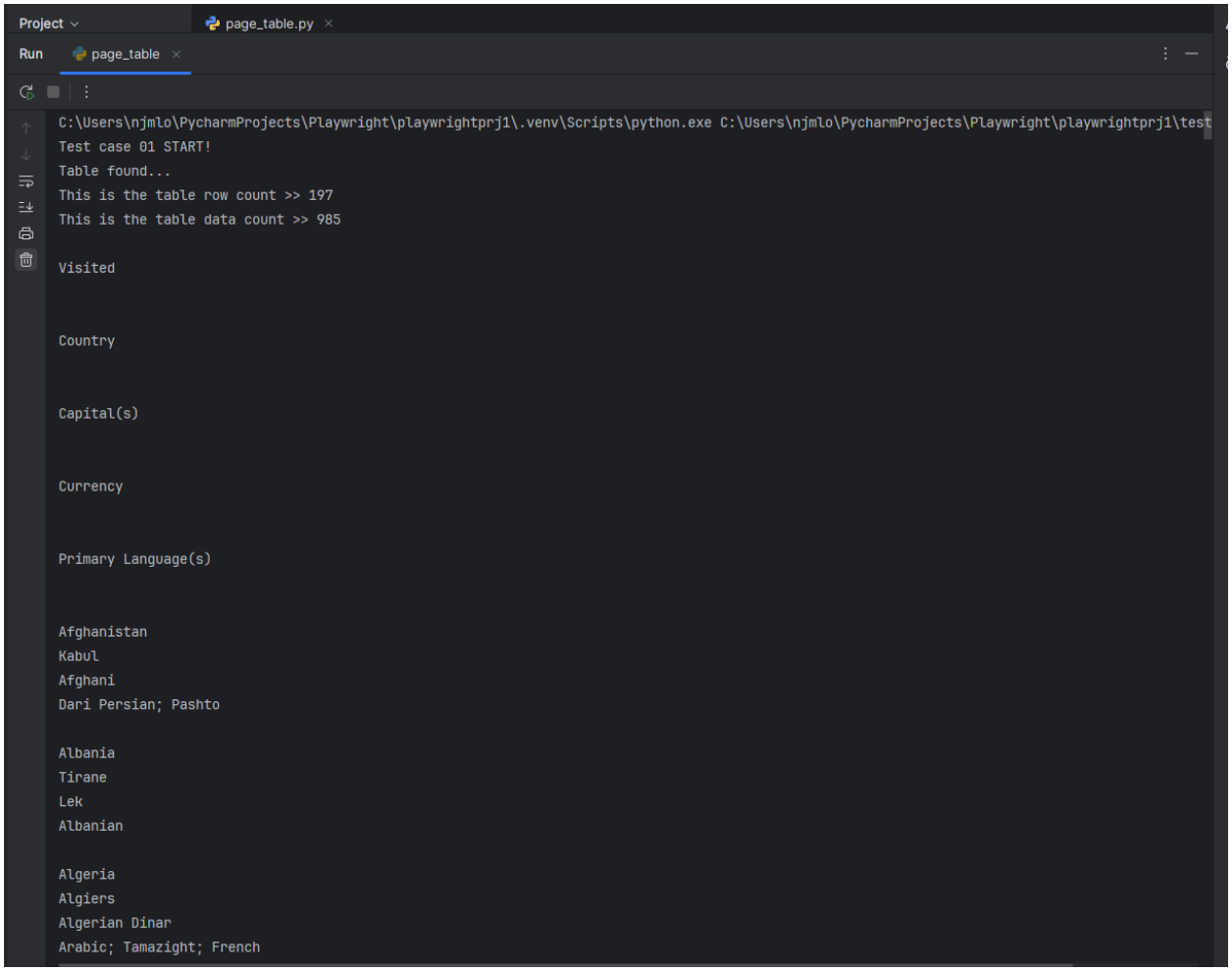


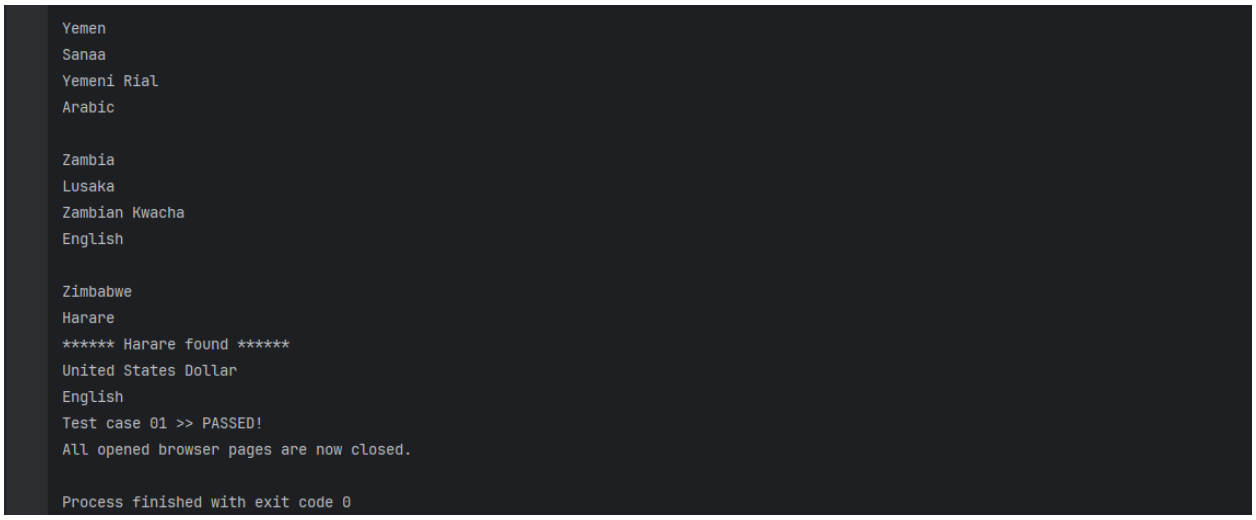
Syntax:

a. Test case 01 - web page table handling

Using the XPath locator, create and run the code:



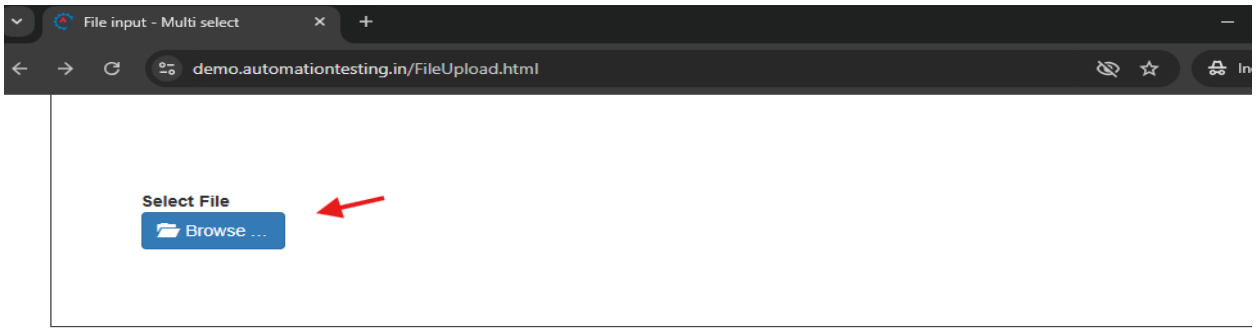

Project
page_table.py
Run
page_table
C:\Users\njmlo\PycharmProjects\Playwright\playwrightprj1\.venv\Scripts\python.exe C:\Users\njmlo\PycharmProjects\Playwright\playwrightprj1\test
Test case 01 START!
Table found...
This is the table row count >> 197
This is the table data count >> 985
Visited
Country
Capital(s)
Currency
Primary Language(s)
Afghanistan
Kabul
Afghani
Dari Persian; Pashto
Albania
Tirane
Lek
Albanian
Algeria
Algiers
Algerian Dinar
Arabic; Tamazight; French

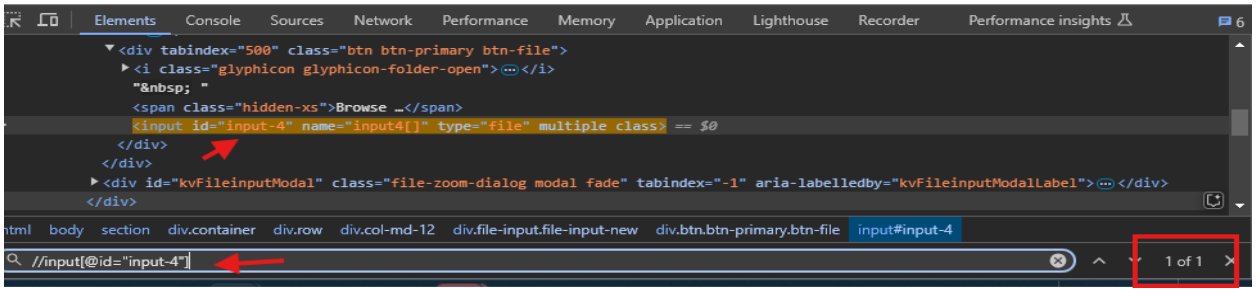

Yemen
Sanaa
Yemeni Rial
Arabic
Zambia
Lusaka
Zambian Kwacha
English
Zimbabwe
Harare
****** Harare found ******
United States Dollar
English
Test case 01 >> PASSED!
All opened browser pages are now closed.
Process finished with exit code 0

14. Working on web page upload and download of files

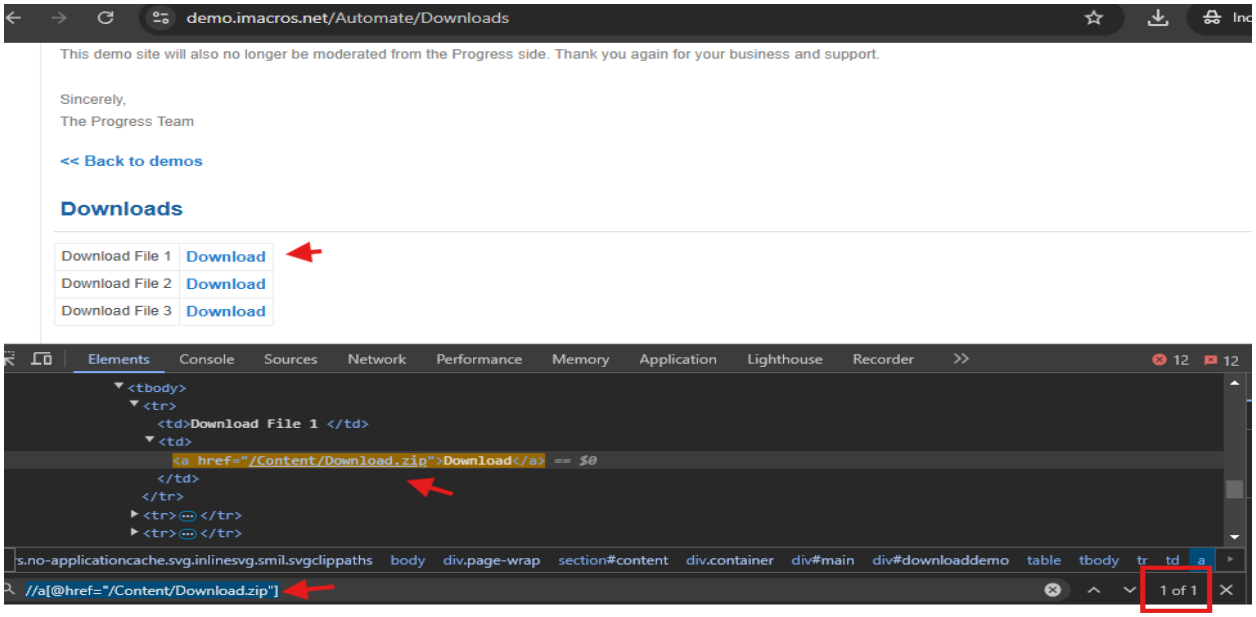
Test page url:
https://demo.automationtesting.in/FileUpload.html
https://demo.imacros.net/Automate/Downloads

Locate (inspect) the element attributes for use as its locator:
Upload page…





Download page…

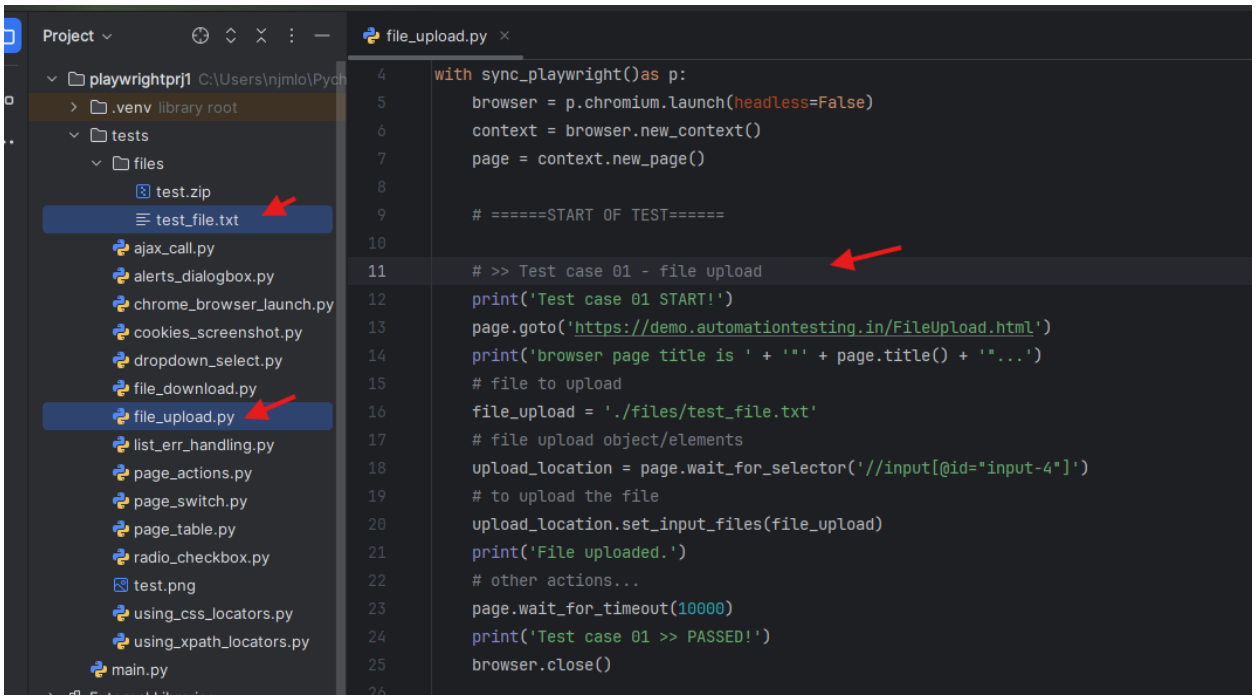


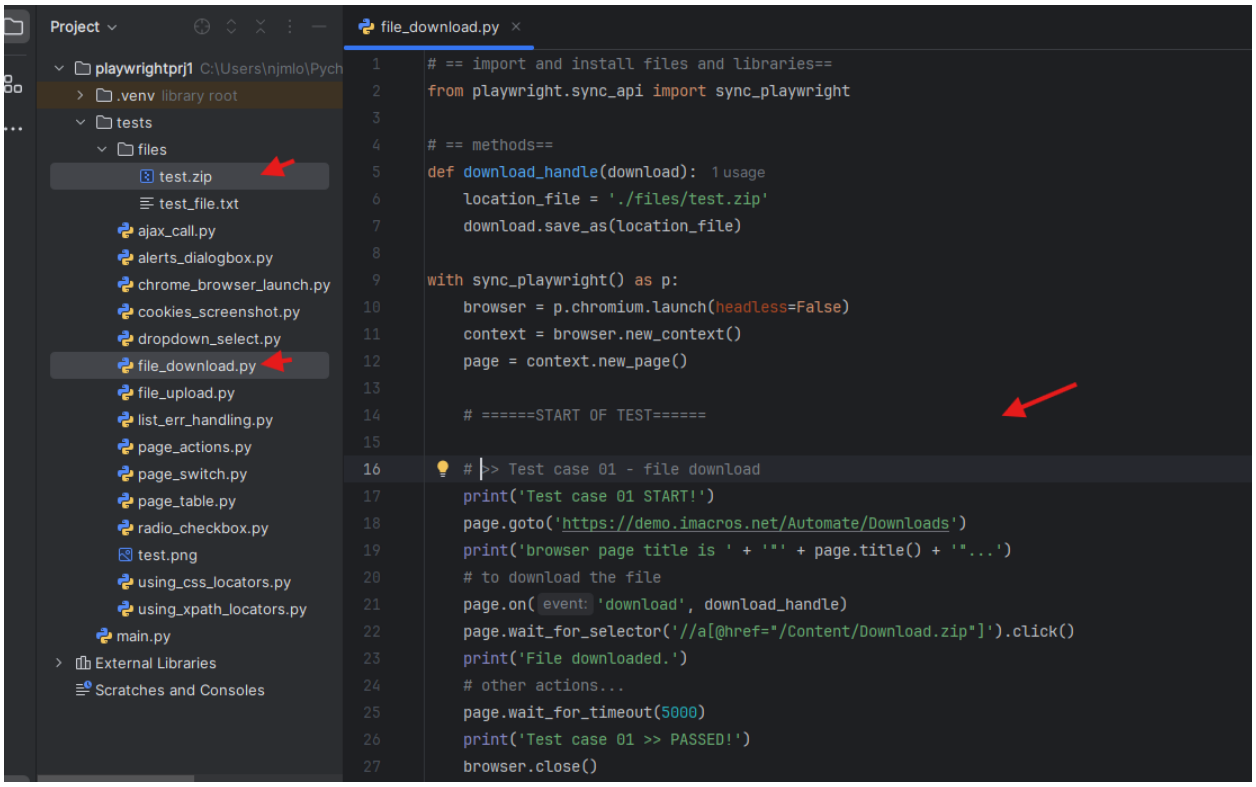
Syntax:

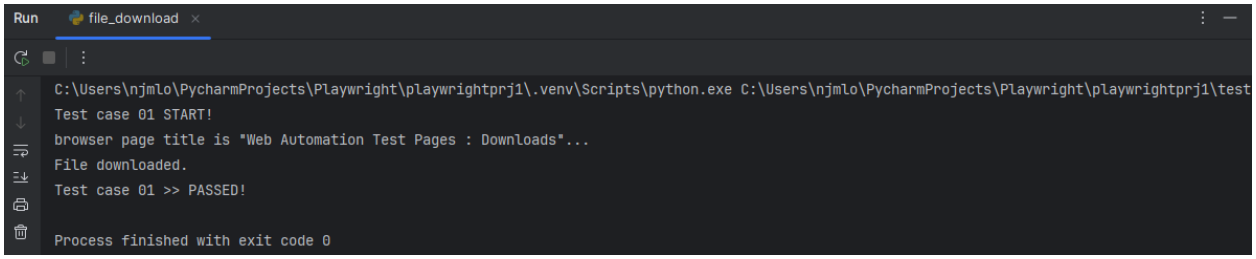
a. Test case 01 – file upload

b. Test case 01 – file download

Using the XPath locator, create and run the code:



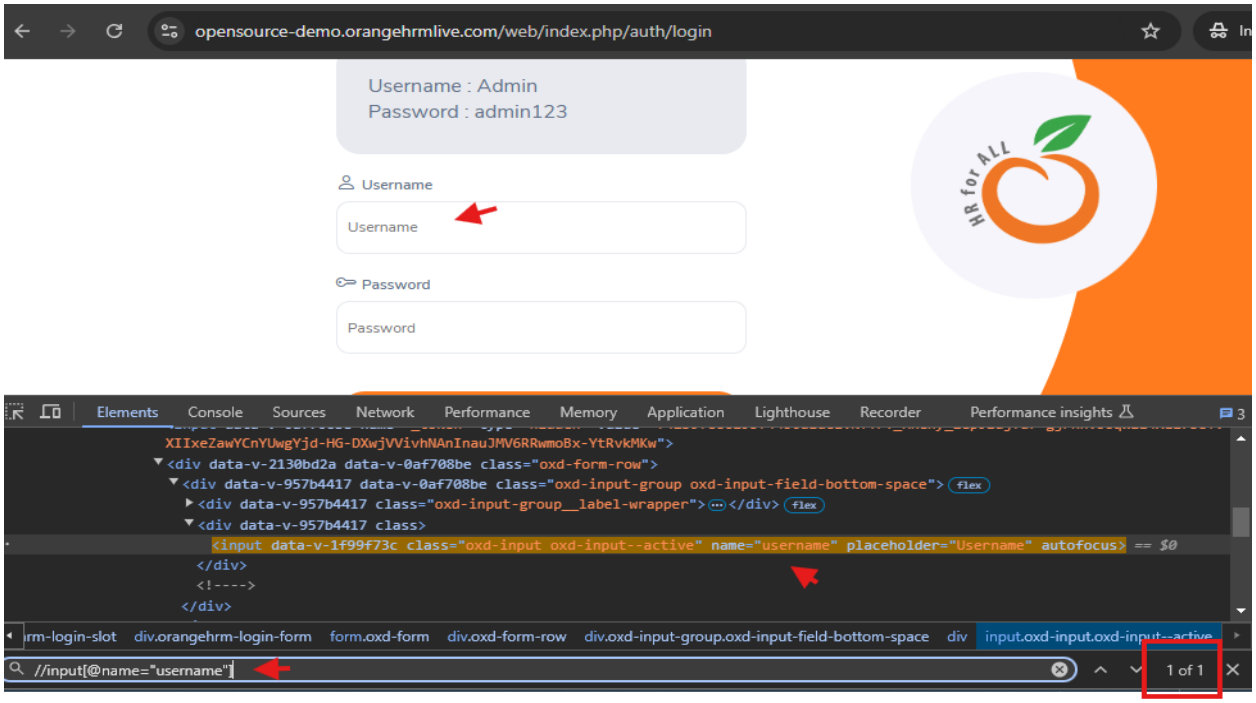

Project
playwrightprj1 C:\Users\njmlo\Pych
.venv library root
tests
files
test.zip
test_file.txt
ajax_call.py
alerts_dialogbox.py
chrome_browser_launch.py
cookies_screenshot.py
dropdown_select.py
file_download.py
file_upload.py
list_err_handling.py
page_actions.py
page_switch.py
page_table.py
radio_checkbox.py
test.png
using_css_locators.py
using_xpath_locators.py
main.py
External Libraries
Scratches and Consoles
file_download.py
# == import and install files and libraries==
from playwright.sync_api import sync_playwright
# == methods==
def download_handle(download): 1 usage
location_file = './files/test.zip'
download.save_as(location_file)
with sync_playwright() as p:
browser = p.chromium.launch(headless=False)
context = browser.new_context()
page = context.new_page()
# ======START OF TEST======
# >> Test case 01 - file download
print('Test case 01 START!')
page.goto('https://demo.imacros.net/Automate/Downloads')
print('browser page title is ' + '"' + page.title() + '"...')
# to download the file
page.on(event: 'download', download_handle)
page.wait_for_selector('//a[@href="/Content/Download.zip"]').click()
print('File downloaded.')
# other actions...
page.wait_for_timeout(5000)
print('Test case 01 >> PASSED!')
browser.close()

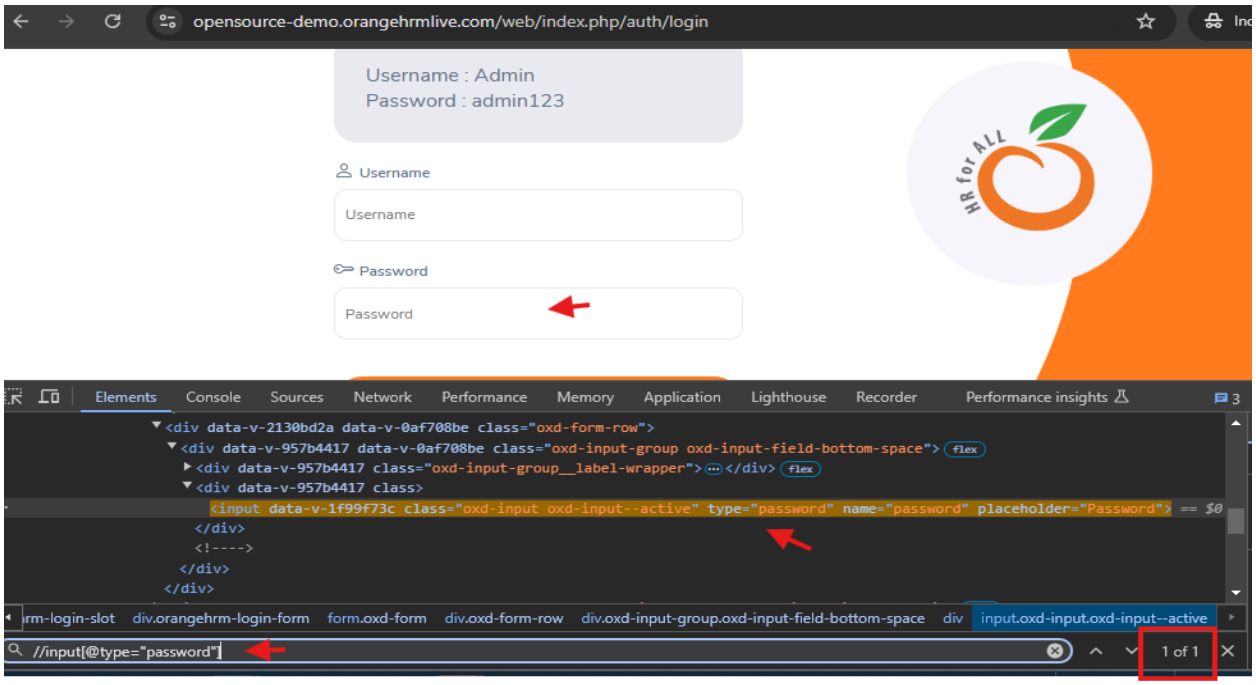

Run
file_download
C:\Users\njmlo\PycharmProjects\Playwright\playwrightprj1\.venv\Scripts\python.exe C:\Users\njmlo\PycharmProjects\Playwright\playwrightprj1\test
Test case 01 START!
browser page title is "Web Automation Test Pages : Downloads"...
File downloaded.
Test case 01 >> PASSED!
Process finished with exit code 0

15. Working on web page desktop video recording and screenshots

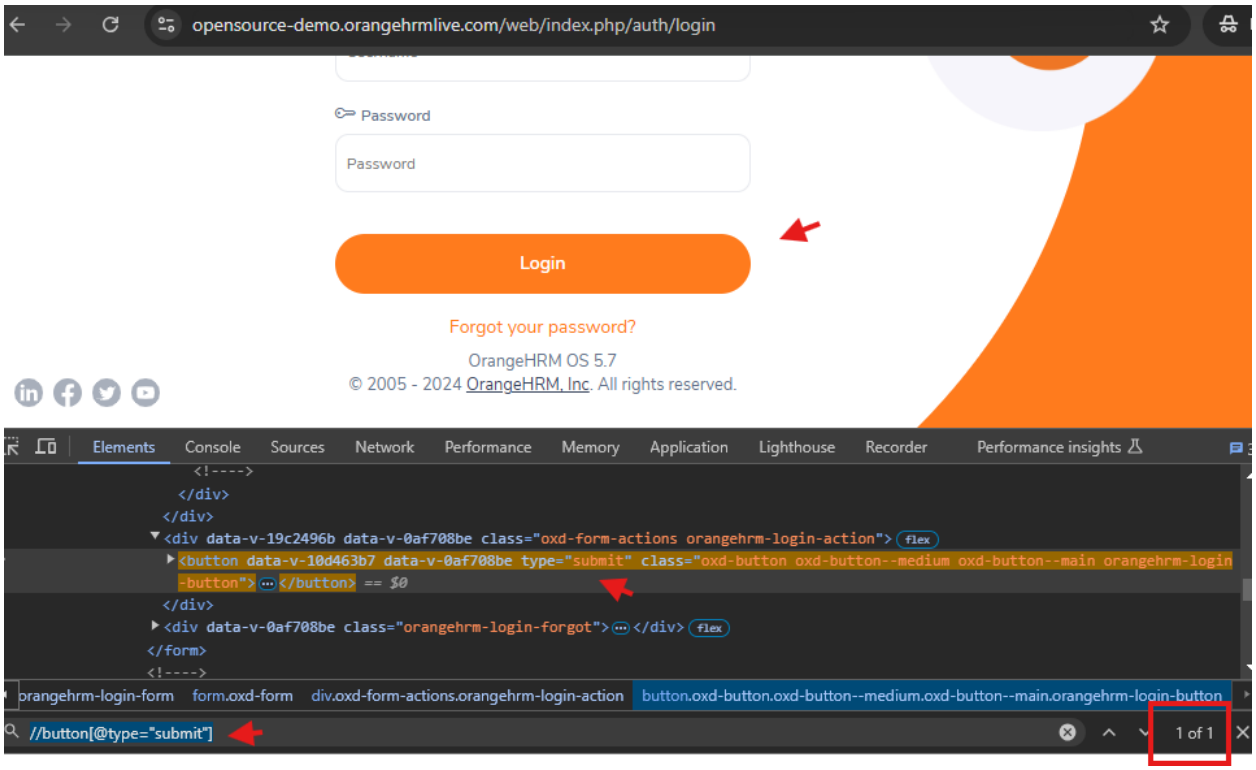
Test page url:
https://opensource-demo.orangehrmlive.com/web/index.php/auth/login

Locate (inspect) the element attributes for use as its locator:
Username and Password textboxes…





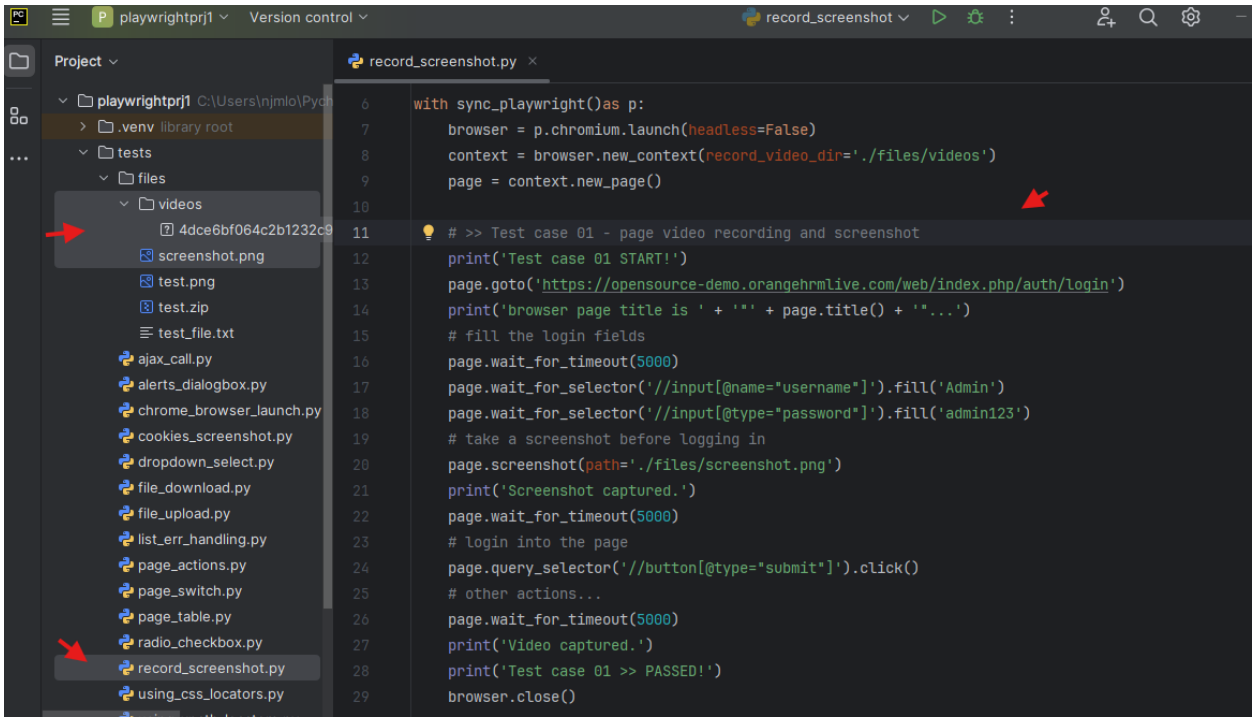
Submit button…

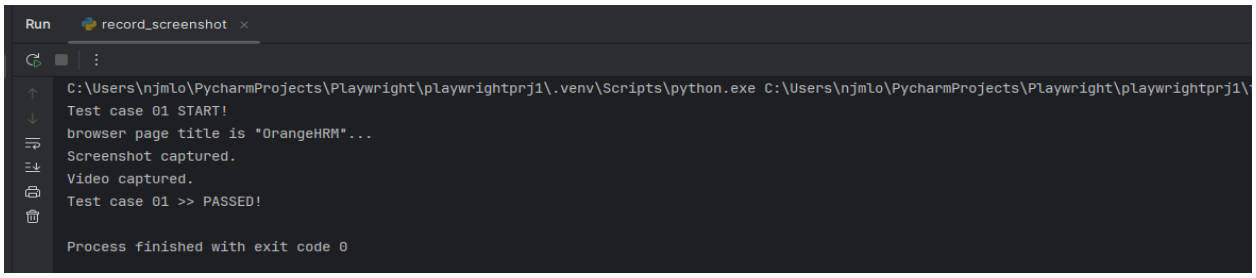


Syntax:

a. Test case 01 - page video recording and screenshot

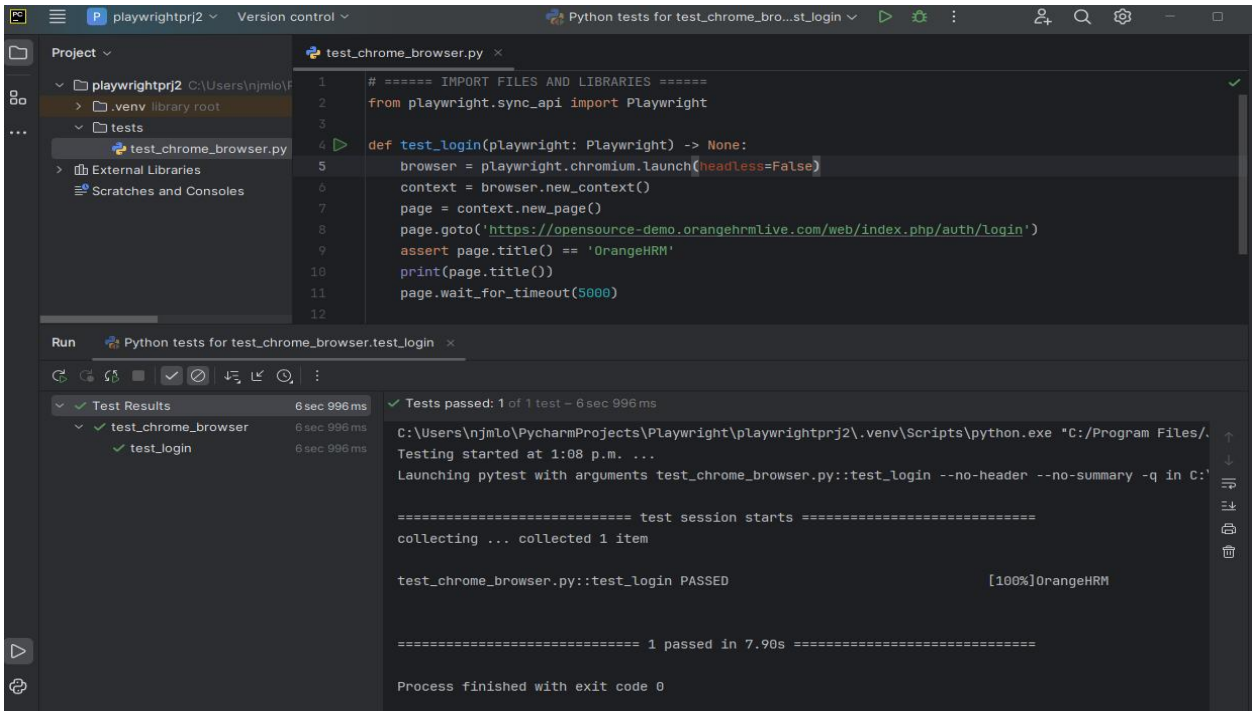
Using the XPath locator, create and run the code:

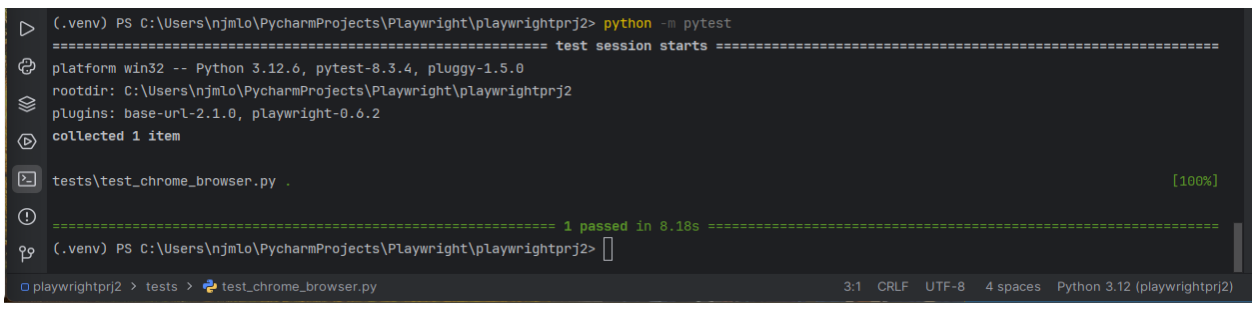




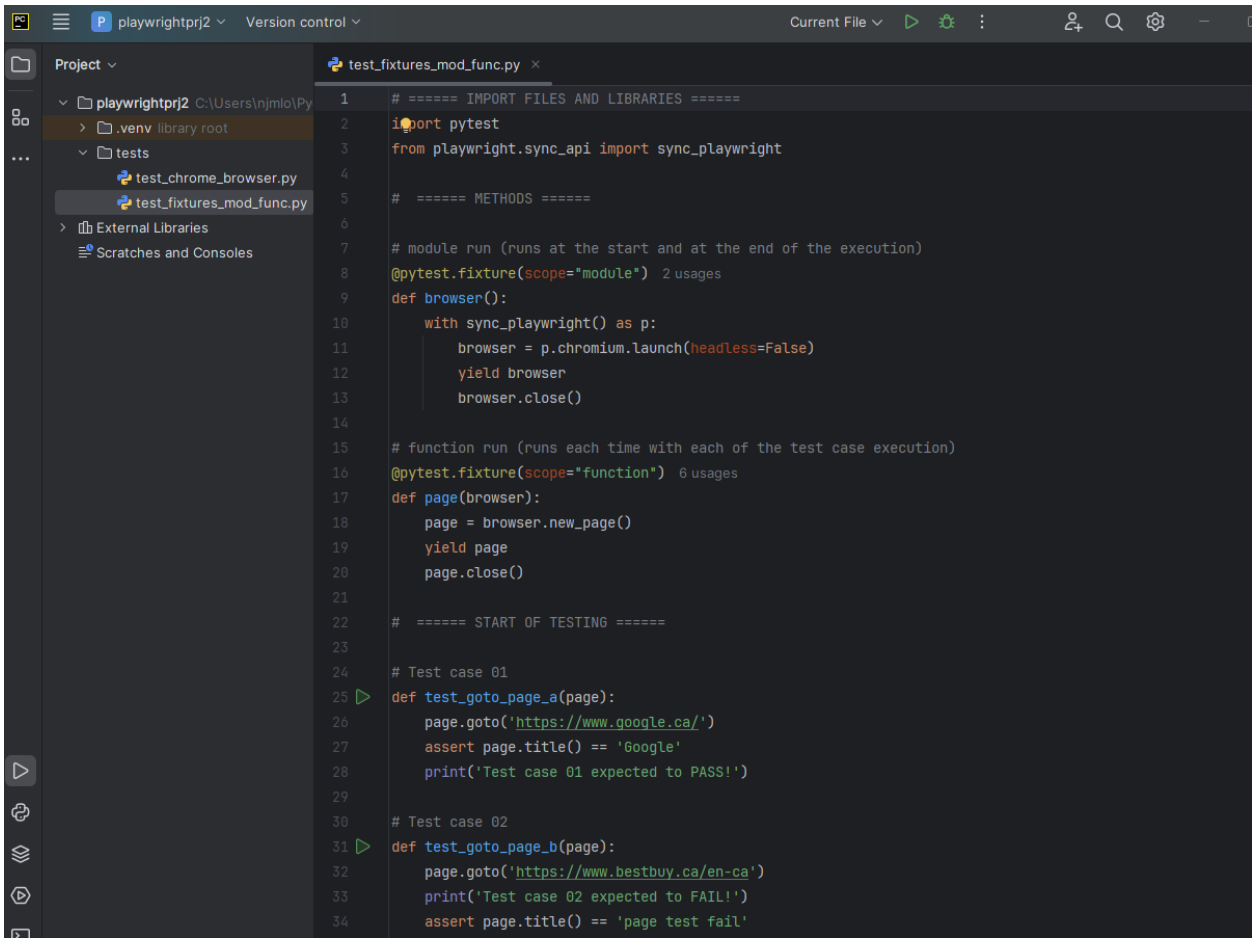
16. Working on web page with Playwright and PyTest

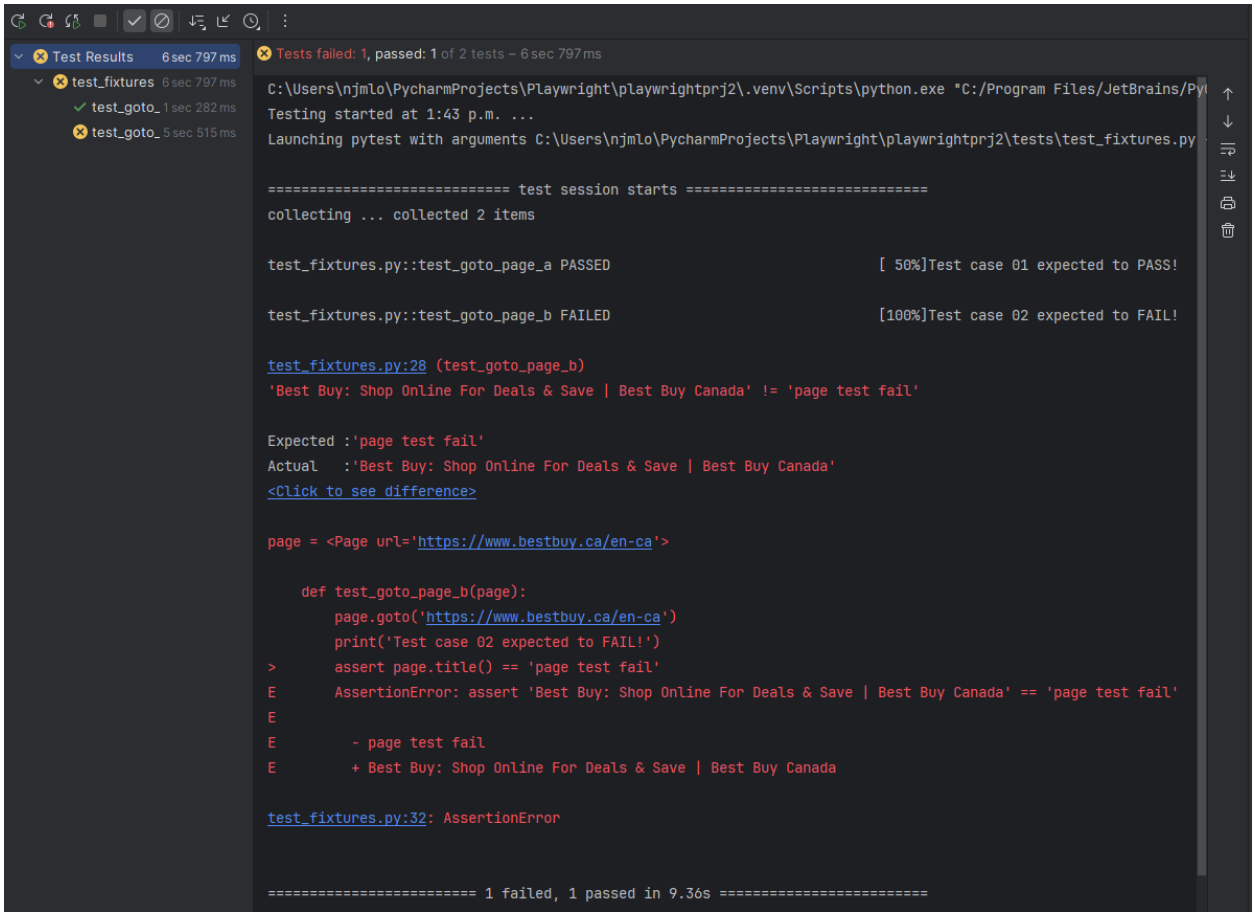
Check PyTest if installed and its version:



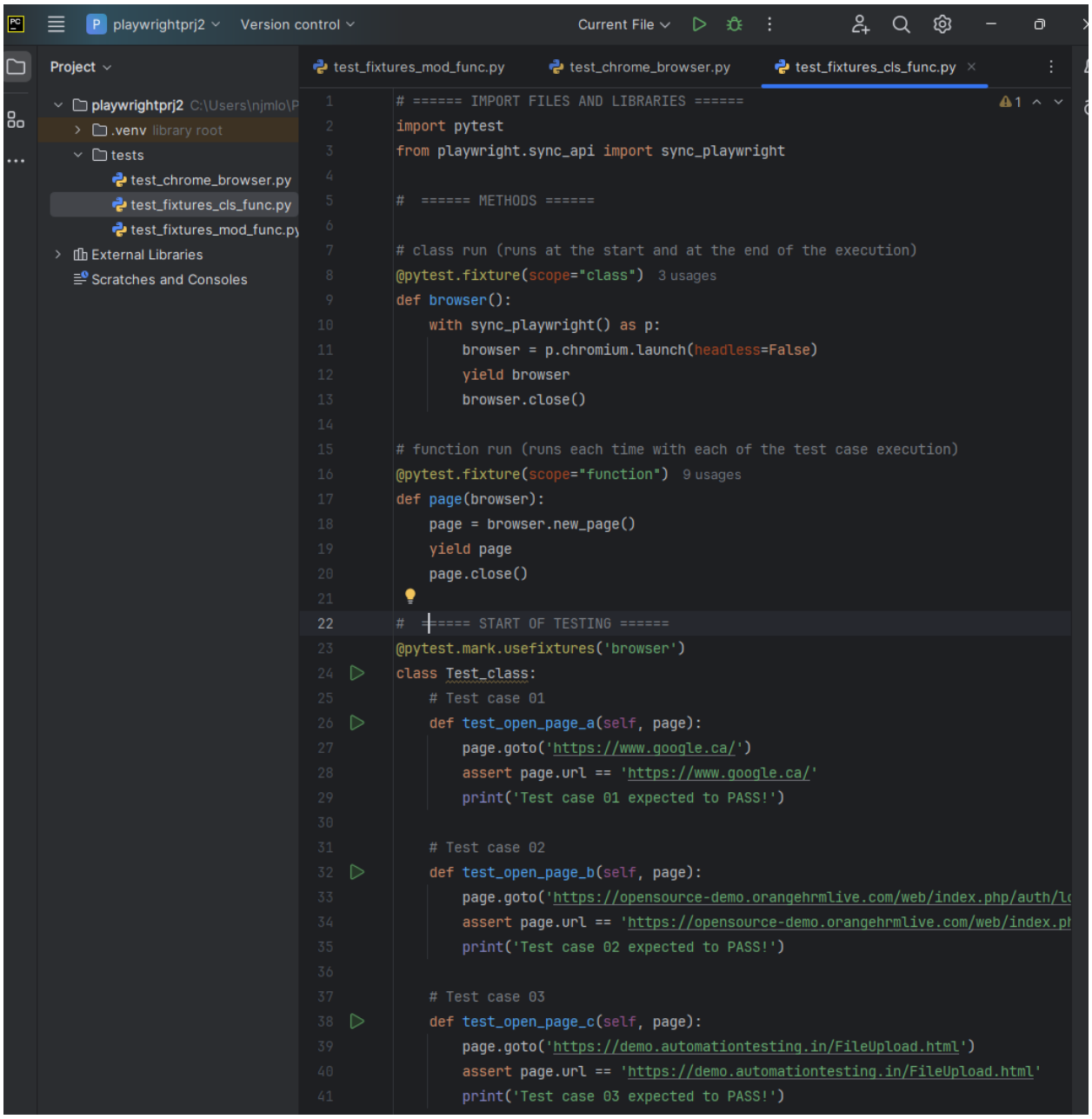


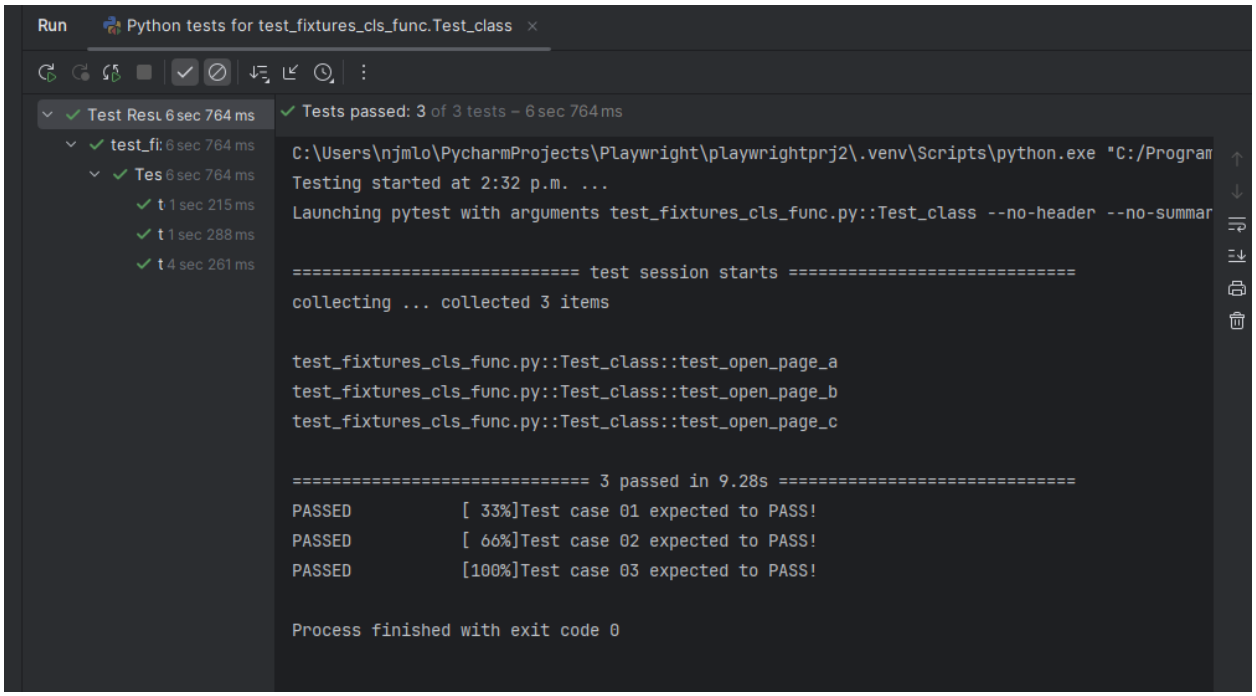
17. Working on web page with Fixtures (module and functions)

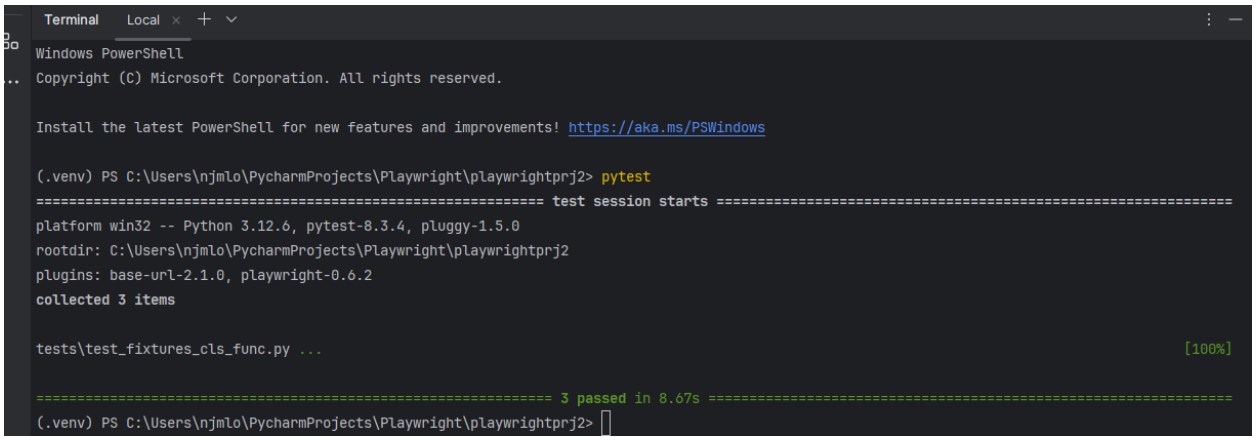



Test Results 6 sec 797 ms
test_fixtures 6 sec 797 ms
test_goto_ 1 sec 282 ms
test_goto_ 5 sec 515 ms
Tests failed: 1, passed: 1 of 2 tests – 6 sec 797 ms
C:\Users\njmlo\PycharmProjects\Playwright\playwrightprj2\.venv\Scripts\python.exe "C:/Program Files/JetBrains/Py
Testing started at 1:43 p.m. ...
Launching pytest with arguments C:\Users\njmlo\PycharmProjects\Playwright\playwrightprj2\tests\test_fixtures.py
============================= test session starts =============================
collecting ... collected 2 items
test_fixtures.py::test_goto_page_a PASSED [50%]Test case 01 expected to PASS!
test_fixtures.py::test_goto_page_b FAILED [100%]Test case 02 expected to FAIL!
test_fixtures.py:28 (test_goto_page_b)
'Best Buy: Shop Online For Deals & Save | Best Buy Canada' != 'page test fail'
Expected :'page test fail'
Actual :'Best Buy: Shop Online For Deals & Save | Best Buy Canada'
<Click to see difference>
page = <Page url='https://www.bestbuy.ca/en-ca'>
def test_goto_page_b(page):
page.goto('https://www.bestbuy.ca/en-ca')
print('Test case 02 expected to FAIL!')
> assert page.title() == 'page test fail'
E AssertionError: assert 'Best Buy: Shop Online For Deals & Save | Best Buy Canada' == 'page test fail'
E
E - page test fail
E + Best Buy: Shop Online For Deals & Save | Best Buy Canada
test_fixtures.py:32: AssertionError
========================= 1 failed, 1 passed in 9.36s =========================

18. Working on web page with Fixtures (class and functions)



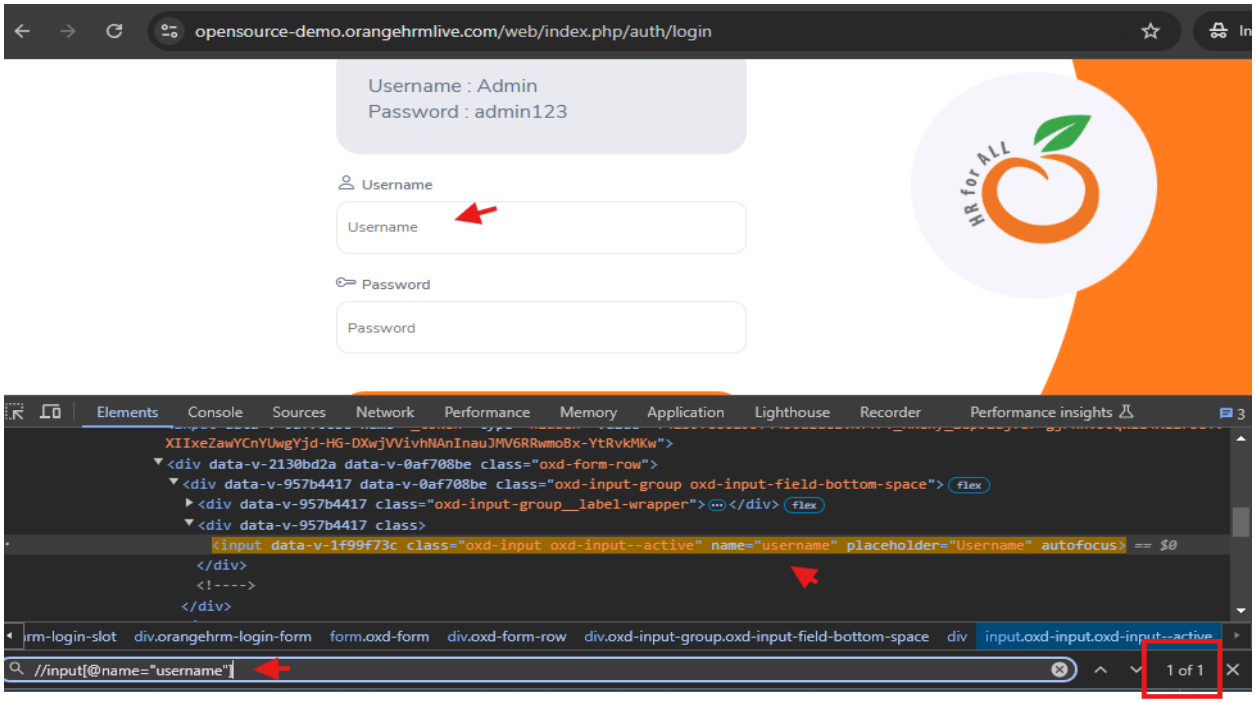

Run
Python tests for test_fixtures_cls_func.Test_class
Test Resu 6 sec 764 ms
test_fi: 6 sec 764 ms
Tes 6 sec 764 ms
t 1 sec 215 ms
t 1 sec 288 ms
t 4 sec 261 ms
Tests passed: 3 of 3 tests – 6 sec 764 ms
C:\Users\njmlo\PycharmProjects\Playwright\playwrightprj2\.venv\Scripts\python.exe "C:/Program
Testing started at 2:32 p.m. ...
Launching pytest with arguments test_fixtures_cls_func.py::Test_class --no-header --no-summar
============================= test session starts =============================
collecting ... collected 3 items
test_fixtures_cls_func.py::Test_class::test_open_page_a
test_fixtures_cls_func.py::Test_class::test_open_page_b
test_fixtures_cls_func.py::Test_class::test_open_page_c
============================== 3 passed in 9.28s ==============================
PASSED [33%]Test case 01 expected to PASS!
PASSED [66%]Test case 02 expected to PASS!
PASSED [100%]Test case 03 expected to PASS!
Process finished with exit code 0

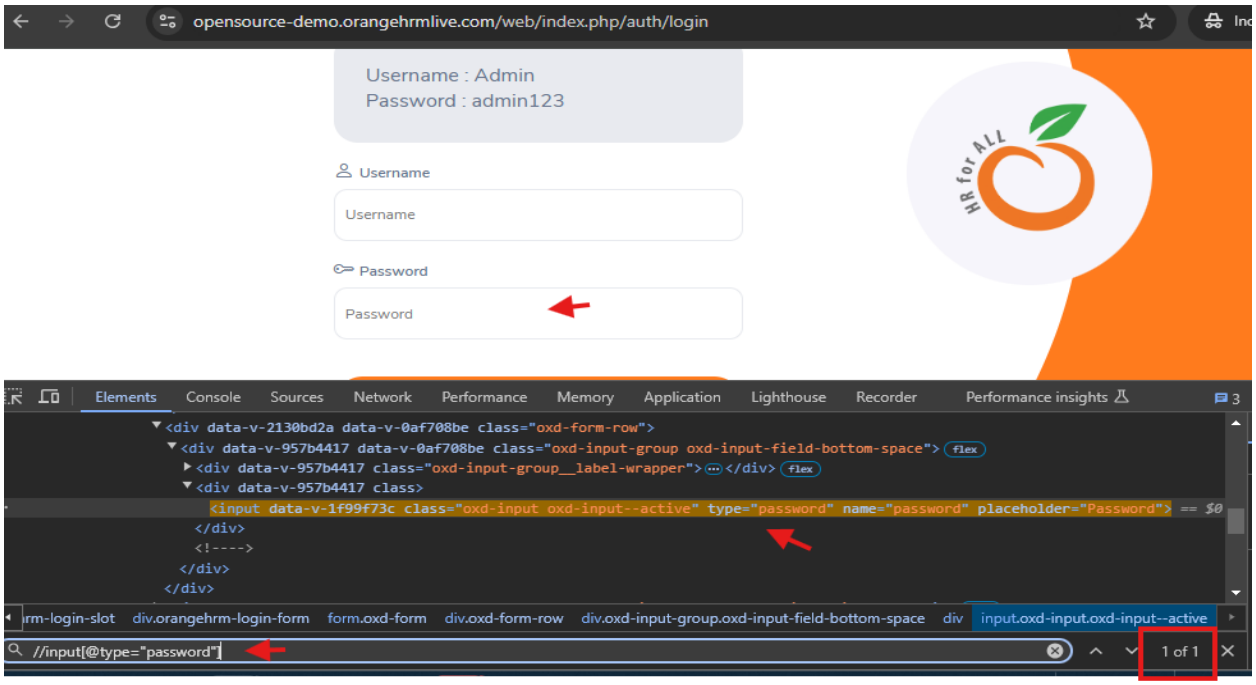

Terminal
Local
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.
Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows
(.venv) PS C:\Users\njmlo\PycharmProjects\Playwright\playwrightprj2> pytest
============================= test session starts =============================
platform win32 -- Python 3.12.6, pytest-8.3.4, pluggy-1.5.0
rootdir: C:\Users\njmlo\PycharmProjects\Playwright\playwrightprj2
plugins: base-url-2.1.0, playwright-0.6.2
collected 3 items
tests\test_fixtures_cls_func.py ... [100%]
============================== 3 passed in 8.67s ==============================
(.venv) PS C:\Users\njmlo\PycharmProjects\Playwright\playwrightprj2>

19. Working on web page parameterizing tests in PyTest with Playwright

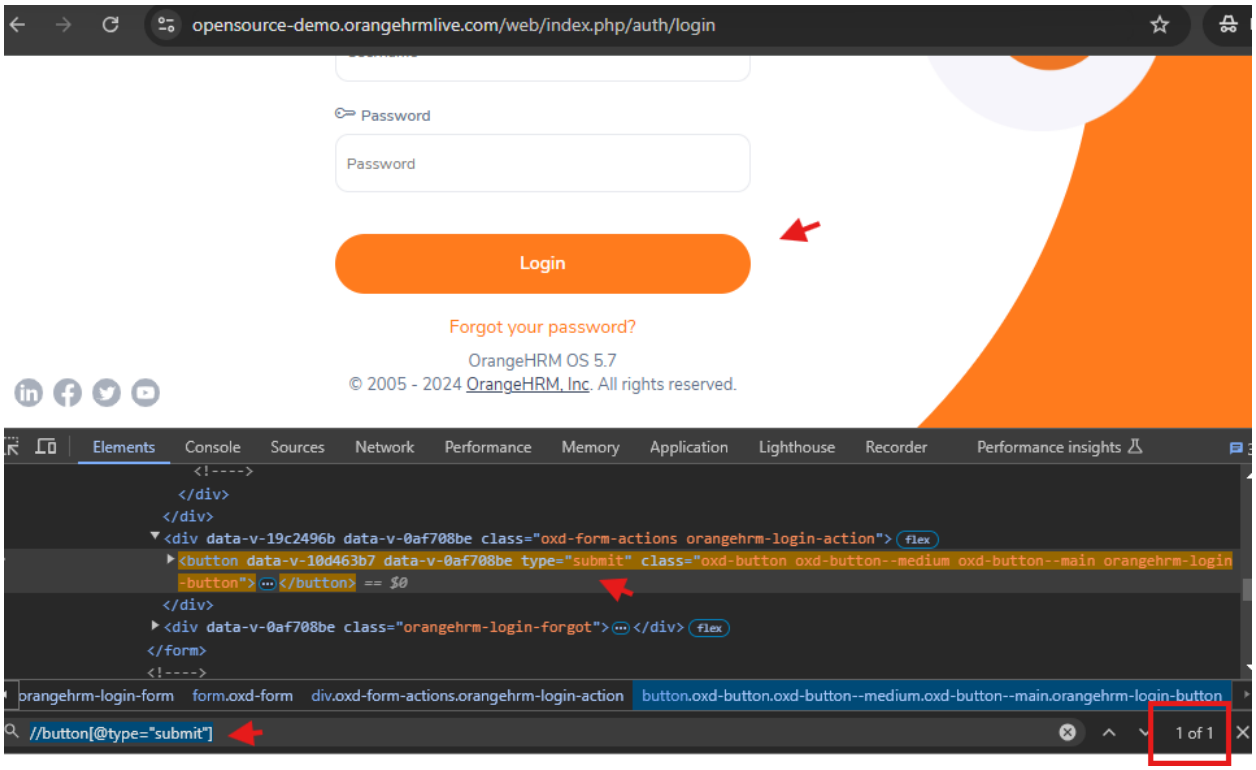
Test page url:
https://opensource-demo.orangehrmlive.com/web/index.php/auth/login

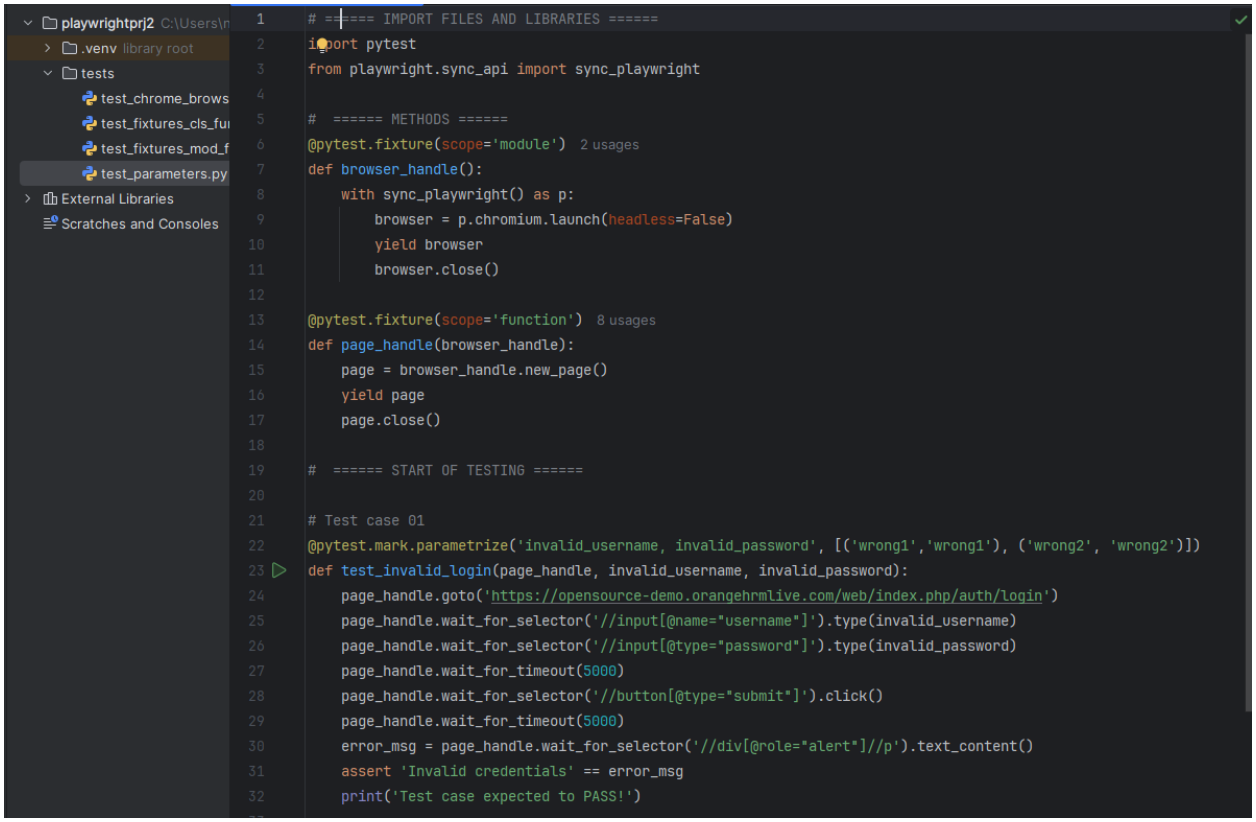
Locate (inspect) the element attributes for use as its locator:
Username and Password textboxes…

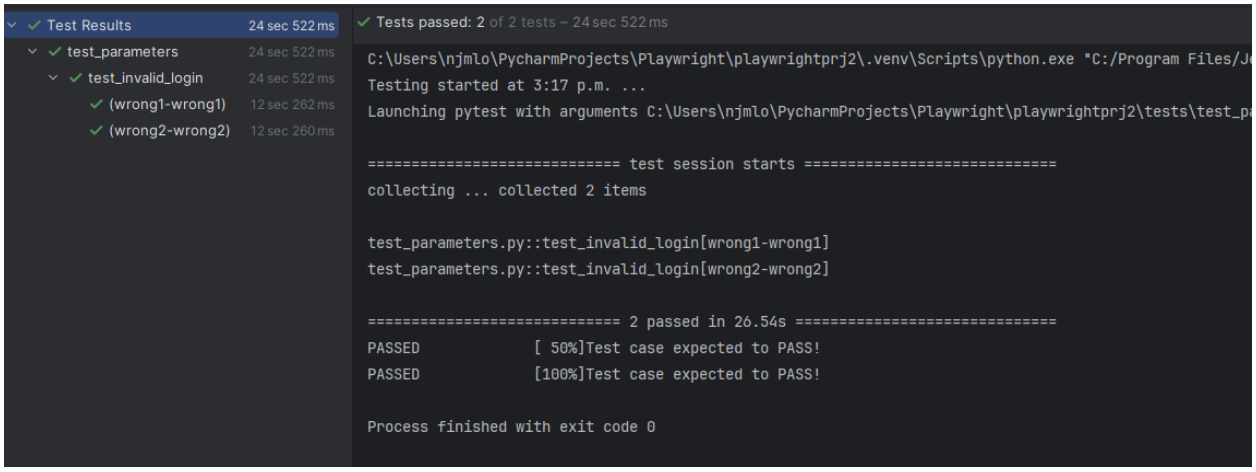




Submit button…







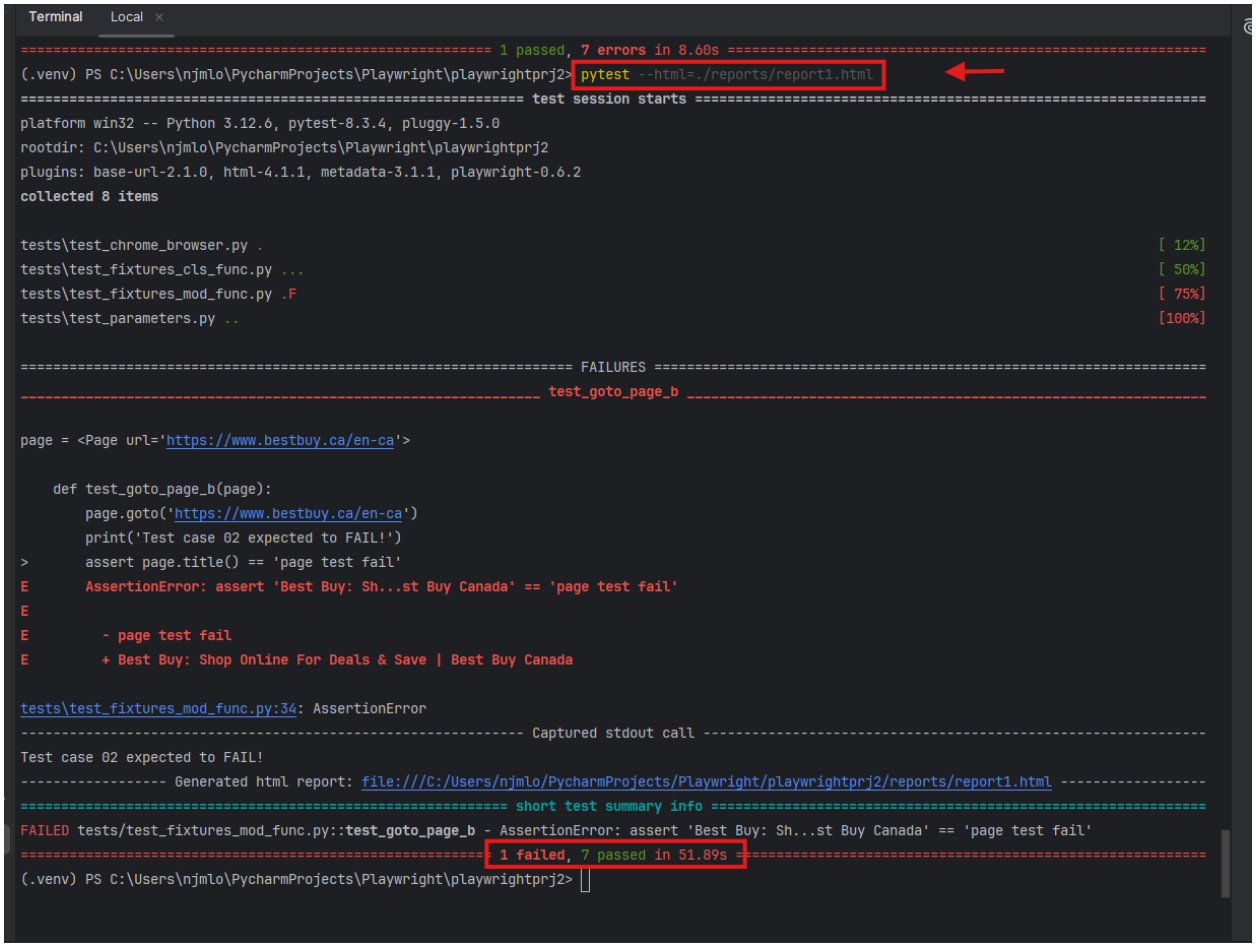
20. Working on web page test reporting in PyTest with Playwright (PyTest HTML plugin)

Install the PyTest-HTML

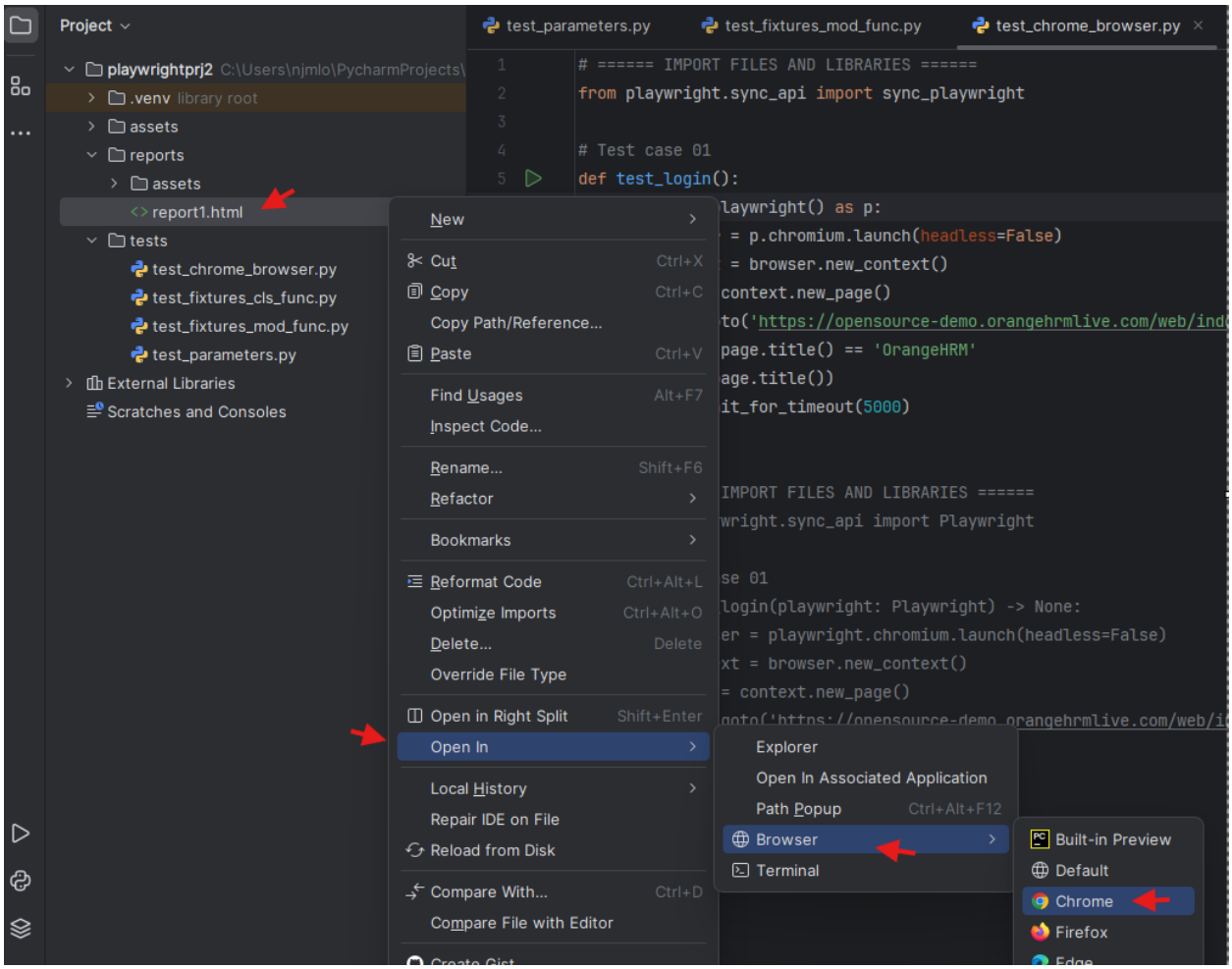
```
(.venv) PS C:\Users\njmlo\PycharmProjects\Playwright\playwrightprj2> pip install pytest-html
Collecting pytest-html
  Using cached pytest_html-4.1.1-py3-none-any.whl.metadata (3.9 kB)
Collecting jinja2>=3.0.0 (from pytest-html)
  Using cached jinja2-3.1.4-py3-none-any.whl.metadata (2.6 kB)
Collecting pytest-metadata>=2.0.0 (from pytest-html)
  Using cached pytest_metadata-3.1.1-py3-none-any.whl.metadata (8.6 kB)
Requirement already satisfied: pytest>=7.0.0 in c:\users\njmlo\pycharmprojects\playwright\playwrightprj2\.venv\lib\site-packages (from pytest-html)
 (8.3.4)
Collecting MarkupSafe>=2.0 (from jinja2>=3.0.0->pytest-html)
  Downloading MarkupSafe-3.0.2-cp312-cp312-win_amd64.whl.metadata (4.1 kB)
Requirement already satisfied: colorama in c:\users\njmlo\pycharmprojects\playwright\playwrightprj2\.venv\lib\site-packages (from pytest>=7.0.0->py
test-html) (0.4.6)
Requirement already satisfied: iniconfig in c:\users\njmlo\pycharmprojects\playwright\playwrightprj2\.venv\lib\site-packages (from pytest>=7.0.0->p
ytest-html) (2.0.0)
Requirement already satisfied: packaging in c:\users\njmlo\pycharmprojects\playwright\playwrightprj2\.venv\lib\site-packages (from pytest>=7.0.0->p
ytest-html) (24.2)
Requirement already satisfied: pluggy<2,>=1.5 in c:\users\njmlo\pycharmprojects\playwright\playwrightprj2\.venv\lib\site-packages (from pytest>=7.0
.0->pytest-html) (1.5.0)
Using cached pytest_html-4.1.1-py3-none-any.whl (23 kB)
Using cached jinja2-3.1.4-py3-none-any.whl (133 kB)
Using cached pytest_metadata-3.1.1-py3-none-any.whl (11 kB)
Downloading MarkupSafe-3.0.2-cp312-cp312-win_amd64.whl (15 kB)
Installing collected packages: MarkupSafe, pytest-metadata, jinja2, pytest-html
Successfully installed MarkupSafe-3.0.2 jinja2-3.1.4 pytest-html-4.1.1 pytest-metadata-3.1.1
```

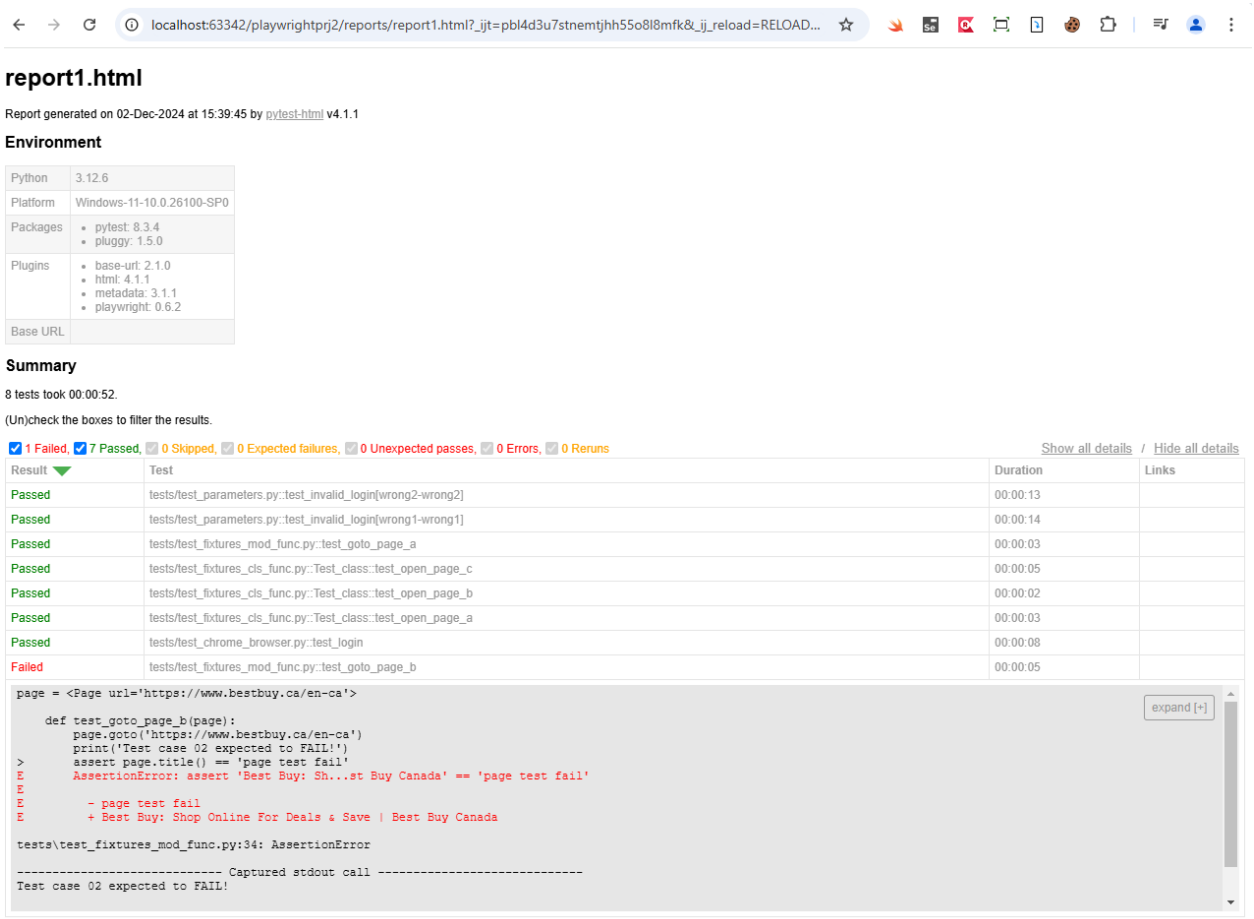
Cmd line syntax:

pytest --html=./reports/report1.html



Look for the generated report





localhost:63342/playwrightprj2/reports/report1.html?_ijt=pbl4d3u7stnemtjhh55o8l8mfk&_ij_reload=RELOAD...

report1.html

Report generated on 02-Dec-2024 at 15:39:45 by pytest-html v4.1.1

Environment

Python	3.12.6
Platform	Windows-11-10.0.26100-SP0
Packages	• pytest: 8.3.4 • pluggy: 1.5.0
Plugins	• base-url: 2.1.0 • html: 4.1.1 • metadata: 3.1.1 • playwright: 0.6.2
Base URL	

Summary

8 tests took 00:00:52.

(Un)check the boxes to filter the results.

1 Failed, 7 Passed, 0 Skipped, 0 Expected failures, 0 Unexpected passes, 0 Errors, 0 Reruns

Show all details / Hide all details

Result	Test	Duration	Links
Passed	tests/test_parameters.py::test_invalid_login[wrong2-wrong2]	00:00:13	
Passed	tests/test_parameters.py::test_invalid_login[wrong1-wrong1]	00:00:14	
Passed	tests/test_fixtures_mod_func.py::test_goto_page_a	00:00:03	
Passed	tests/test_fixtures_cls_func.py::Test_class::test_open_page_c	00:00:05	
Passed	tests/test_fixtures_cls_func.py::Test_class::test_open_page_b	00:00:02	
Passed	tests/test_fixtures_cls_func.py::Test_class::test_open_page_a	00:00:03	
Passed	tests/test_chrome_browser.py::test_login	00:00:08	
Failed	tests/test_fixtures_mod_func.py::test_goto_page_b	00:00:05	

expand [+]

```
page = <Page url='https://www.bestbuy.ca/en-ca'>

    def test_goto_page_b(page):
        page.goto('https://www.bestbuy.ca/en-ca')
        print('Test case 02 expected to FAIL!')
>       assert page.title() == 'page test fail'
E       AssertionError: assert 'Best Buy: Sh...st Buy Canada' == 'page test fail'
E
E         - page test fail
E         + Best Buy: Shop Online For Deals & Save | Best Buy Canada

tests\test_fixtures_mod_func.py:34: AssertionError

----------------------------- Captured stdout call -----------------------------
Test case 02 expected to FAIL!
```

**********END**********