

Test Automation using:

- Python
- Selenium IDE and Webdriver
- PyCharm
- PyTest
- Libraries (ie., time)
- Github repository

Document highlight:

- Tools installation and setup
- Run a post installation test
- Interacting with elements in a web page
- Record, Playback, and Export (Selenium IDE)
- Pytest with Selenium and Python (generate html reports)
- More Selenium, Python, Pytest sampling codes and test run
 - Pytest fixtures, set up, and tear down
 - Parameterization
 - Run a test case using the command line/terminal
- Page Object Model (POM) sampling codes and test run

1. Tools installation and setup

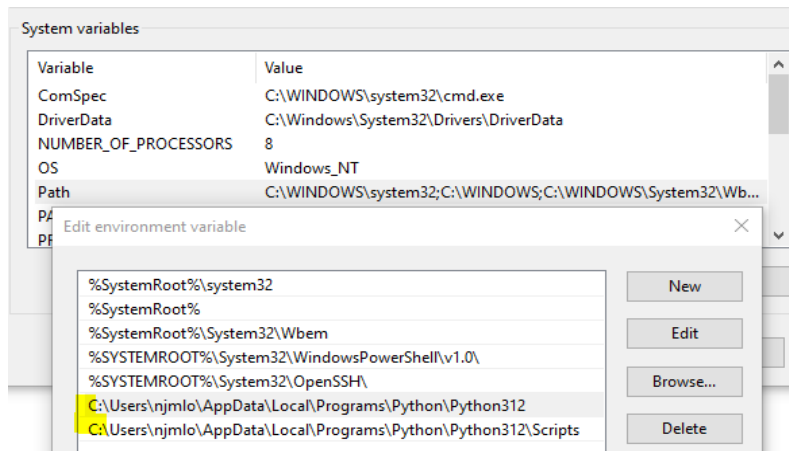
Python (version at the time of documentation):

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\njmlo> python -V
Python 3.12.6
PS C:\Users\njmlo>
```

Add Python environment variable under System Variable Path:



Determine Python version, install “requests” library:

```
(.venv) PS C:\Users\njmlo\PycharmProjects\project-a> python -V
Python 3.12.6
(.venv) PS C:\Users\njmlo\PycharmProjects\project-a> pip install requests
```

Pip version (version at the time of documentation):

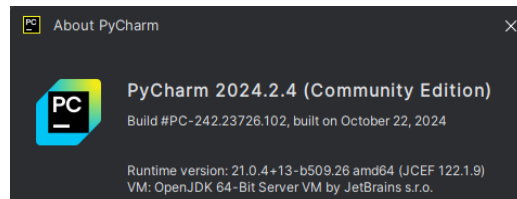
```
C:\Windows\System32>python.exe -m pip install --upgrade pip
Requirement already satisfied: pip in c:\users\njml\appdata\local\programs\python\python312\lib\site-packages (24.2)
Collecting pip
  Using cached pip-24.3.1-py3-none-any.whl.metadata (3.7 kB)
Using cached pip-24.3.1-py3-none-any.whl (1.8 MB)
Installing collected packages: pip
  Attempting uninstall: pip
    Found existing installation: pip 24.2
    Uninstalling pip-24.2:
      Successfully uninstalled pip-24.2
Successfully installed pip-24.3.1
```

```
C:\Windows\System32>pip --version
pip 24.3.1 from C:\Users\njml\AppData\Local\Programs\Python\Python312\Lib\site-packages\pip (python 3.12)
```

PyCharm Community Edition (IDE version at the time of documentation):

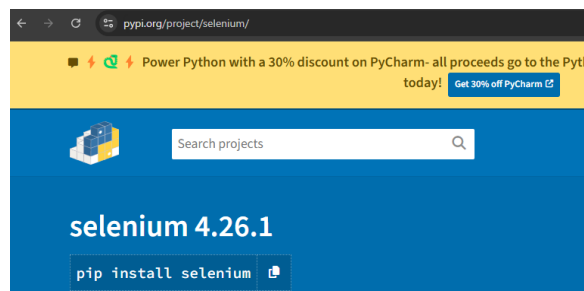
Download:

<https://www.jetbrains.com/pycharm/download>



Selenium install (version at the time of documentation):

```
C:\Windows\System32>pip install -U selenium
Collecting selenium
  Downloading selenium-4.26.1-py3-none-any.whl.metadata (7.1 kB)
Installing collected packages: sortedcontainers, websocket-client, typing_extensions, sniffio, pysocks, pycparser, h11, attrs, wsproto, outcome, cffi, trio, trio-websocket, selenium
Successfully installed attrs-24.2.0 cffi-1.17.1 h11-0.14.0 outcome-1.3.0.post0 pycparser-2.22 pysocks-1.7.1 selenium-4.26.1 sniffio-1.3.1 sortedcontainers-2.4.0 trio-0.27.0 trio-websocket-0.11.1 typing_extensions-4.12.2 websocket-client-1.8.0 wsproto-1.2.0
```



```
Administrator: Command Prompt
Microsoft Windows [Version 10.0.22631.4460]
(c) Microsoft Corporation. All rights reserved.

C:\Windows\System32>pip check selenium
No broken requirements found.

C:\Windows\System32>pip show selenium
Name: selenium
Version: 4.26.1
Summary: Official Python bindings for Selenium WebDriver
Home-page: https://www.selenium.dev
Author:
Author-email:
License: Apache 2.0
Location: C:\Users\njmlo\AppData\Local\Programs\Python\Python312\Lib\site-packages
Requires: certifi, trio, trio-websocket, typing_extensions, urllib3, websocket-client
Required-by:

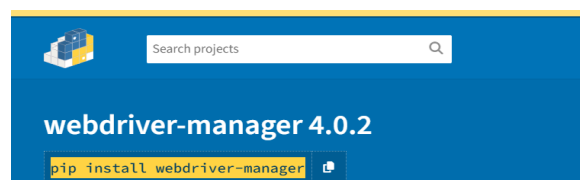
C:\Windows\System32>
```

Webdriver install (version at the time of documentation):

```
C:\Windows\System32>pip install webdriver-manager
Collecting webdriver-manager
  Downloading webdriver_manager-4.0.2-py2.py3-none-any.whl.metadata (12 kB)
```

```
Installing collected packages: python-dotenv, webdriver-manager
Successfully installed python-dotenv-1.0.1 webdriver-manager-4.0.2

C:\Windows\System32>
```

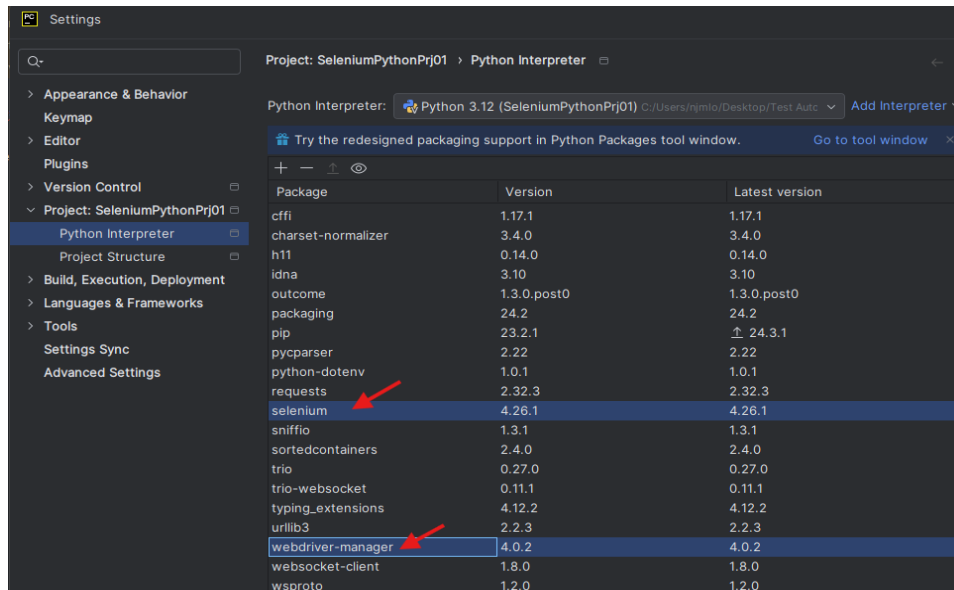


```
C:\Windows\System32>pip check webdriver-manager
No broken requirements found.

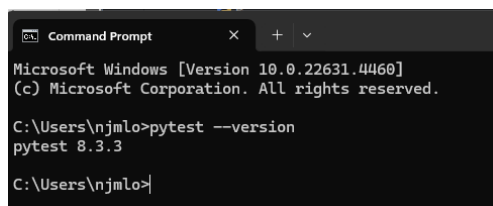
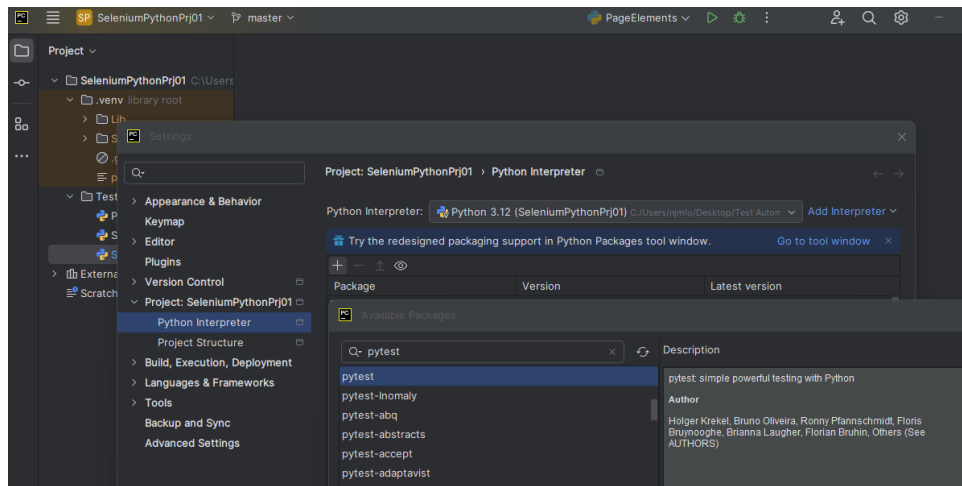
C:\Windows\System32>pip show webdriver-manager
Name: webdriver-manager
Version: 4.0.2
Summary: Library provides the way to automatically manage drivers for different browsers
Home-page: https://github.com/SergeyPirogov/webdriver_manager
Author: Sergey Pirogov
Author-email: automationremarks@gmail.com
License:
Location: C:\Users\njmlo\AppData\Local\Programs\Python\Python312\Lib\site-packages
Requires: packaging, python-dotenv, requests
Required-by:

C:\Windows\System32>
```

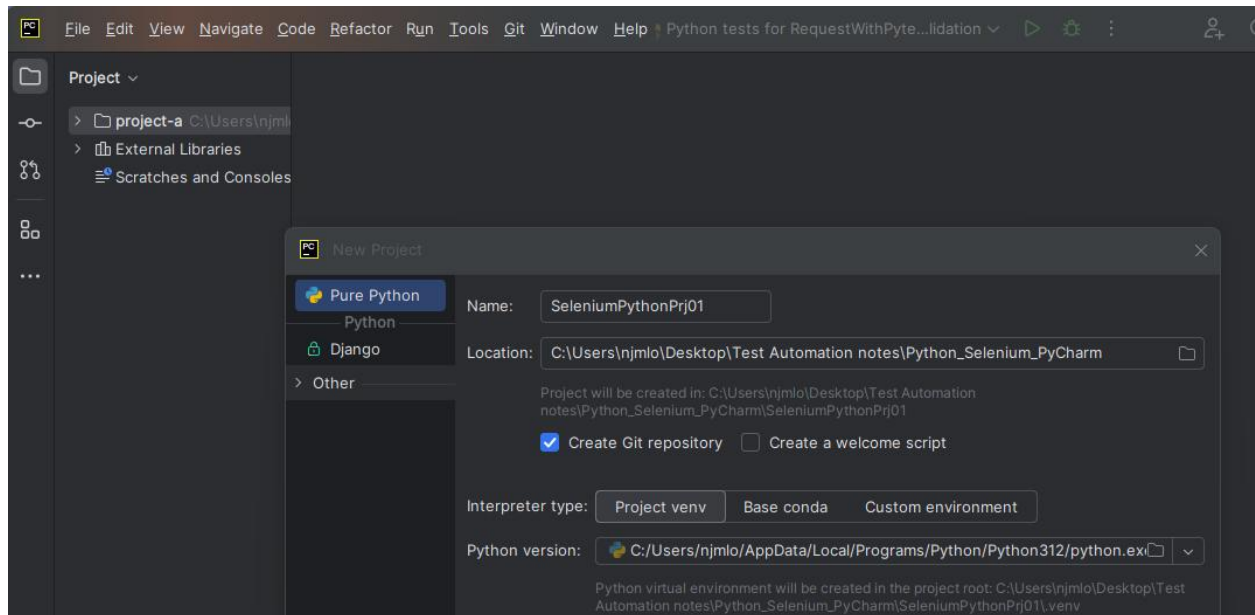
In Pycharm, validate Python interpreter:



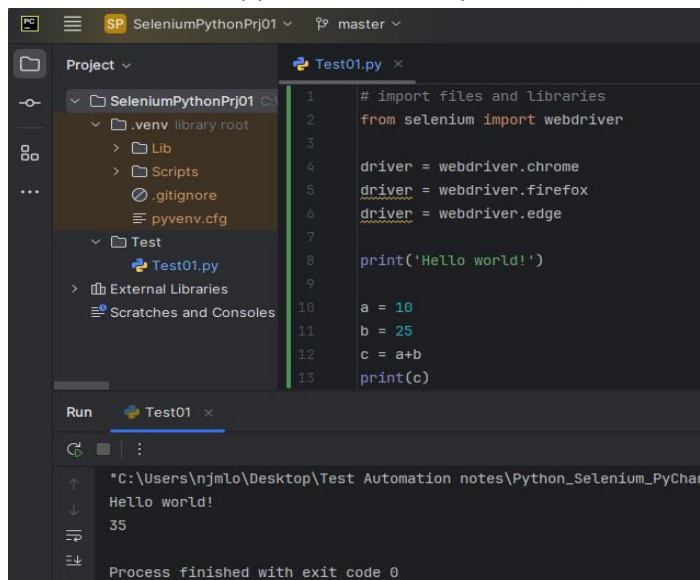
Add Pytest in the package:



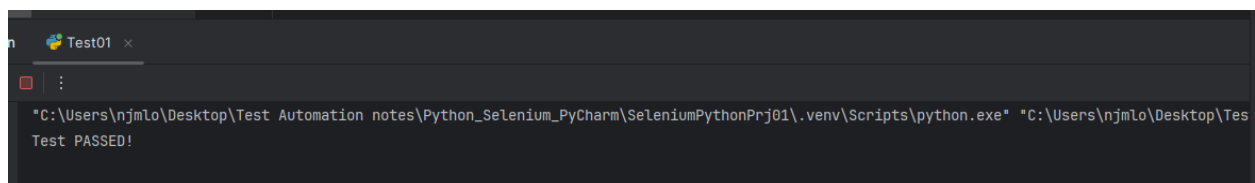
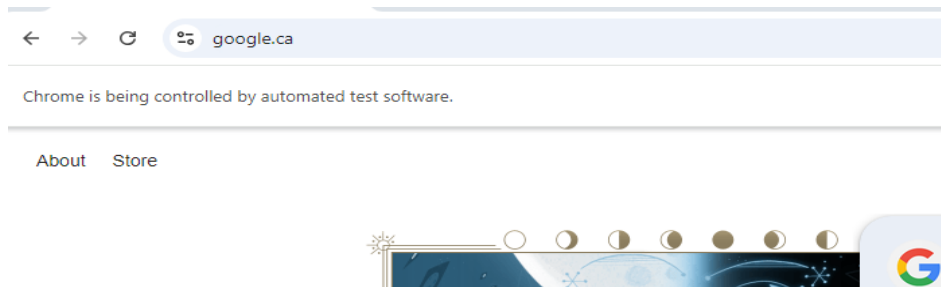
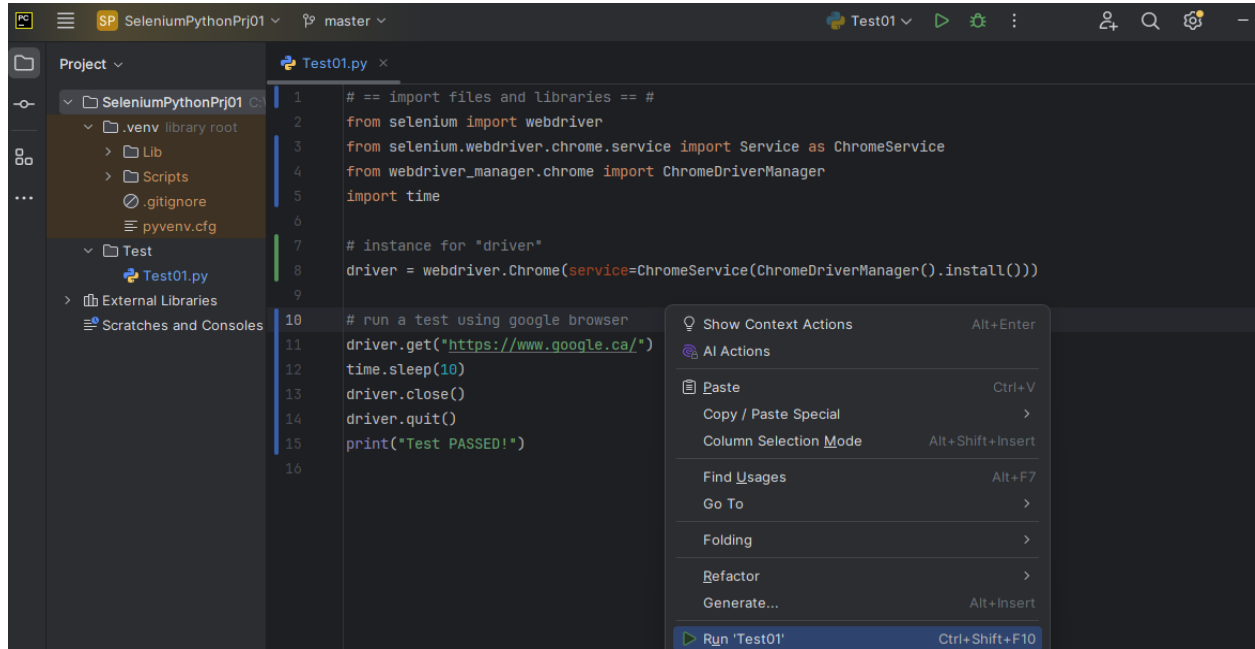
Create a New Project:



Create a folder and python file, run a quick test (this is to ensure initial components are setup properly):



2. Run a simple test using Google search page



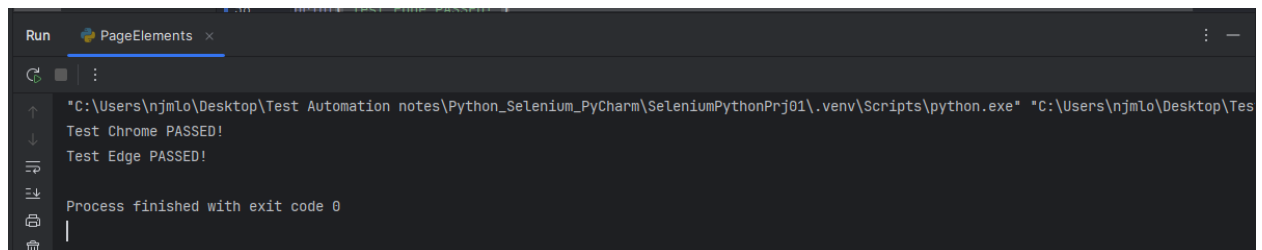
3. Interacting with elements in a web page

Chrome...

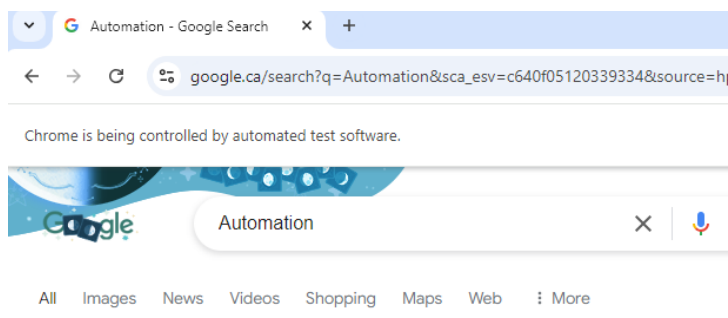
```
PageElements.py x
1  # == import files and libraries == #
2  from selenium import webdriver
3  from selenium.webdriver.chrome.service import Service as ChromeService
4  from selenium.webdriver.support.wait import WebDriverWait
5  from webdriver_manager.chrome import ChromeDriverManager
6  from selenium.webdriver.common.by import By
7  from selenium.webdriver.support import expected_conditions as ec
8  import time
9
10 # == extra import files and libraries == #
11 # from selenium.webdriver import Keys
12
13 # instance for CHROME Browser...
14 chromeBrowser = webdriver.Chrome(service=ChromeService(ChromeDriverManager().install()))
15 # run a test using browser
16 chromeBrowser.get('https://www.google.ca')
17
18 # find the Search box...
19 googleSearchBox = chromeBrowser.find_element(By.ID, value="APjFqb")
20 # insert a text...
21 googleSearchBox.send_keys("Automation")
22 # click the Google Search button...
23 waitChrome = WebDriverWait(chromeBrowser, timeout=5)
24 elementChrome = waitChrome.until(ec.element_to_be_clickable((By.NAME,"btnK")))
25 chromeBrowser.find_element(By.NAME, value="btnK").click()
26 # another way to test the search button (mirrors the action of hitting the enter button after typing what to search
27 # >> googleSearchBox.send_keys(Keys.RETURN)
28
29 # additional further actions...
30 time.sleep(5)
31 chromeBrowser.close()
32 chromeBrowser.quit()
33 # display a confirmation of test result...
34 print("Test Chrome PASSED!")
```

Edge...

```
36 # instance for EDGE Browser...
37 # if __name__ == '__main__':
38 edgeBrowser = webdriver.Edge()
39 # run a test using browser
40 edgeBrowser.get('https://www.google.ca')
41
42 # find the Search box...
43 googleSearchBox = edgeBrowser.find_element(By.ID, value: "APjFqb")
44 # insert a text...
45 googleSearchBox.send_keys("Automation")
46 # click the Google Search button...
47 waitEdge = WebDriverWait(edgeBrowser, timeout: 5)
48 elementEdge = waitEdge.until(ec.element_to_be_clickable((By.NAME, "btnK")))
49 edgeBrowser.find_element(By.NAME, value: "btnK").click()
50 # another way to test the search button (mirrors the action of hitting the enter button after typing what to search)
51 # >> googleSearchBox.send_keys(Keys.RETURN)
52
53 # additional further actions...
54 time.sleep(5)
55 edgeBrowser.close()
56 edgeBrowser.quit()
57 # display a confirmation of test result...
58 print("Test Edge PASSED!")
```



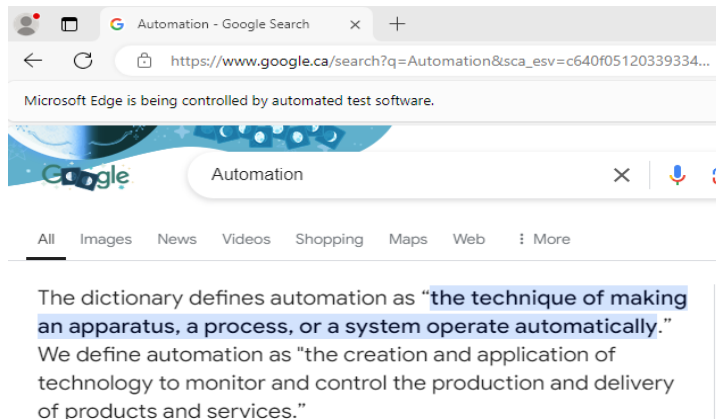
Chrome...



The dictionary defines automation as “the technique of making an apparatus, a process, or a system operate automatically.”

We define automation as “the creation and application of technology to monitor and control the production and delivery of products and services.”

Edge...

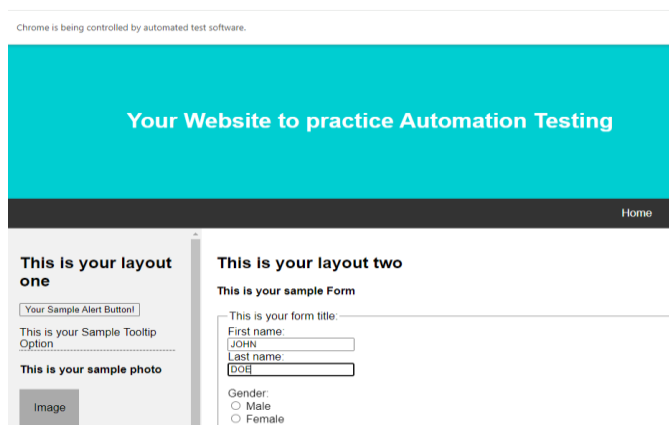


Run another test calling a different test page...

```
36 # another instance for CHROME Browser...
37 chromeBBrowser = webdriver.Chrome(service=ChromeService(ChromeDriverManager().install()))
38 # run another test using browser
39 chromeBBrowser.get('https://trytestingthis.netlify.app')
40 # insert text(s)...
41 time.sleep(5)
42 chromeBBrowser.find_element(By.ID, value: "fname").send_keys("JOHN")
43 chromeBBrowser.find_element(By.ID, value: "lname").send_keys("DOE")
44 time.sleep(5)
45 chromeBBrowser.find_element(By.XPATH, value: "//button[@class = 'btn btn-success']").click()
46
47 # additional further actions...
48 time.sleep(5)
49 chromeBBrowser.close()
50 chromeBBrowser.quit()
51 # display a confirmation of test result...
52 print("Test Chrome (B) PASSED!")
```

```
"C:\Users\njm\Lo\Desktop\Test Automation notes\Python_Selenium_PyCharm\SeleniumPythonPrj01\.venv\Scripts\python.exe" "C:\Users\njm\Lo\Desktop\T
Test Chrome (A) PASSED!
Test Chrome (B) PASSED!
Test Edge PASSED!

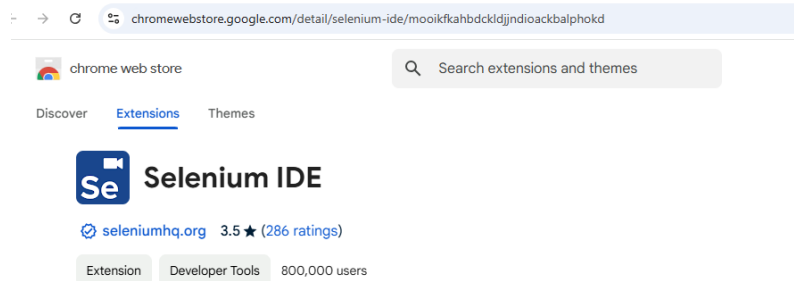
Process finished with exit code 0
```



4. Record, Playback, and Export (Selenium IDE)

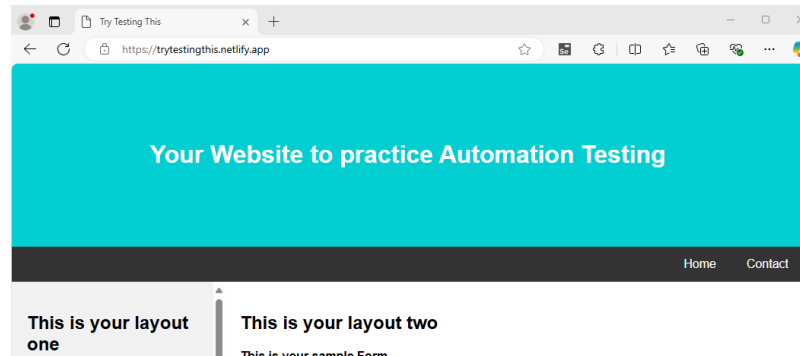
Download Selenium IDE and add to Chrome:

<https://www.selenium.dev/selenium-ide/>

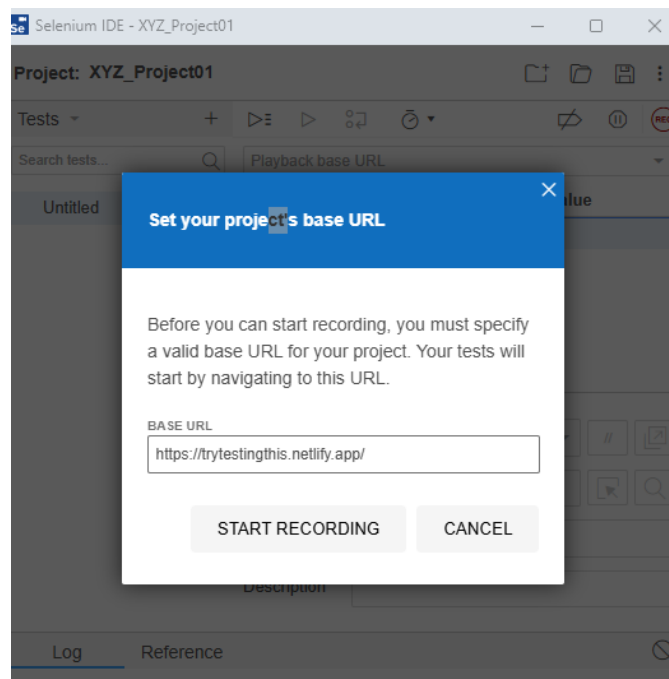


Test web site:

<https://trytestingthis.netlify.app/>



Record the test...



Stop recording...

Selenium IDE - XYZ_Project01*

Project: XYZ_Project01*

Tests +
Search tests...
Untitled*

https://trytestingthis.netlify.app

23
24
25
26
27
28
29
30

Name your new test

Please provide a name for your new test.

TEST NAME
test-MainScreen01

You can change it at any time by clicking the : icon next to its name in the tests panel.

OK LATE

Selenium IDE - XYZ_Project01*

Project: XYZ_Project01*

Tests +
Search tests...
test-MainScreen01*

https://trytestingthis.netlify.app

	Command	Target
23	click	id=a
24	click	id=quantity
25	type	id=quantity
26	click	id=quantity
27	click	id=quantity
28	type	id=quantity
29	click	id=quantity
30	click	name=message
31	type	name=message
32	assert element present	css=.btn
33	click	id=username
34	type	id=username
35	click	id=pwd
36	type	id=pwd

(Re)run the test...

Selenium IDE - XYZ_Project01*

Project: XYZ_Project01*

Tests + [Run] [Stop] [Refresh] [Clock]

Search tests... Run <https://trytestingthis.netlify.app>

✓ test-MainScreen01*	Command	Target	Value
27	✓ click	id=quantity	
28	✓ type	id=quantity	2
29	✓ click	id=quantity	
30	✓ click	name=message	
31	✓ type	name=message	The cat was playing in the garden. Great!
32	✓ assert element present	css=.btn	
33	✓ click	id=uname	
34	✓ type	id=uname	test
35	✓ click	id=pwd	

Command [] [//] []

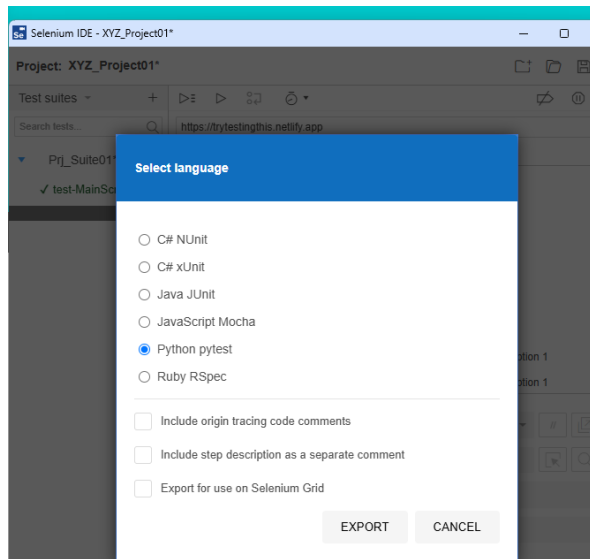
Target [] [] []

Value []

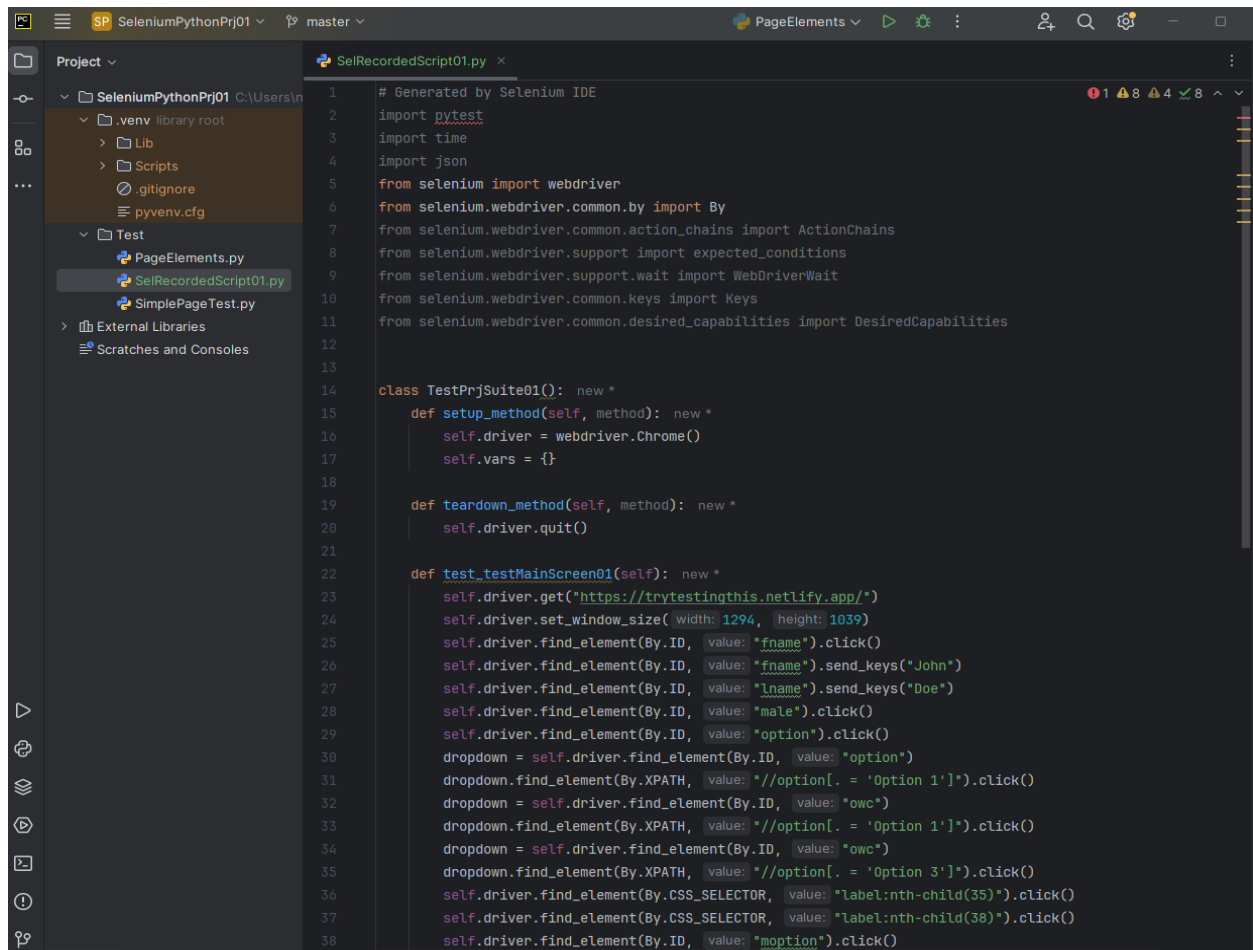
Description []

Log	Reference
34. type on id=uname with value test OK	16:31:28
35. click on id=pwd OK	16:31:28
36. type on id=pwd with value test OK	16:31:28
37. click on css=input:nth-child(10) OK	16:31:28
38. assertElementPresent on css=h2 OK	16:31:29
39. click on linkText=here OK	16:31:29
'test-MainScreen01' completed successfully	16:31:29

Export suite...



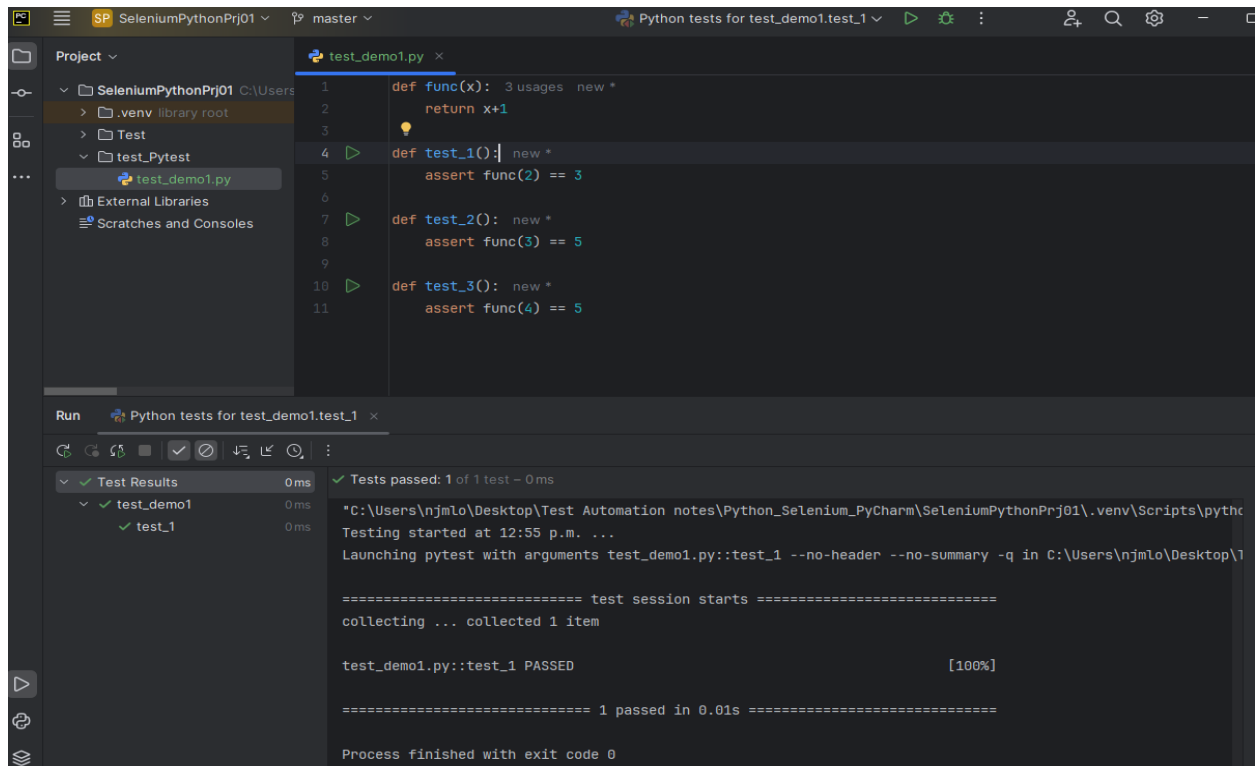
Create a python file in editor and dump the (recorded) code...



Note: this is one way to help in identifying what unique attribute can be used for element under test

5. Using Pytest with Selenium and Python

Quick test run of passed and failed cases...



The screenshot shows the PyCharm IDE with a project named 'SeleniumPythonPrj01'. The file explorer on the left shows the project structure, including a 'Test' directory with 'test_demo1.py'. The main editor displays the code for 'test_demo1.py':

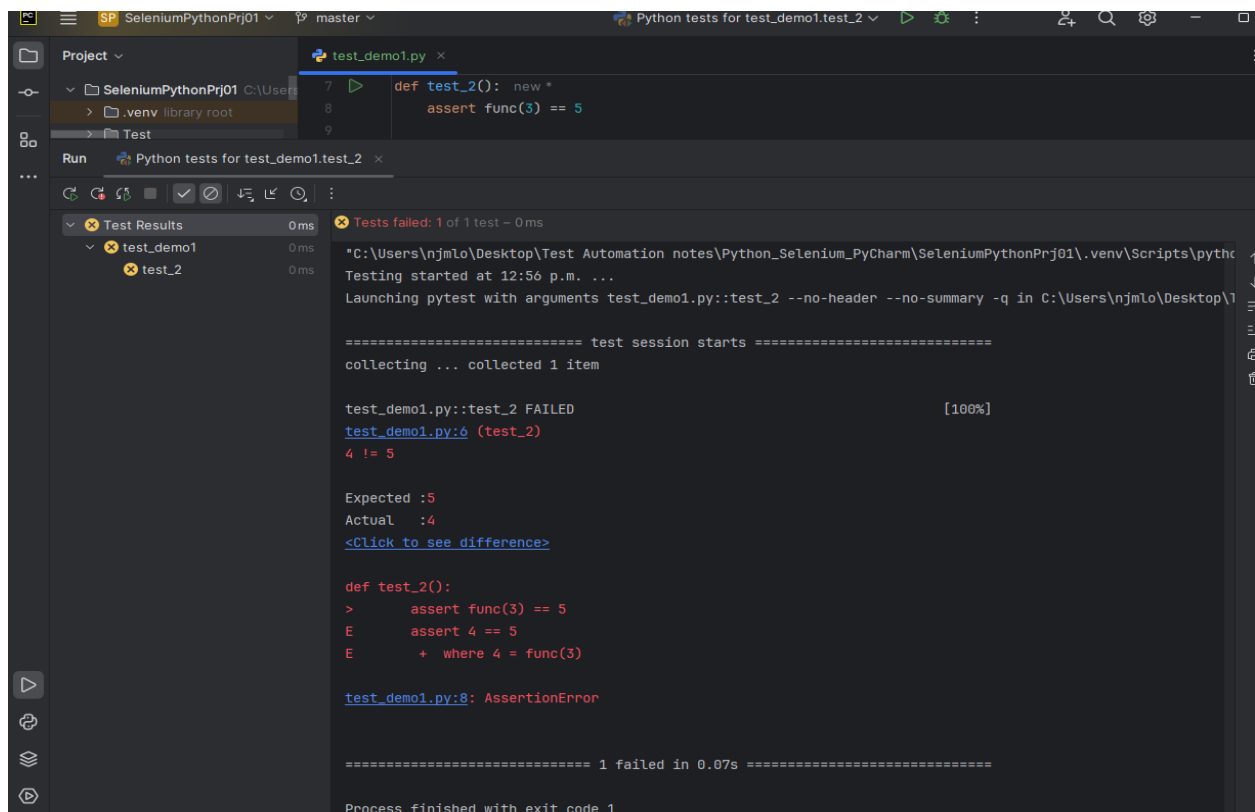
```
1 def func(x):  
2     return x+1  
3  
4 def test_1():  
5     assert func(2) == 3  
6  
7 def test_2():  
8     assert func(3) == 5  
9  
10 def test_3():  
11     assert func(4) == 5
```

The Run tool window at the bottom shows the results of running 'Python tests for test_demo1.test_1'. The test results are:

- Test Results: 0ms
- test_demo1: 0ms
- test_1: 0ms

The test output shows that the test passed:

```
Tests passed: 1 of 1 test - 0ms  
*C:\Users\njmlo\Desktop\Test Automation notes\Python_Selenium_PyCharm\SeleniumPythonPrj01\.venv\Scripts\pythc  
Testing started at 12:55 p.m. ...  
Launching pytest with arguments test_demo1.py::test_1 --no-header --no-summary -q in C:\Users\njmlo\Desktop\1  
===== test session starts =====  
collecting ... collected 1 item  
  
test_demo1.py::test_1 PASSED [100%]  
  
===== 1 passed in 0.01s =====  
Process finished with exit code 0
```



The screenshot shows the PyCharm IDE with the same project. The file explorer shows the 'Test' directory with 'test_demo1.py'. The main editor displays the code for 'test_demo1.py':

```
7 def test_2():  
8     assert func(3) == 5  
9
```

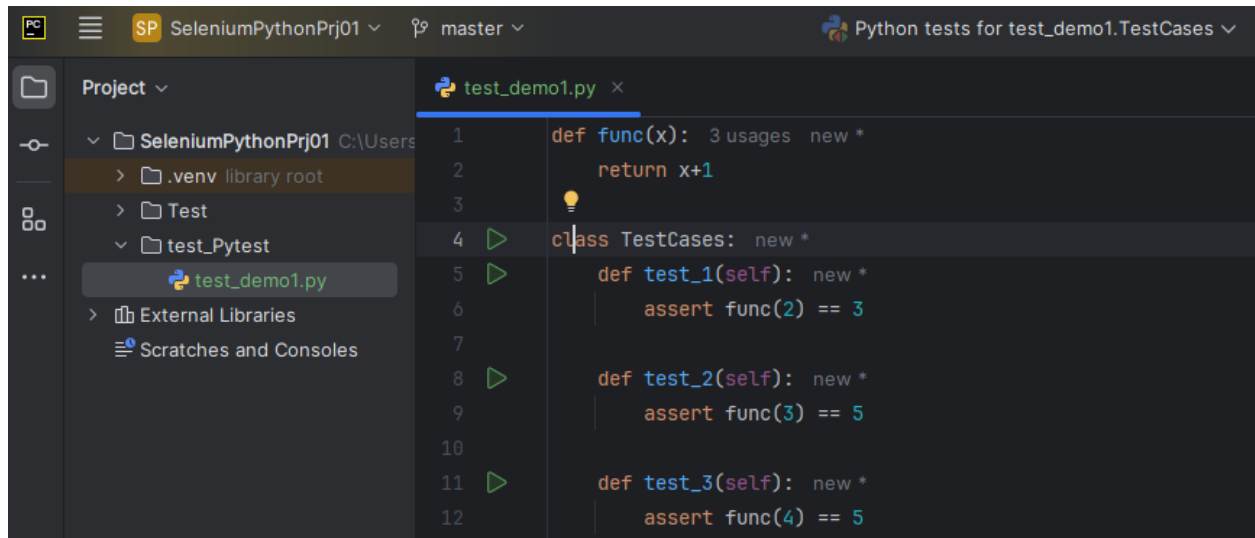
The Run tool window at the bottom shows the results of running 'Python tests for test_demo1.test_2'. The test results are:

- Test Results: 0ms
- test_demo1: 0ms
- test_2: 0ms

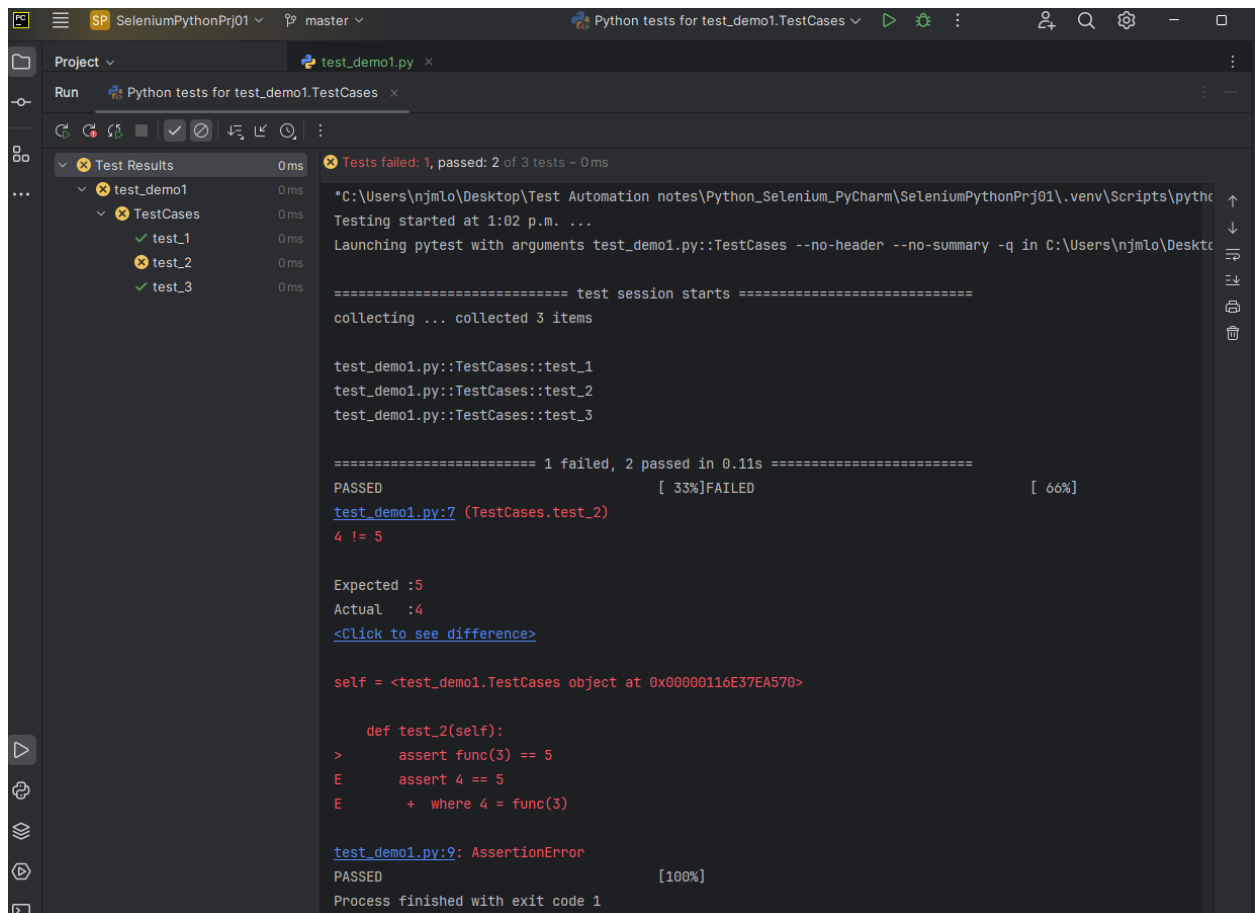
The test output shows that the test failed:

```
Tests failed: 1 of 1 test - 0ms  
*C:\Users\njmlo\Desktop\Test Automation notes\Python_Selenium_PyCharm\SeleniumPythonPrj01\.venv\Scripts\pythc  
Testing started at 12:56 p.m. ...  
Launching pytest with arguments test_demo1.py::test_2 --no-header --no-summary -q in C:\Users\njmlo\Desktop\1  
===== test session starts =====  
collecting ... collected 1 item  
  
test_demo1.py::test_2 FAILED [100%]  
test_demo1.py:6 (test_2)  
4 != 5  
  
Expected :5  
Actual   :4  
<Click to see difference>  
  
def test_2():  
>     assert func(3) == 5  
E       assert 4 == 5  
E       + where 4 = func(3)  
  
test_demo1.py:8: AssertionError  
  
===== 1 failed in 0.07s =====  
Process finished with exit code 1
```

Create and run a class for the set of test cases...

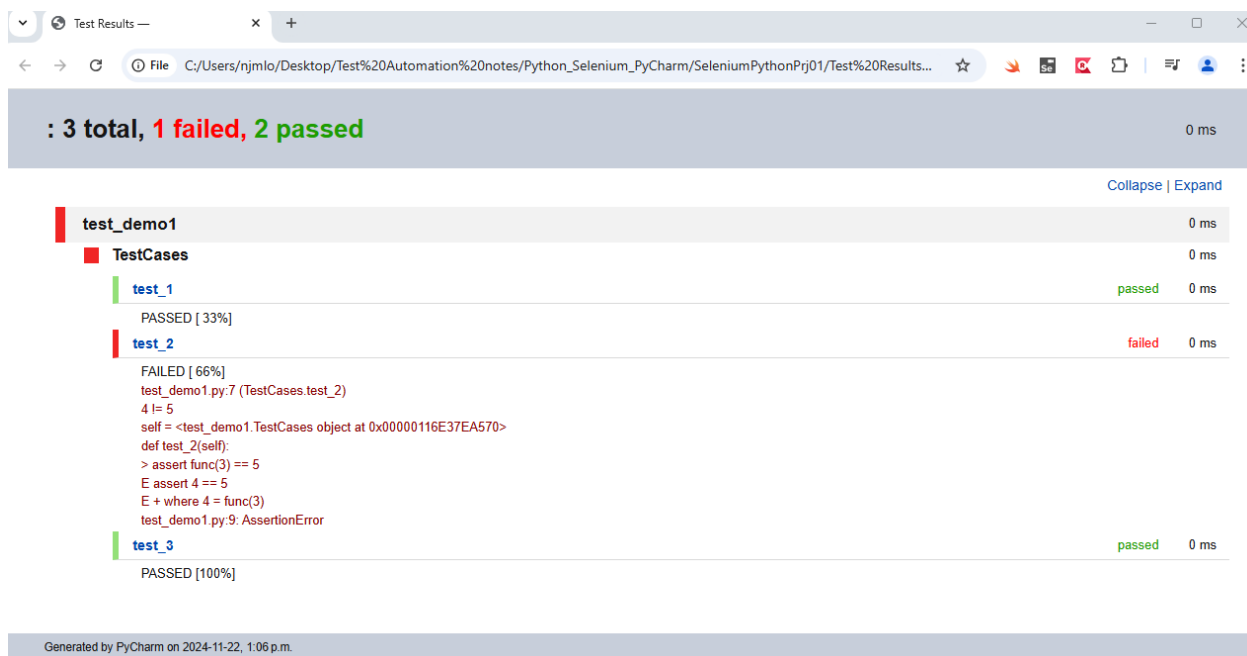
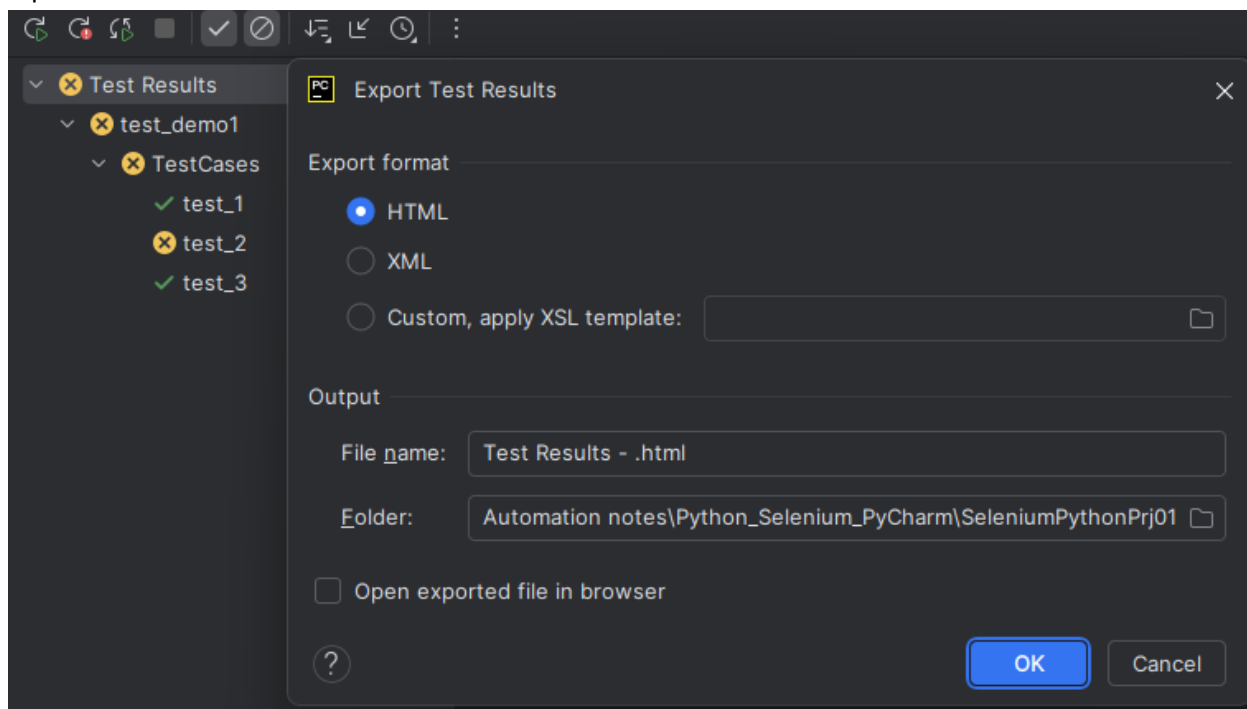


```
def func(x):  
    return x+1  
  
class TestCases:  
    def test_1(self):  
        assert func(2) == 3  
  
    def test_2(self):  
        assert func(3) == 5  
  
    def test_3(self):  
        assert func(4) == 5
```



```
Test Results  
test_demo1  
  TestCases  
    test_1  
    test_2  
    test_3  
test_demo1.py:7 (TestCases.test_2)  
4 != 5  
Expected :5  
Actual :4  
test_demo1.py:9: AssertionError  
PASSED [100%]  
Process finished with exit code 1
```

Export the test result...



Run Pytest from the cmd line or IDE terminal...

```
Command Prompt
Microsoft Windows [Version 10.0.22631.4460]
(c) Microsoft Corporation. All rights reserved.

C:\Users\njmlo>cd..
C:\Users>cd..
C:\>cd C:\Users\njmlo\Desktop\Test Automation notes\Python_Selenium_PyCharm\SeleniumPythonPrj01

C:\Users\njmlo\Desktop\Test Automation notes\Python_Selenium_PyCharm\SeleniumPythonPrj01> pytest .\test_Pytest\test_demo1.py
===== test session starts =====
platform win32 -- Python 3.12.6, pytest-8.3.3, pluggy-1.5.0
rootdir: C:\Users\njmlo\Desktop\Test Automation notes\Python_Selenium_PyCharm\SeleniumPythonPrj01
plugins: html-4.1.1, metadata-3.1.1
collected 3 items

test_Pytest\test_demo1.py .F. [100%]

===== FAILURES =====
TestCases.test_2

self = <test_demo1.TestCases object at 0x000001FA559C6960>

    def test_2(self):
>     assert func(3) == 5
E       assert 4 == 5
E       + where 4 = func(3)

test_Pytest\test_demo1.py:9: AssertionError
===== short test summary info =====
FAILED test_Pytest/test_demo1.py::TestCases::test_2 - assert 4 == 5
===== 1 failed, 2 passed in 0.10s =====

C:\Users\njmlo\Desktop\Test Automation notes\Python_Selenium_PyCharm\SeleniumPythonPrj01>
```

Using the cmd line to find a keyword in the test case to run...

```
PS C:\Users\njmlo\Desktop\Test Automation notes\Python_Selenium_PyCharm\SeleniumPythonPrj01> pytest .\test_Pytest -k test_1
===== test session starts =====
platform win32 -- Python 3.12.6, pytest-8.3.3, pluggy-1.5.0
rootdir: C:\Users\njmlo\Desktop\Test Automation notes\Python_Selenium_PyCharm\SeleniumPythonPrj01
plugins: html-4.1.1, metadata-3.1.1
collected 3 items / 2 deselected / 1 selected

test_Pytest\test_demo1.py . [100%]

===== 1 passed, 2 deselected in 0.01s =====

PS C:\Users\njmlo\Desktop\Test Automation notes\Python_Selenium_PyCharm\SeleniumPythonPrj01>
```

```
PS C:\Users\njmlo\Desktop\Test Automation notes\Python_Selenium_PyCharm\SeleniumPythonPrj01> python -m pytest .\test_Pytest\
===== test session starts =====
platform win32 -- Python 3.12.6, pytest-8.3.3, pluggy-1.5.0
rootdir: C:\Users\njmlo\Desktop\Test Automation notes\Python_Selenium_PyCharm\SeleniumPythonPrj01
plugins: html-4.1.1, metadata-3.1.1
collected 3 items

test_Pytest\test_demo1.py .F. [100%]

===== FAILURES =====
TestCases.test_2

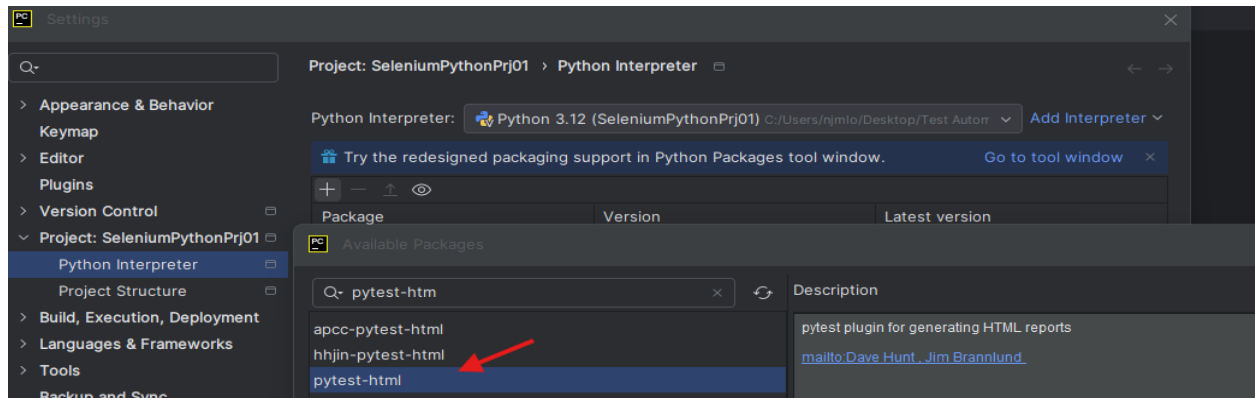
self = <test_demo1.TestCases object at 0x000001FADBE81FA0>

    def test_2(self):
>     assert func(3) == 5
E       assert 4 == 5
E       + where 4 = func(3)

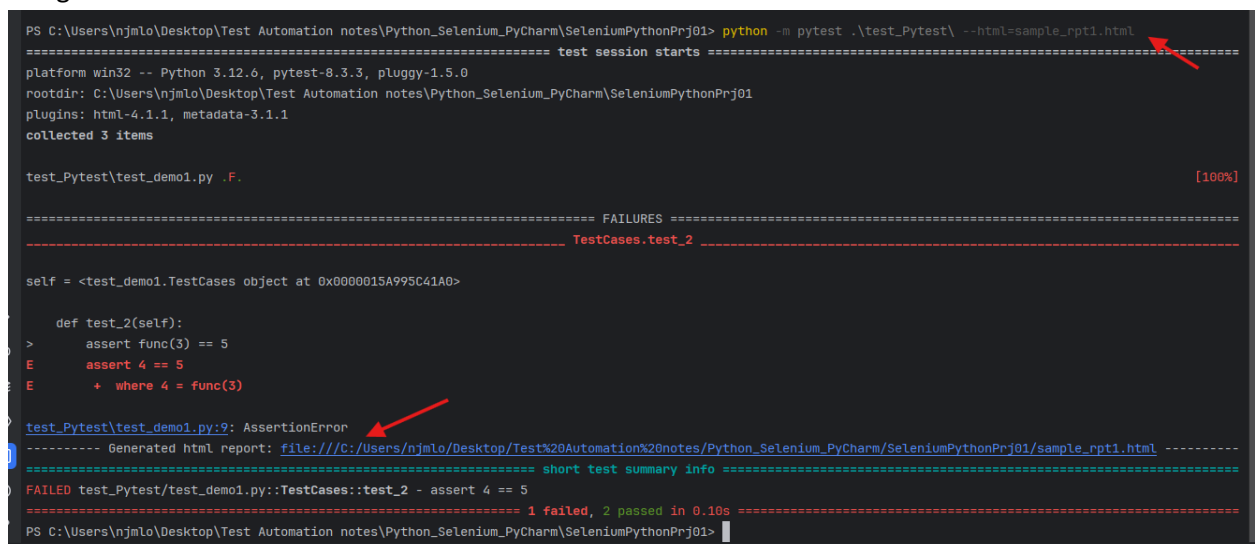
test_Pytest\test_demo1.py:9: AssertionError
===== short test summary info =====
FAILED test_Pytest/test_demo1.py::TestCases::test_2 - assert 4 == 5
===== 1 failed, 2 passed in 0.08s =====

PS C:\Users\njmlo\Desktop\Test Automation notes\Python_Selenium_PyCharm\SeleniumPythonPrj01>
```

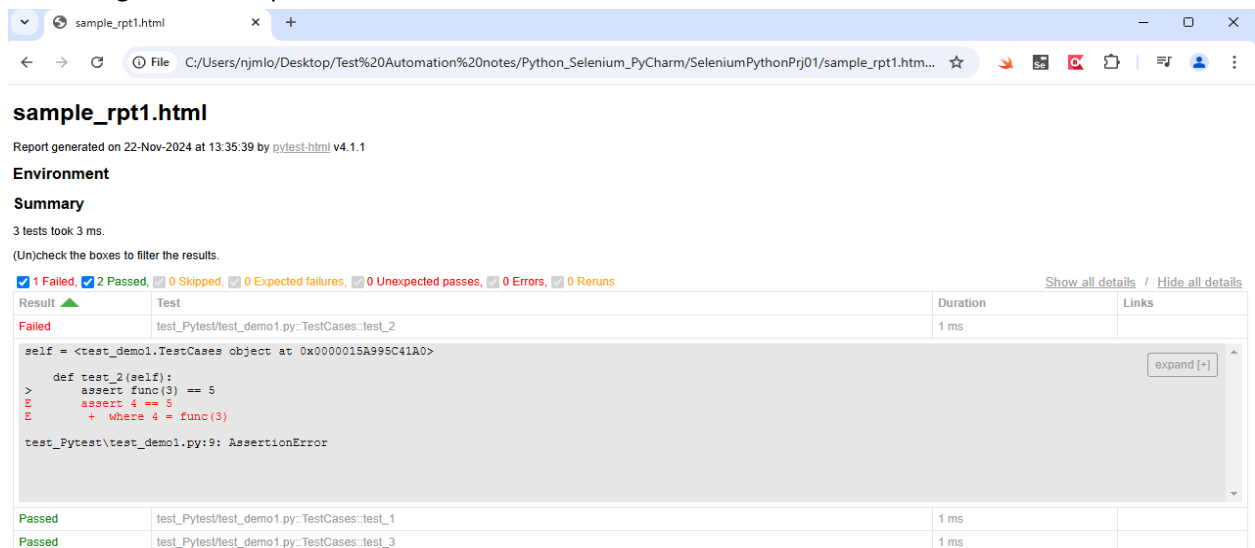
Add another type of reporting tool...



Using the cmd line or terminal...



View the generated report...



6. More Selenium, Python, Pytest sampling codes and test run

Run a google search test...

The screenshot shows the PyCharm IDE with a project named 'SeleniumPythonPrj01'. The file explorer on the left shows a directory structure with files like 'PageElements.py', 'SelRecordedScript01.py', 'SimplePage.py', 'test_Pytest', 'selenium_pytest01.py', and 'selenium_pytest02.py'. The 'Run' button is highlighted with a red arrow. The main editor displays the code for 'selenium_pytest02.py', which imports Selenium and Pytest libraries and defines a test function 'test_googles_search()'. A red arrow points to the 'test_googles_search' function. Below the code editor, a browser window titled 'Automation - Google Search' is open, showing the Google search page with the search term 'Automation'. A red arrow points to the search bar.

```
1  # == import files and libraries == #
2  import pytest
3  from selenium import webdriver
4  from selenium.webdriver.chrome.service import Service as ChromeService
5  from selenium.webdriver.support.wait import WebDriverWait
6  from webdriver_manager.chrome import ChromeDriverManager
7  from selenium.webdriver.common.by import By
8  from selenium.webdriver.support import expected_conditions as ec
9  import time
10
11 # make this as a class with "test" as a starting class name so Pytest is able to recognize it as a test
12 def test_googles_search():
13     # instance for CHROME Browser...
14     chrome_browser = webdriver.Chrome(service=ChromeService(ChromeDriverManager().install()))
15     # chrome_browser = webdriver.Chrome()
16     # run a test using browser
17     chrome_browser.get('https://www.google.ca')
18     # find the Search box...
19     google_search_box = chrome_browser.find_element(By.ID, 'APjFqb')
```

The screenshot shows the PyCharm Run console output for the test 'test_googles_search'. The output indicates that the test passed successfully. The console text is as follows:

```
Run Python tests for selenium_pytest02.test_googlesSearch
✓ Test Results 13 sec 229 ms
  ✓ selenium_pytest02 13 sec 229 ms
    ✓ test_googlesSearch 13 sec 229 ms

Tests passed: 1 of 1 test - 13 sec 229 ms

"C:\Users\njmlo\Desktop\Test Automation notes\Python_Selenium_PyCharm\SeleniumPythonPrj01\.venv\Scripts\python.exe"
Testing started at 2:24 p.m. ...
Launching pytest with arguments selenium_pytest02.py::test_googlesSearch --no-header --no-summary -q in C:\Users\njmlo\Desktop\Test Automation notes\Python_Selenium_PyCharm\SeleniumPythonPrj01

===== test session starts =====
collecting ... collected 1 item

selenium_pytest02.py::test_googlesSearch PASSED [100%]Test Chrome PASSED!

===== 1 passed in 18.04s =====

Process finished with exit code 0
```

Create a Pytest fixtures and leverage what can be set up and tear down...

The screenshot shows the PyCharm IDE with a file named `selenium_pytest02.py` open. The code defines a Pytest fixture `chrome_browser` and a test function `test_googles_search`. A red arrow points to the `chrome_browser` fixture definition. Below the code editor, the 'Run' toolbar is visible, and a red arrow points to the 'test_googles_search' test item in the 'Running tests...' panel. To the right, a browser window is open showing a Google search for 'Automation'. A red arrow points to the search bar. The browser address bar shows the URL `google.ca/search?q=Automation&scas_esv=0c814c2424608e41&source=hp&ei=hRhBZ6vwH7KC0PEP5q-osAM&iifl...`. The search results show 'Automation' as the top result.

```
9 # Create a Pytest fixture and leverage what can be set up and tear down...
10 @pytest.fixture() 4 usages new *
11 def chrome_browser():
12     # instance for CHROME Browser...
13     chrome_browser = webdriver.Chrome(service=ChromeService(ChromeDriverManager().install())) #chromeBrowser = webdriver.Chrome()
14     chrome_browser.implicitly_wait(10)
15     yield chrome_browser
16     chrome_browser.close()
17     chrome_browser.quit()
18
19 # make this as a class with "test" as a starting class name so Pytest is able to recognize it as a set to be run
20 def test_googles_search(chrome_browser): new *
21     # run a test using browser
22     chrome_browser.get('https://www.google.ca')
23     # find the Search box...
24     google_search_box = chrome_browser.find_element(By.ID, value="APjFqb")
25     # insert a text...
26     google_search_box.send_keys("Automation")
27     # click the Google Search button...
28     chrome_browser.find_element(By.NAME, value="btnK").click()
29     # additional further actions...
30     time.sleep(10)
31     # display a confirmation of test result...
32     print("Test Chrome PASSED!")
```

The screenshot shows the 'Run' output window in PyCharm. The output displays the test results for `test_googles_search`. The test passed successfully. The output text is as follows:

```
Run Python tests for selenium_pytest02.test_googles_search
Test Results 12 sec 609 ms
selenium_pytest(12 sec 609 ms)
test_googles_ 12 sec 609 ms
Tests passed: 1 of 1 test - 12 sec 609 ms
"C:\Users\njm\o\Desktop\Test Automation notes\Python_Selenium_PyCharm\SeleniumPythonPrj01\.venv\Scripts\pythc
Testing started at 3:49 p.m. ...
Launching pytest with arguments selenium_pytest02.py::test_googles_search --no-header --no-summary -q in C:\U

===== test session starts =====
collecting ... collected 1 item

selenium_pytest02.py::test_googles_search PASSED [100%]Test Chrome PASSED!

===== 1 passed in 22.35s =====

Process finished with exit code 0
```

Parameterization in Pytest...

```
selenium_pytest03.py x
10
11 # Create a Pytest fixture and leverage what can be set up and tear down...
12 @pytest.fixture() 6 usages
13 def chrome_browser():
14     # instance for CHROME Browser...
15     service = ChromeService(ChromeDriverManager().install())
16     chrome_browser = webdriver.Chrome(service=service)
17     # chrome_browser = webdriver.Chrome(service=ChromeService(ChromeDriverManager().install())) #chromeBrowser = webdriver
18     # additional further actions...
19     chrome_browser.implicitly_wait(5)
20     yield chrome_browser
21     chrome_browser.close()
22     chrome_browser.quit()
23
24 # Parameterization in Pytest...
25 @pytest.mark.parametrize('username, password',[
26     ('test','test'),
27     ('wrong_usr1', 'wrong_pwd1'),
28     ('wrong_usr2', 'wrong_pwd2'),
29     ('test', 'test'),
30 ])
31
32 # make this as a class with "test" as a starting class name so Pytest is able to recognize it as a set to be run
33 def test_googles_search(chrome_browser, username, password):
34     # run another test using browser
35     chrome_browser.get('https://trytestingthis.netlify.app')
36     time.sleep(10)
37     chrome_browser.find_element(By.ID, value="uname").send_keys(username)
38     chrome_browser.find_element(By.ID, value="pwd").send_keys(password)
39     time.sleep(2)
40     chrome_browser.find_element(By.XPATH, value="//input[@value = 'Login']").click()
41     # additional further actions...
42     assert "Successful" in chrome_browser.page_source
43     time.sleep(5)
44     # display a confirmation of test result...
45     print("Test Chrome run DONE!")
```

Run Python tests for selenium_pytest03.test_googles_search x		
Tests failed: 2, passed: 2 of 4 tests		
Running tests...	1m 02s 450ms	PASSED [100%]Test Chrome run DONE!
selenium_pytest03	1m 02s 450ms	
test_googles_search	1m 02s 450ms	
✓ (test-test0)	17 sec 791 ms	
✗ (wrong_usr1-wrong_pwd1)	12 sec 937 ms	
✗ (wrong_usr2-wrong_pwd2)	12 sec 814 ms	
✓ (test-test1)	18 sec 908 ms	

Using the cmd line or terminal...

```
PS C:\Users\njmlo\Desktop\Test Automation notes\Python_Selenium_PyCharm\SeleniumPythonPrj01> pytest .\test_Pytest\selenium_pytest03.py
===== test session starts =====
platform win32 -- Python 3.12.6, pytest-8.3.3, pluggy-1.5.0
rootdir: C:\Users\njmlo\Desktop\Test Automation notes\Python_Selenium_PyCharm\SeleniumPythonPrj01
plugins: html-4.1.1, metadata-3.1.1
collected 4 items

test_Pytest\selenium_pytest03.py
DevTools listening on ws://127.0.0.1:52617/devtools/browser/d2a02bae-d3a0-4366-addc-a7aff4ff69f4
Created TensorFlow Lite XNNPACK delegate for CPU.
.
DevTools listening on ws://127.0.0.1:52675/devtools/browser/91aa3ddf-6c28-4c61-9989-d9949bae4977
F
DevTools listening on ws://127.0.0.1:52714/devtools/browser/b67bb04e-8aa1-47cb-8762-22aabf095c54
Created TensorFlow Lite XNNPACK delegate for CPU.
F
DevTools listening on ws://127.0.0.1:52753/devtools/browser/c2ceb1a-e23d-4103-bbc4-e1097bff41c7
Created TensorFlow Lite XNNPACK delegate for CPU.

[100%]

===== short test summary info =====
FAILED test_Pytest\selenium_pytest03.py::test_googles_search[wrong_usr1-wrong_pwd1] - selenium.common.exceptions.UnexpectedAlertPresentException: Alert Text: Wrong
Credentials! Try again!
FAILED test_Pytest\selenium_pytest03.py::test_googles_search[wrong_usr2-wrong_pwd2] - selenium.common.exceptions.UnexpectedAlertPresentException: Alert Text: Wrong
Credentials! Try again!
===== 2 failed, 2 passed in 89.81s (0:01:29) =====
PS C:\Users\njmlo\Desktop\Test Automation notes\Python_Selenium_PyCharm\SeleniumPythonPrj01>
```

Export the test result...

File C:/Users/njmlo/Desktop/Test%20Automation%20notes/Python_Selenium_PyCharm/SeleniumPythonPrj01/Test%20Results... 16:35 CRLF UTF-8 4 spaces Python 3.12 (SeleniumPythonPrj01)

: 4 total, 2 failed, 2 passed 1 m 2 s

Collapse | Expand

selenium_pytest03		1 m 2 s
test_googles_search		1 m 2 s
(test-test0)	passed	17.79 s
(wrong_usr1-wrong_pwd1)	failed	12.94 s
(wrong_usr2-wrong_pwd2)	failed	12.81 s
(test-test1)	passed	18.91 s

Generated by PyCharm on 2024-11-22, 4:47 p.m.

```
PS C:\Users\njmlo\Desktop\Test Automation notes\Python_Selenium_PyCharm\SeleniumPythonPrj01> pytest .\test_Pytest\selenium_pytest03.py --html=sample.rpt2.html
===== test session starts =====
platform win32 -- Python 3.12.6, pytest-8.3.3, pluggy-1.5.0
rootdir: C:\Users\njmlo\Desktop\Test Automation notes\Python_Selenium_PyCharm\SeleniumPythonPrj01
plugins: html-4.1.1, metadata-3.1.1
collected 4 items

-- Generated html report: file:///C:/Users/njmlo/Desktop/Test%20Automation%20notes/Python_Selenium_PyCharm/SeleniumPythonPrj01/sample.rpt2.html --
===== short test summary info =====
FAILED test_Pytest\selenium_pytest03.py::test_googles_search[wrong_usr1-wrong_pwd1] - selenium.common.exceptions.UnexpectedAlertPresentException: A
lert Text: Wrong Credentials! Try again!
FAILED test_Pytest\selenium_pytest03.py::test_googles_search[wrong_usr2-wrong_pwd2] - selenium.common.exceptions.UnexpectedAlertPresentException: A
lert Text: Wrong Credentials! Try again!
===== 2 failed, 2 passed in 91.81s (0:01:31) =====
PS C:\Users\njmlo\Desktop\Test Automation notes\Python_Selenium_PyCharm\SeleniumPythonPrj01>
```

← → ↺ 📄 File C:/Users/njmlo/Desktop/Test%20Automation%20notes/Python_Selenium_PyCharm/SeleniumPy... 🔍 ☆ 🚀 📄 📄 📄 📄 📄 📄 📄 📄

sample.rpt2.html

Report generated on 22-Nov-2024 at 17:05:53 by [pytest-html](#) v4.1.1

Environment

Python	3.12.6
Platform	Windows-11-10.0.22631-SP0
Packages	<ul style="list-style-type: none"> • pytest: 8.3.3 • pluggy: 1.5.0
Plugins	<ul style="list-style-type: none"> • html: 4.1.1 • metadata: 3.1.1

Summary

4 tests took 00:01:22.

(Un)check the boxes to filter the results.

☒ 2 Failed,
 ☒ 2 Passed,
 ☐ 0 Skipped,
 ☒ 0 Expected failures,
 ☒ 0 Unexpected passes,
 ☐ 0 Errors,
 ☐ 0 Reruns

[Show all details](#) / [Hide all details](#)

Result	Test	Duration	Links
Failed	test_Pytest/selenium_pytest03.py::test_googles_search[wrong_usr1-wrong_pwd1]	00:00:18	
<pre> chrome_browser = <selenium.webdriver.chrome.webdriver.WebDriver (session="4681c7305884e3807172aae537e5e50f")>, username = 'wrong_usr1' password = 'wrong_pwd1' @pytest.mark.parametrize('username, password',[('test', 'test'), ('wrong_usr1', 'wrong_pwd1'), ('wrong_usr2', 'wrong_pwd2'), ('test', 'test'),]) # make this as a class with "test" as a starting class name so Pytest is able to recognize it as a set to be run def test_googles_search(chrome_browser, username, password): # run another test using Browser chrome_browser.get('https://trytestingthis.netlify.app') time.sleep(10) chrome_browser.find_element(By.ID, "uname").send_keys(username) chrome_browser.find_element(By.ID, "pwd").send_keys(password) </pre>			
Failed	test_Pytest/selenium_pytest03.py::test_googles_search[wrong_usr2-wrong_pwd2]	00:00:19	
<pre> chrome_browser = <selenium.webdriver.chrome.webdriver.WebDriver (session="1fd25951a47539361e3d9f111b4063fa")>, username = 'wrong_usr2' password = 'wrong_pwd2' @pytest.mark.parametrize('username, password',[('test', 'test'), ('wrong_usr1', 'wrong_pwd1'), ('wrong_usr2', 'wrong_pwd2'), ('test', 'test'),]) # make this as a class with "test" as a starting class name so Pytest is able to recognize it as a set to be run def test_googles_search(chrome_browser, username, password): # run another test using Browser chrome_browser.get('https://trytestingthis.netlify.app') time.sleep(10) chrome_browser.find_element(By.ID, "uname").send_keys(username) chrome_browser.find_element(By.ID, "pwd").send_keys(password) </pre>			
Passed	test_Pytest/selenium_pytest03.py::test_googles_search[test-test0]	00:00:23	
Passed	test_Pytest/selenium_pytest03.py::test_googles_search[test-test1]	00:00:22	

Note: if running the test in the command line is throwing issues, downgrade (for now) the Selenium to version 4.9.1...

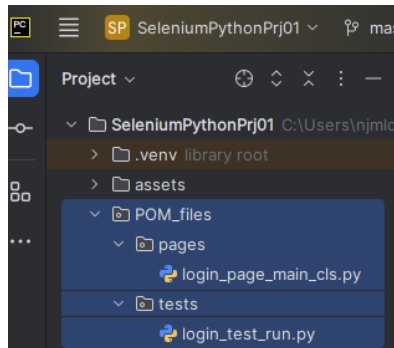
```

C:\Users\njmlo>pip install selenium==4.9.1
Collecting selenium==4.9.1
  Downloading selenium-4.9.1-py3-none-any.whl.metadata (7.2 kB)

```

7. Page Object Model (POM)

Create folders and files...



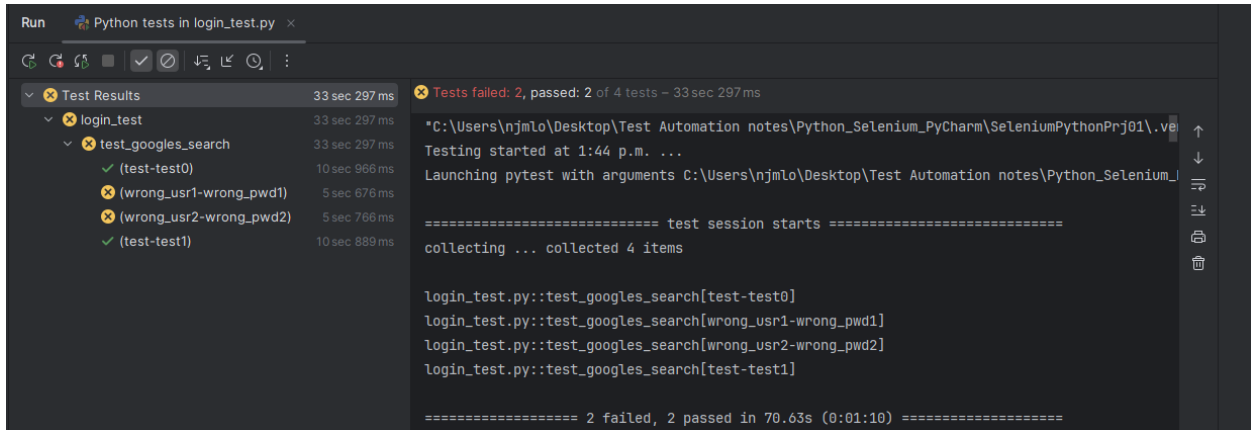
Web elements and methods (in login_page_main_cls.py file)...

```
login_test_run.py  login_page_main_cls.py x
1  # == import files and libraries == #
2  from selenium.webdriver.common.by import By
3
4  # define the class...
5  class LoginPage: 2 usages  njmlopez17
6
7      # define the constructor function that takes the driver parameter...
8      def __init__(self, driver): njmlopez17
9          self.driver = driver
10         # identify the page element locators...
11         self.username_textbox = (By.ID, "uname")
12         self.password_textbox = (By.ID, "pwd")
13         self.login_button = (By.XPATH, "//input[@value = 'Login']")
14
15     # define the action function...
16     def login_actions(self, url, username, password): 1 usage njmlopez17
17         # open the page...
18         self.driver.get(url)
19         # enter username...
20         self.driver.find_element(*self.username_textbox).send_keys(username)
21         # enter password...
22         self.driver.find_element(*self.password_textbox).send_keys(password)
23         # click button...
24         self.driver.find_element(*self.login_button).click()
25         # additional further actions...
26         assert "Successful" in self.driver.page_source
```


Test methods (in login_test_run.py file)...

```
login_test_run.py x login_page_main_cls.py
1  # == import files and libraries == #
2  import pytest
3  from selenium import webdriver
4  from selenium.webdriver.chrome.service import Service as ChromeService
5  from webdriver_manager.chrome import ChromeDriverManager
6  import time
7
8  # import the login page class so it can be used in this set of test run...
9  from POM_files.pages.login_page_main_cls import LoginPage
10
11 # Create a Pytest fixture and leverage what can be set up and tear down...
12 @pytest.fixture() 2 usages njmlopez17
13 def driver():
14     # instance for CHROME Browser...
15     service = ChromeService(ChromeDriverManager().install())
16     driver = webdriver.Chrome(service=service)
17     # additional further actions...
18     driver.implicitly_wait(5)
19     yield driver
20     driver.close()
21     driver.quit()
22
23 # Parameterization in Pytest...
24 @pytest.mark.parametrize('username, password',[ njmlopez17
25     ('test','test'),
26     ('wrong_usr1', 'wrong_pwd1'),
27     ('wrong_usr2', 'wrong_pwd2'),
28     ('test', 'test'),
29 ])
30
31 # make this as a class with "test" as a starting class name so Pytest is able to recognize it as a set to be run
32 def test_googles_search(driver, username, password):
33     # use and call the class function parameters (from the class source file) to use all of its methods...
34     login_page = LoginPage(driver)
35     time.sleep(5)
36     login_page.login_actions(url='https://trytestingthis.netlify.app', username, password)
37     time.sleep(5)
38     # display a confirmation of test result...
39     print("Test Chrome (login page) DONE!")
```

Test results...



```
Run Python tests in login_test.py x
Test Results 33 sec 297 ms
  login_test 33 sec 297 ms
    test_googles_search 33 sec 297 ms
      (test-test0) 10 sec 966 ms
      (wrong_usr1-wrong_pwd1) 5 sec 676 ms
      (wrong_usr2-wrong_pwd2) 5 sec 766 ms
      (test-test1) 10 sec 889 ms
Tests failed: 2, passed: 2 of 4 tests - 33 sec 297 ms
"C:\Users\njmlo\Desktop\Test Automation notes\Python_Selenium_PyCharm\SeleniumPythonPrj01\venv\Scripts\python.exe" -m pytest .\POM_files\tests\login_test_run.py --html=pom_result.rpt1.html
Testing started at 1:44 p.m. ...
Launching pytest with arguments C:\Users\njmlo\Desktop\Test Automation notes\Python_Selenium_PyCharm\SeleniumPythonPrj01\venv\Scripts\python.exe -m pytest .\POM_files\tests\login_test_run.py --html=pom_result.rpt1.html

===== test session starts =====
collecting ... collected 4 items

login_test.py::test_googles_search[test-test0]
login_test.py::test_googles_search[wrong_usr1-wrong_pwd1]
login_test.py::test_googles_search[wrong_usr2-wrong_pwd2]
login_test.py::test_googles_search[test-test1]

===== 2 failed, 2 passed in 70.63s (0:01:10) =====
```

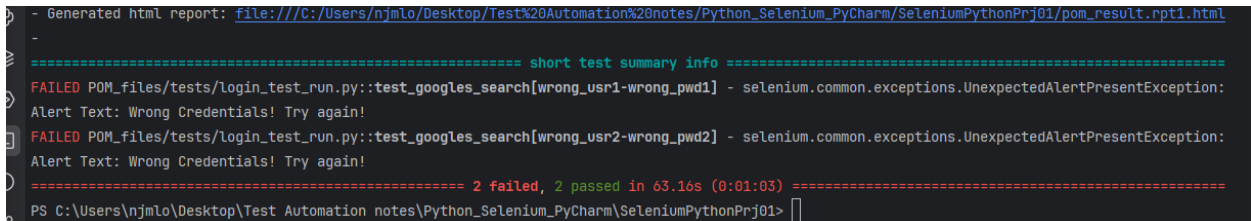
Test report...



```
PS C:\Users\njmlo\Desktop\Test Automation notes\Python_Selenium_PyCharm\SeleniumPythonPrj01> python -m pytest .\POM_files\tests\login_test_run.py --html=pom_result.rpt1.html
===== test session starts =====
platform win32 -- Python 3.12.6, pytest-8.3.3, pluggy-1.5.0
rootdir: C:\Users\njmlo\Desktop\Test Automation notes\Python_Selenium_PyCharm\SeleniumPythonPrj01
plugins: html-4.1.1, metadata-3.1.1
collected 4 items

POM_files\tests\login_test_run.py
DevTools listening on ws://127.0.0.1:50681/devtools/browser/87d38c08-0b86-4dd2-8091-2b7654183a8c
DevTools listening on ws://127.0.0.1:50718/devtools/browser/d001ec2c-f13c-4158-aa2b-76770f9958a8
F
DevTools listening on ws://127.0.0.1:50743/devtools/browser/17b2013b-0de2-4821-bdd5-b94496739c5e
F
DevTools listening on ws://127.0.0.1:50766/devtools/browser/a6089250-e016-4110-97c3-ff24a35ca34b

[100%]
```



```
- Generated html report: file:///C:/Users/njmlo/Desktop/Test%20Automation%20notes/Python_Selenium_PyCharm/SeleniumPythonPrj01/pom_result.rpt1.html
-
===== short test summary info =====
FAILED POM_files\tests\login_test_run.py::test_googles_search[wrong_usr1-wrong_pwd1] - selenium.common.exceptions.UnexpectedAlertPresentException:
Alert Text: Wrong Credentials! Try again!
FAILED POM_files\tests\login_test_run.py::test_googles_search[wrong_usr2-wrong_pwd2] - selenium.common.exceptions.UnexpectedAlertPresentException:
Alert Text: Wrong Credentials! Try again!
===== 2 failed, 2 passed in 63.16s (0:01:03) =====
PS C:\Users\njmlo\Desktop\Test Automation notes\Python_Selenium_PyCharm\SeleniumPythonPrj01>
```

pom_result.rpt1.html

C:/Users/njmlo/Desktop/Test%20Automation%20notes/Python_Selenium_PyCharm/SeleniumPythonPrj01/pom_result...

pom_result.rpt1.html

Report generated on 24-Nov-2024 at 14:12:20 by [pytest-html](#) v4.1.1

Environment

Python	3.12.6
Platform	Windows-11-10.0.26100-SP0
Packages	<ul style="list-style-type: none">• pytest 8.3.3• pluggy 1.5.0
Plugins	<ul style="list-style-type: none">• html 4.1.1• metadata 3.1.1

Summary

4 tests took 00:00:54.

(Un)check the boxes to filter the results.

☒ 2 Failed, ☒ 2 Passed, ☐ 0 Skipped, ☐ 0 Expected failures, ☐ 0 Unexpected passes, ☐ 0 Errors, ☒ 0 Reruns

Show all details / Hide all details

Result	Test	Duration	Links
Failed	POM_files/tests/login_test_run.py::test_googles_search[wrong_usr1-wrong_pwd1]	00:00:11	
<pre>driver = <selenium.webdriver.chrome.webdriver.WebDriver (session="b4cc9e5blac2044904f6e78737436935")>, username = 'wrong_usr1' password = 'wrong_pwd1' @pytest.mark.parametrize('username, password',[('test', 'test'), ('wrong_usr1', 'wrong_pwd1'), ('wrong_usr2', 'wrong_pwd2'), ('test', 'test'),]) # make this as a class with "test" as a starting class name so Pytest is able to recognize it as a set to be run def test_googles_search(driver, username, password): # use and call the class function parameters (from the class source file) to use all of its methods... login_page = LoginPage(driver) time.sleep(5) login_page.login_actions('https://trytestingthis.netlify.app', username, password)</pre>			
Failed	POM_files/tests/login_test_run.py::test_googles_search[wrong_usr2-wrong_pwd2]	00:00:11	
<pre>driver = <selenium.webdriver.chrome.webdriver.WebDriver (session="e5b2ab80763bc79e4ea0236db656ae7e")>, username = 'wrong_usr2' password = 'wrong_pwd2' @pytest.mark.parametrize('username, password',[('test', 'test'), ('wrong_usr1', 'wrong_pwd1'), ('wrong_usr2', 'wrong_pwd2'), ('test', 'test'),]) # make this as a class with "test" as a starting class name so Pytest is able to recognize it as a set to be run def test_googles_search(driver, username, password): # use and call the class function parameters (from the class source file) to use all of its methods... login_page = LoginPage(driver) time.sleep(5) login_page.login_actions('https://trytestingthis.netlify.app', username, password)</pre>			
Passed	POM_files/tests/login_test_run.py::test_googles_search[test-test0]	00:00:16	
Passed	POM_files/tests/login_test_run.py::test_googles_search[test-test1]	00:00:16	

*****END*****