

=====

Tutor: Testing World InfoTech

Reference: UDEMY

**Course: Step by Step Rest API Testing using Python + Pytest
+Allure**

Content: Testing Framework - PyTest

=====

1. Course URL: <https://www.udemy.com/course/api-testing-python/>
2. Document prepared by: **Rajat Verma**
 - a. <https://www.linkedin.com/in/rajat-v-3b0685128/>
 - b. <https://github.com/rajatt95>
 - c. <https://rajatt95.github.io/>

Softwares:

1. Programming language - **Python**
 - a. <https://www.python.org/downloads/>
2. IDE - **PyCharm**
 - a. <https://www.jetbrains.com/pycharm/download/#section=mac>
3. **PIP** (Package Installation Manager of Python)
 - a. To install packages
4. Package:
 - a. **Requests** (API testing)
 - b. **JSONPath** (To parse JSON)
 - c. **PyTest** (Testing framework)
 - d. **PyTest-HTML** (HTML Report)

1. Learnings from Course (UDEMY - TWI - API testing using Python-PyTest-Allure)

a. Links:

- i. The Testing World:
 1. <https://www.thetestingworld.com/>
- ii. JSON Path:
 1. <https://jsonpath.com/>
- iii. API testing:
 1. <https://reqres.in/>
 2. <https://httpbin.org/>

-
- iv. Help in resolving problems faced during the course:
 1. <https://stackoverflow.com/questions/35998992/py-test-command-not-found-but-library-is-installed>
 - v. PyTest:
 1. <https://docs.pytest.org/en/stable/how-to/mark.html>
 2. https://pytest-html.readthedocs.io/en/latest/user_guide.html
-

- b. PyTest:
 - i. **Unit Testing framework** for Python programming language
 - ii. **Execution of test cases (with file name):**
 1. `python3 -m pytest`
`PyTest_Basics/01_Prefix_Test_MustForPyTest.py`
 - a. **PyTest_Basics** is a *folder* name
 - b. **01_Prefix_Test_MustForPyTest.py** is the *file* name
 2. Method name should start with '`test_`'
 - a. It is required for **PyTest** to understand that this method is a Test case.
 - iii. **Execution of test cases (present inside a folder/directory):**
 1. `python3 -m pytest PyTest_Basics`
 - a. **PyTest_Basics** is a folder name
 - b. NOTE:
 - i. Foldername should also start with '`test_`'
 - iv. **Execution Using verbose (-v):**
 1. `python3 -m pytest -v PyTest_Basics`
 2. This will show the details
 - a. Which test case got passed/failed/skipped
 - v. **Execution Using (-s):**
 1. `python3 -m pytest -v -s PyTest_Basics`
 2. This will show the print statements in the console
 - vi. **Decorators:**
 1. To skip the test case:
 - a. `@pytest.mark.skip('Skipping this test case intentionally.....')`
 2. To skip the test case on the basis of condition:
 - a. `@pytest.mark.skipif(num1>99, reason='Skipping this test case intentionally.....')`

vii. Execute a specific test case:

1. Full name of test case

a. `python3 -m pytest -v -s -k test_login_API PyTest_Basics/`

i. `test_login_API` - test case name

ii. `PyTest_Basics/` - test case is present under this directory

2. Partial name of test case

a. `python3 -m pytest -v -s -k login_API PyTest_Basics/`

viii. TAGGING/GROUPING:

1. `python3 -m pytest -v -s -m Smoke PyTest_Basics`

a. `PyTest_Basics`- test cases are present under this directory

b. `Smoke` - Tagging name

c. NOTE:

i. These tag names (Smoke, Sanity) are User-defined.

2. Execution for Tagged test cases (condition):

a. `python3 -m pytest -v -s -m "not Regression" PyTest_Basics`

i. Execute the test cases which are not tagged as Regression

b. `python3 -m pytest -v -s -m "Smoke and Regression" PyTest_Basics`

i. Execute the test cases which are tagged as both Smoke and Regression

3. How to disable Warnings (getting on Console):

a. 2 Approaches:

i. Approach #1 -> `--disable-pytest-warnings`

1. `python3 -m pytest -v -s --disable-pytest-warnings -m "Smoke" PyTest_Basics`

ii. Approach #2 ->

1. Register the custom Marker (using `Pytest.ini` file)

4. PyTest Fixtures:

a. Test Level

- i. `@pytest.fixture()`
- ii. Before -> Test1 -> After
Before -> Test2 -> After
Before -> Test3 -> After
Before -> Test4 -> After

b. Module level

- i. `@pytest.fixture(scope="module")`
 - ii. Before -> Test1 -> Test2 -> Test3 -> Test4 -> After
-

c. Commands:

i. To execute the Python program

1. Right-click -> Run 'FILE_NAME'
2. Terminal ->

a. `python3 FILE_NAME.py`

ii. pip install requests

1. To install requests package using PIP
2. requests package is used for API automation

iii. `pip show requests` Not Working

`python3 -m pip show requests` - Working

1. To know the details about any installed package

iv. `pip install -U requests`

1. To upgrade any installed package

v. `pip install jsonpath`

1. To install requests package using PIP
2. jsonpath package is used for JSON parsing

vi. `pip install pytest`

1. To install pytest package using PIP
2. This package is Unit Testing framework for Python programming language

vii. `pip install pytest-html`

1. To install `pytest-html` package using PIP

viii. `python3 -m pytest -v -s PyTest_Basics --html=report.html`

1. Execute the test cases
-
-

=====Section 12: PyTest : Unit Testing Framework for Python=====

-----85. PyTest Introduction -----

1. PyTest is one of the ***Unit Testing frameworks*** for the Python programming language
2. **Why PyTest?**
 - a. Options to conditionally execute the test cases
 - b. Setup for Pre-Requisite and Post Script
 - i. Before and After the test execution,
 1. If you want to get an operation done, we can do using these
 - c. Assertions
 - i. Actual result vs Expected result
 - d. Report Generation
3. Other Unit testing frameworks for Python are:
 - a. UnitTest
 - b. Nose

-----86. Pytest Installation: pip install pytest -----

1. **pip install pytest**

-----87. Write Test Case in Pytest format -----

1. Method name should start with '**test_**'
 - a. It is required for **PyTest** to understand that this method is a Test case.
 - b. There is no annotation like
 - i. @Test -> We have for TestNG framework (when we work with Java programming language)

```
14     # https://stackoverflow.com/questions/35998992/py-test-command-not-found
15
16     # PyTest will not consider this method as a Test case
17     # Because method name is not starting with a prefix as 'test'
18     def tc_login_UI():
19         print('This is tc_login_UI.....')
20
21
22     def test_login_API():
23         print('This is test_login_API.....')
```

2.

- We have a green icon with the method (test_login_API)

3. Test execution using Terminal:

```
Terminal: Local + ▾
rajasverma@rajat-MacBook-Air _03_PyTest_TWI % python3 -m pytest PyTest_Basics/01_Prefix_Test_MustForPyTest.py
=====
platform darwin -- Python 3.10.5, pytest-7.1.2, pluggy-1.0.0
rootdir: /Users/rajasverma/Desktop/Work/Local_PyCharm_WS/UDEMY_TWI_APITesting_Python/_03_PyTest_TWI
collected 1 item

PyTest_Basics/01_Prefix_Test_MustForPyTest.py .

=====
1 passed in 0.00s
rajasverma@rajat-MacBook-Air _03_PyTest_TWI %
```

- i. **python3 -m pytest PyTest_Basics/01_Prefix_Test_MustForPyTest.py**

-----88. Write Multiple Test Cases and Execute test cases together -----

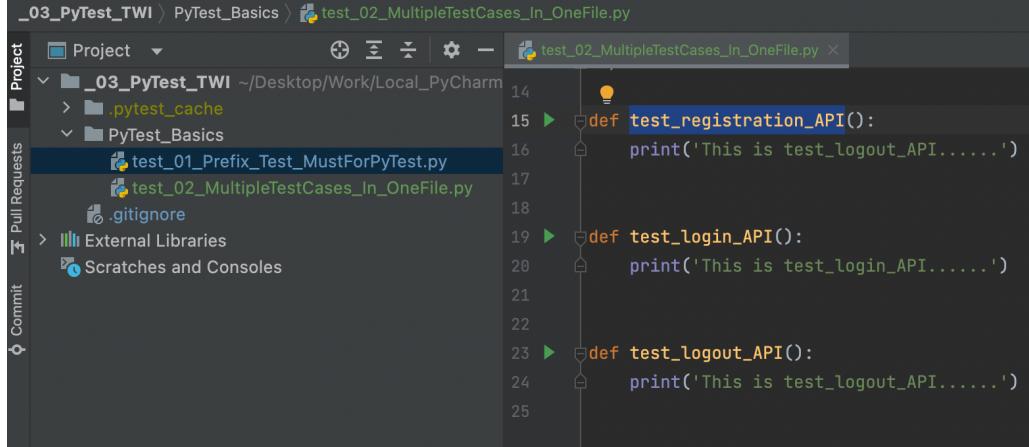
1. Execution of test cases (present inside a folder/directory):

a. `python3 -m pytest PyTest_Basics`

i. PyTest_Basics is a folder name

ii. NOTE:

1. Foldername should also start with '`test_`'



b.

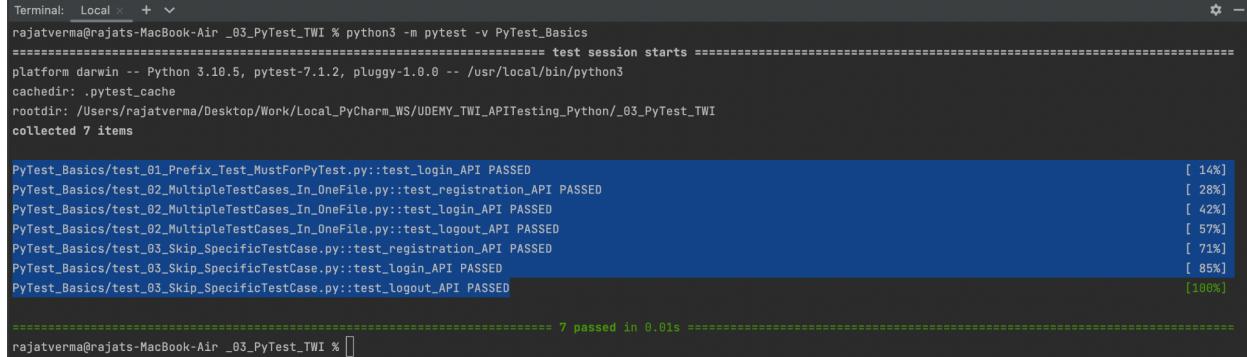
-----89. Write Multiple Test Cases in a File: Check Execution Options -----

1. Execution Using verbose (-v):

a. `python3 -m pytest -v PyTest_Basics`

b. This will show the details

i. Which test case got passed/failed/skipped



C.

```
PyTest_Basics/test_01_Prefix_Test_MustForPyTest.py::test_login_API PASSED
PyTest_Basics/test_02_MultipleTestCases_In_OneFile.py::test_registration_API PASSED
PyTest_Basics/test_02_MultipleTestCases_In_OneFile.py::test_login_API PASSED
PyTest_Basics/test_02_MultipleTestCases_In_OneFile.py::test_logout_API PASSED
PyTest_Basics/test_03_Skip_SpecificTestCase.py::test_registration_API PASSED
PyTest_Basics/test_03_Skip_SpecificTestCase.py::test_login_API PASSED
PyTest_Basics/test_03_Skip_SpecificTestCase.py::test_logout_API PASSED

=====
d. ===== 7 passed in 0.01s
```

1. Execution Using (-s):

a. **python3 -m pytest -v -s PyTest_Basics**

b. This will show the print statements in console

```
Terminal: Local + ▾
rajatverma@rajats-MacBook-Air _03_PyTest_TWI % python3 -m pytest -v -s PyTest_Basics

=====
platform darwin -- Python 3.10.5, pytest-7.1.2, pluggy-1.0.0 -- /usr/local/bin/python3
cachedir: .pytest_cache
rootdir: /Users/rajatverma/Desktop/Work/Local_PyCharm_WS/UDEMY_TWI_APITesting_Python/_03_PyTest_TWI
collected 7 items

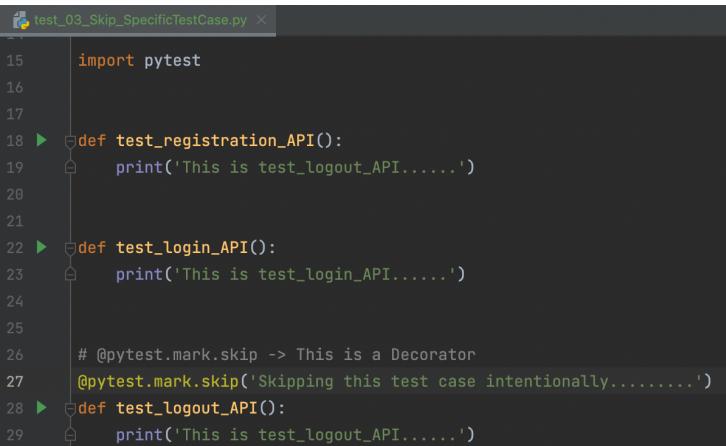
PyTest_Basics/test_01_Prefix_Test_MustForPyTest.py::test_login_API This is test_login_API.....
PASSED
PyTest_Basics/test_02_MultipleTestCases_In_OneFile.py::test_registration_API This is test_logout_API.....
PASSED
PyTest_Basics/test_02_MultipleTestCases_In_OneFile.py::test_login_API This is test_login_API.....
PASSED
PyTest_Basics/test_02_MultipleTestCases_In_OneFile.py::test_logout_API This is test_logout_API.....
PASSED
PyTest_Basics/test_03_Skip_SpecificTestCase.py::test_registration_API This is test_logout_API.....
PASSED
PyTest_Basics/test_03_Skip_SpecificTestCase.py::test_login_API This is test_login_API.....
PASSED
PyTest_Basics/test_03_Skip_SpecificTestCase.py::test_logout_API This is test_logout_API.....
PASSED

=====
rajatverma@rajats-MacBook-Air _03_PyTest_TWI %
```

-----90. Skip Test Cases | Execute test cases conditionally -----

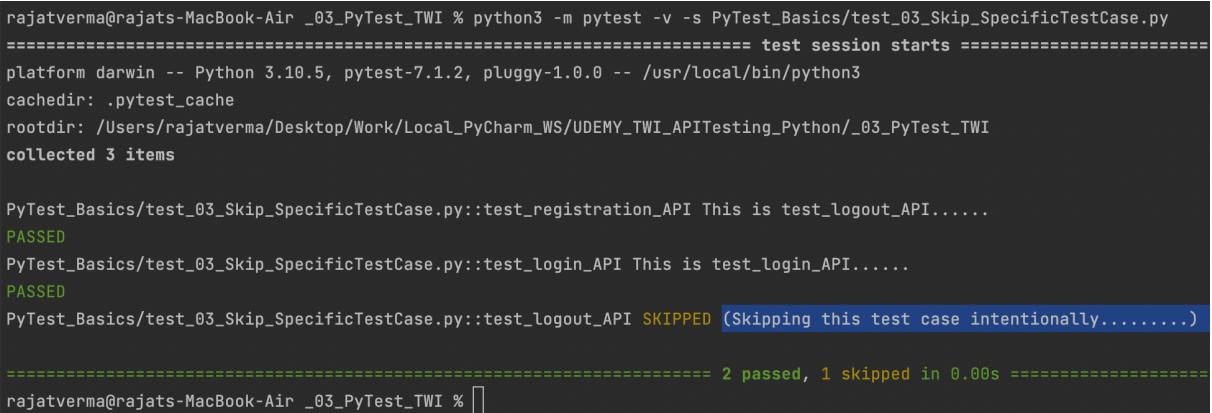
1. To skip a Test case, we have to use the Decorator - **skip**:

a. `@pytest.mark.skip('Skipping this test case intentionally.....')`



```
test_03_Skip_SpecificTestCase.py
15     import pytest
16
17
18     def test_registration_API():
19         print('This is test_logout_API.....')
20
21
22     def test_login_API():
23         print('This is test_login_API.....')
24
25
26     # @pytest.mark.skip -> This is a Decorator
27     @pytest.mark.skip('Skipping this test case intentionally.....')
28
29     def test_logout_API():
30         print('This is test_logout_API.....')
```

b.



```
rajabverma@rajats-MacBook-Air _03_PyTest_TWI % python3 -m pytest -v -s PyTest_Basics/test_03_Skip_SpecificTestCase.py
=====
platform darwin -- Python 3.10.5, pytest-7.1.2, pluggy-1.0.0 -- /usr/local/bin/python3
cachedir: .pytest_cache
rootdir: /Users/rajabverma/Desktop/Work/Local_PyCharm_WS/UDEMY_TWI_APITesting_Python/_03_PyTest_TWI
collected 3 items

PyTest_Basics/test_03_Skip_SpecificTestCase.py::test_registration_API This is test_logout_API.....  
PASSED
PyTest_Basics/test_03_Skip_SpecificTestCase.py::test_login_API This is test_login_API.....  
PASSED
PyTest_Basics/test_03_Skip_SpecificTestCase.py::test_logout_API SKIPPED (Skipping this test case intentionally.....)

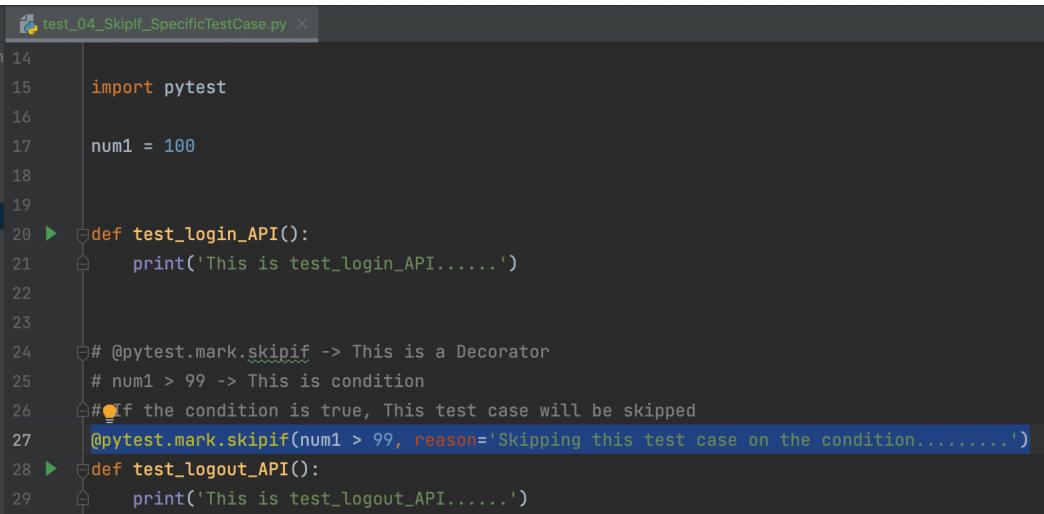
=====
rajabverma@rajats-MacBook-Air _03_PyTest_TWI %
```

c.

2. To skip a Test case, we have to use the Decorator - **skipif**:

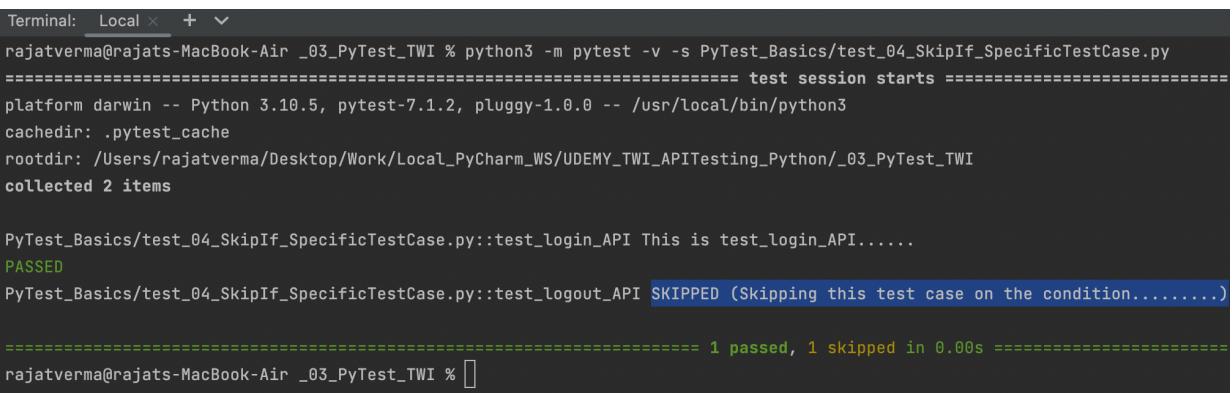
- a. `@pytest.mark.skipif(num1>99, reason='Skipping this test case intentionally.....')`

b.



```
test_04_SkipIf_SpecificTestCase.py
14
15     import pytest
16
17     num1 = 100
18
19
20 ► def test_login_API():
21     print('This is test_login_API.....')
22
23
24     # @pytest.mark.skipif -> This is a Decorator
25     # num1 > 99 -> This is condition
26     # If the condition is true, This test case will be skipped
27     @pytest.mark.skipif(num1 > 99, reason='Skipping this test case on the condition.....')
28 ► def test_logout_API():
29     print('This is test_logout_API.....')
```

c.



```
Terminal: Local + ▾
rajatverma@rajats-MacBook-Air _03_PyTest_TWI % python3 -m pytest -v -s PyTest_Basics/test_04_SkipIf_SpecificTestCase.py
=====
test session starts =====
platform darwin -- Python 3.10.5, pytest-7.1.2, pluggy-1.0.0 -- /usr/local/bin/python
cachedir: .pytest_cache
rootdir: /Users/rajatverma/Desktop/Work/Local_PyCharm_WS/UDEMY_TWI_APITesting_Python/_03_PyTest_TWI
collected 2 items

PyTest_Basics/test_04_SkipIf_SpecificTestCase.py::test_login_API This is test_login_API.....  
PASSED
PyTest_Basics/test_04_SkipIf_SpecificTestCase.py::test_logout_API SKIPPED (Skipping this test case on the condition.....)

===== 1 passed, 1 skipped in 0.00s =====
rajatverma@rajats-MacBook-Air _03_PyTest_TWI %
```

3. Execute a specific test case:

a. Full name of test case

i. `python3 -m pytest -v -s -k test_login_API PyTest_Basics/`

1. `test_login_API` - test case name
2. `PyTest_Basics/` - test case is present under this directory

```
Terminal: Local × + ▾
rajabverma@rajats-MacBook-Air _03_PyTest_TWI % python3 -m pytest -v -s -k test_login_API PyTest_Basics/
=====
platform darwin -- Python 3.10.5, pytest-7.1.2, pluggy-1.0.0 -- /usr/local/bin/python3
cachedir: .pytest_cache
rootdir: /Users/rajabverma/Desktop/Work/Local_PyCharm_WS/UDEMY_TWI_APITesting_Python/_03_PyTest_TWI
collected 9 items / 5 deselected / 4 selected

PyTest_Basics/test_01_Prefix_Test_MustForPyTest.py::test_login_API This is test_login_API.....
PASSED
PyTest_Basics/test_02_MultipleTestCases_In_OneFile.py::test_login_API This is test_login_API.....
PASSED
PyTest_Basics/test_03_Decorator_skip_SpecificTestCase.py::test_login_API This is test_login_API.....
PASSED
PyTest_Basics/test_04_Decorator_skipIf_SpecificTestCase.py::test_login_API This is test_login_API.....
PASSED

=====
4 passed, 5 deselected in 0.00s =
```

3.

b. Partial name of test case

i. `python3 -m pytest -v -s -k login_API PyTest_Basics/`

```
Terminal: Local × + ▾
rajabverma@rajats-MacBook-Air _03_PyTest_TWI % python3 -m pytest -v -s -k login_API PyTest_Basics/
=====
platform darwin -- Python 3.10.5, pytest-7.1.2, pluggy-1.0.0 -- /usr/local/bin/python3
cachedir: .pytest_cache
rootdir: /Users/rajabverma/Desktop/Work/Local_PyCharm_WS/UDEMY_TWI_APITesting_Python/_03_PyTest_TWI
collected 9 items / 5 deselected / 4 selected

PyTest_Basics/test_01_Prefix_Test_MustForPyTest.py::test_login_API This is test_login_API.....
PASSED
PyTest_Basics/test_02_MultipleTestCases_In_OneFile.py::test_login_API This is test_login_API.....
PASSED
PyTest_Basics/test_03_Decorator_skip_SpecificTestCase.py::test_login_API This is test_login_API.....
PASSED
PyTest_Basics/test_04_Decorator_skipIf_SpecificTestCase.py::test_login_API This is test_login_API.....
PASSED

=====
4 passed, 5 deselected in 0.01s
rajabverma@rajats-MacBook-Air _03_PyTest_TWI %
```

ii.

-----91. Grouping Test Cases: Tagging and Execute using tags -----

1. Tagging/Grouping:

a. Example:

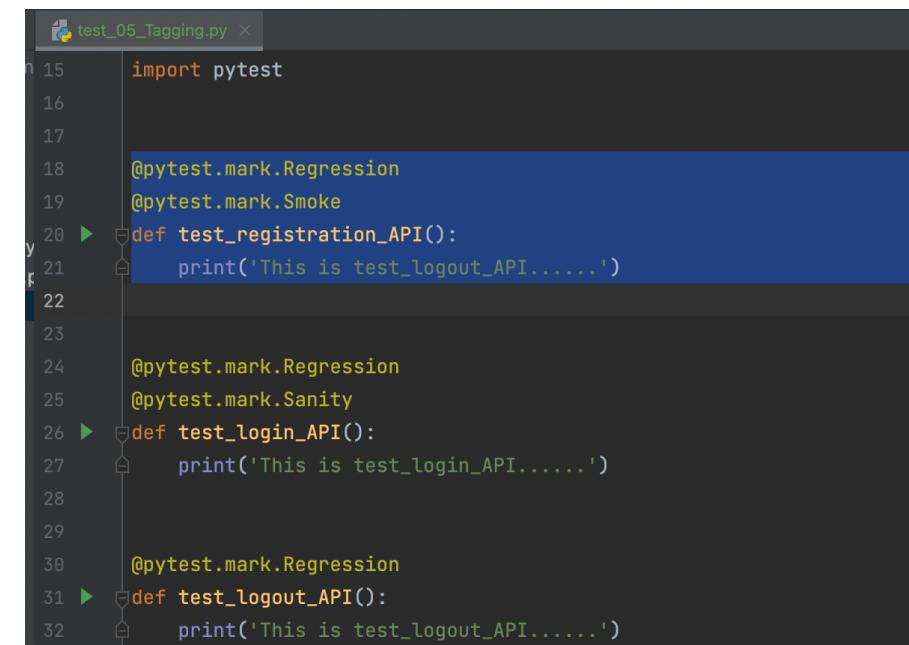
i. We have 500 test cases

1. Now, we only want to execute 100 test cases which covers critical functionalities of the application
 - a. Those can be tagged/grouped as **SMOKE**.

ii. With PyTest,

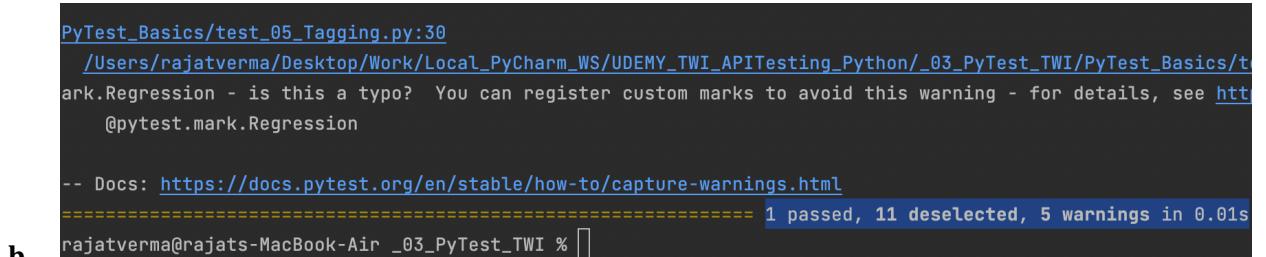
1. We can tag/group the test cases and execute accordingly.

2. Execution for Tagged test cases:



```
test_05_Tagging.py
15 import pytest
16
17
18 @pytest.mark.Regression
19 @pytest.mark.Smoke
20 def test_registration_API():
21     print('This is test_logout_API.....')
22
23
24 @pytest.mark.Regression
25 @pytest.mark.Sanity
26 def test_login_API():
27     print('This is test_login_API.....')
28
29
30 @pytest.mark.Regression
31 def test_logout_API():
32     print('This is test_logout_API.....')
```

a.



```
PyTest_Basics/test_05_Tagging.py:30
/Users/rajatverma/Desktop/Work/Local_PyCharm_WS/UDEMY_TWI_APITesting_Python/_03_PyTest_TWI/PyTest_Basics/test_05_Tagging.py:30: Warning: mark.Regression - is this a typo? You can register custom marks to avoid this warning - for details, see https://docs.pytest.org/en/stable/how-to/capture-warnings.html
@ pytest.mark.Regression

-- Docs: https://docs.pytest.org/en/stable/how-to/capture-warnings.html
=====
1 passed, 11 deselected, 5 warnings in 0.01s
rajamohanreddy@rajamohanreddy-MacBook-Air ~ %
```

b.

i. **python3 -m pytest -v -s -m Smoke PyTest_Basics**

1. **PyTest_Basics**- test cases are present under this directory
2. **Smoke** - Tagging name

ii. **NOTE:**

1. These tag names (Smoke, Sanity) are User-defined.

```

import pytest

# UserDefined_TopPriority is a custom mark
@ pytest.mark.Regression
@ pytest.mark.Smoke
@ pytest.mark.UserDefined_TopPriority
def test_registration_API():
    print('This is test_logout_API.....')

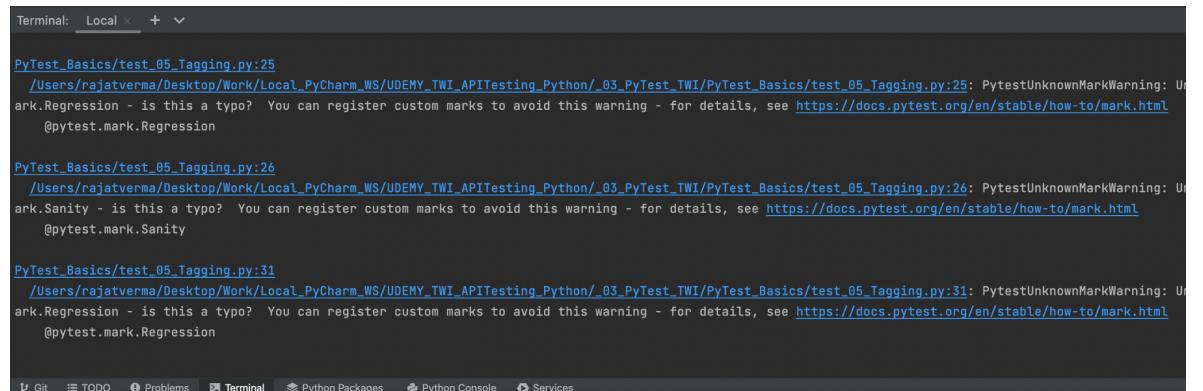
```

2.

1. Execution for Tagged test cases (condition):

- a. **python3 -m pytest -v -s -m "not Regression" PyTest_Basics**
 - i. Execute the test cases which are not tagged as Regression
 - b. **python3 -m pytest -v -s -m "Smoke and Regression" PyTest_Basics**
 - i. Execute the test cases which are tagged as both Smoke and Regression
-

1. How to disable Warnings (getting on Console):



The terminal window shows three warning messages from the PytestUnknownMarkWarning class:

- PyTest_Basics/test_05_Tagging.py:25: PytestUnknownMarkWarning: Unknown mark 'Regression' - is this a typo? You can register custom marks to avoid this warning - for details, see <https://docs.pytest.org/en/stable/how-to/mark.html>
- PyTest_Basics/test_05_Tagging.py:26: PytestUnknownMarkWarning: Unknown mark 'Sanity' - is this a typo? You can register custom marks to avoid this warning - for details, see <https://docs.pytest.org/en/stable/how-to/mark.html>
- PyTest_Basics/test_05_Tagging.py:31: PytestUnknownMarkWarning: Unknown mark 'UserDefined_TopPriority' - is this a typo? You can register custom marks to avoid this warning - for details, see <https://docs.pytest.org/en/stable/how-to/mark.html>

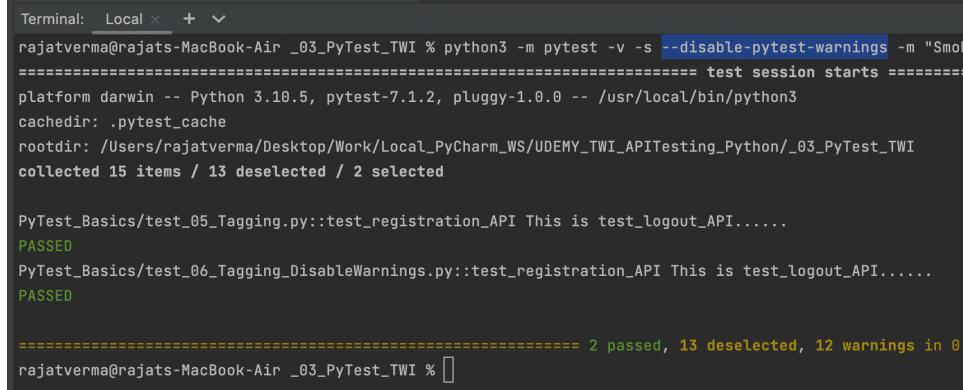
a. Git TODO Problems Terminal Python Packages Python Console Services

- b. We are getting these warnings because
 - i. We are using User-Defined Tags (Smoke/Sanity/Regression)
-

c. 2 Approaches:

i. Approach #1 -> --disable-pytest-warnings

1. `python3 -m pytest -v -s --disable-pytest-warnings -m "Smoke" PyTest_Basics`



```
Terminal: Local + ▾
rajasverma@rajats-MacBook-Air _03_PyTest_TWI % python3 -m pytest -v -s --disable-pytest-warnings -m "Smoke"
=====
platform darwin -- Python 3.10.5, pytest-7.1.2, pluggy-1.0.0 -- /usr/local/bin/python3
cachedir: .pytest_cache
rootdir: /Users/rajasverma/Desktop/Work/Local_PyCharm_WS/UDEMY_TWI_APITesting_Python/_03_PyTest_TWI
collected 15 items / 13 deselected / 2 selected

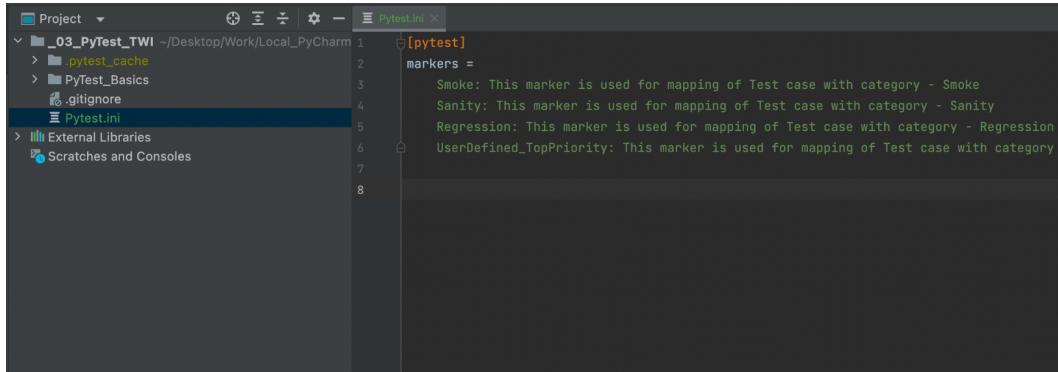
PyTest_Basics/test_05_Tagging.py::test_registration_API This is test_logout_API..... PASSED
PyTest_Basics/test_06_Tagging_DisableWarnings.py::test_registration_API This is test_logout_API..... PASSED

=====
2 passed, 13 deselected, 12 warnings in 0.00s
rajasverma@rajats-MacBook-Air _03_PyTest_TWI %
```

a.

ii. Approach #2 ->

1. Register the custom Marker (using `Pytest.ini` file)

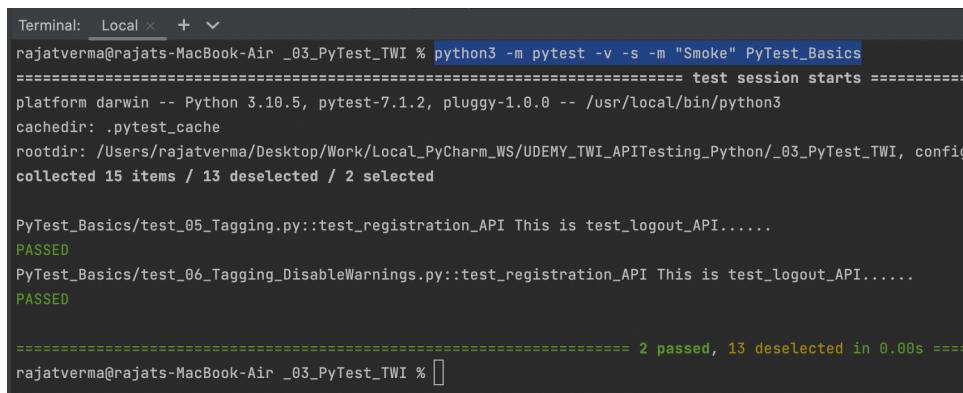


2.

3. Now, Terminal:

a. `python3 -m pytest -v -s -m "Smoke" PyTest_Basics`

i. No warnings will be there



```
Terminal: Local + ▾
rajasverma@rajats-MacBook-Air _03_PyTest_TWI % python3 -m pytest -v -s -m "Smoke" PyTest_Basics
=====
platform darwin -- Python 3.10.5, pytest-7.1.2, pluggy-1.0.0 -- /usr/local/bin/python3
cachedir: .pytest_cache
rootdir: /Users/rajasverma/Desktop/Work/Local_PyCharm_WS/UDEMY_TWI_APITesting_Python/_03_PyTest_TWI, config file: /Users/rajasverma/Desktop/Work/Local_PyCharm_WS/UDEMY_TWI_APITesting_Python/_03_PyTest_TWI/.pytest_cache/pyconfig.ini
collected 15 items / 13 deselected / 2 selected

PyTest_Basics/test_05_Tagging.py::test_registration_API This is test_logout_API..... PASSED
PyTest_Basics/test_06_Tagging_DisableWarnings.py::test_registration_API This is test_logout_API..... PASSED

=====
2 passed, 13 deselected in 0.00s
rajasverma@rajats-MacBook-Air _03_PyTest_TWI %
```

b.

-----92. Assertions: Compare Actual Result with Expected Result -----

1. Assertions:

- a. Compare actual results with expected results
 - i. Data to be the same
 - ii. Data not to be same
 - iii. Compare data and customized message in case of failure

2. Assertions:

a.

```
test_07_Assertions.py
14 import pytest
15
16
17 def test_registration_API():
18     print('This is test_registration_API.....')
19     result_actual = "Hello"
20     result_expected = "Hello !"
21     # Should be equal
22     assert result_actual == result_expected, "Message - Actual results are not matching with Expected results" #
23     # assert result_actual != result_expected # Should not be equal
24
25
26 def test_login_API():
27     print('This is test_login_API.....')
28     result_actual = "Hello"
29     result_expected = "Hello"
30     assert result_actual == result_expected
31
32
33 @pytest.mark.skip('Skipping this test case intentionally.....')
34 def test_logout_API():
35     print('This is test_logout_API.....')
```

b.

```
Terminal: Local × + ▾
E     assert 'Hello' == 'Hello !'
E     - Hello !
E     ?
E     + Hello

PyTest_Basics/test_07_Assertions.py:22: AssertionError
=====
short test summary info =====
FAILED PyTest_Basics/test_07_Assertions.py::test_registration_API - AssertionError: Message - Actual results are not matching with Expected results
=====
1 failed, 1 passed, 1 skipped in 0.03s =====
rajatverma@rajats-MacBook-Air ~_03_PyTest_TWI %
```

-----93. Fixtures : Execute Code Before and After Test Case -----

1. PyTest Fixtures:

a. Levels:

i. Test case Level

1. `@pytest.fixture()`

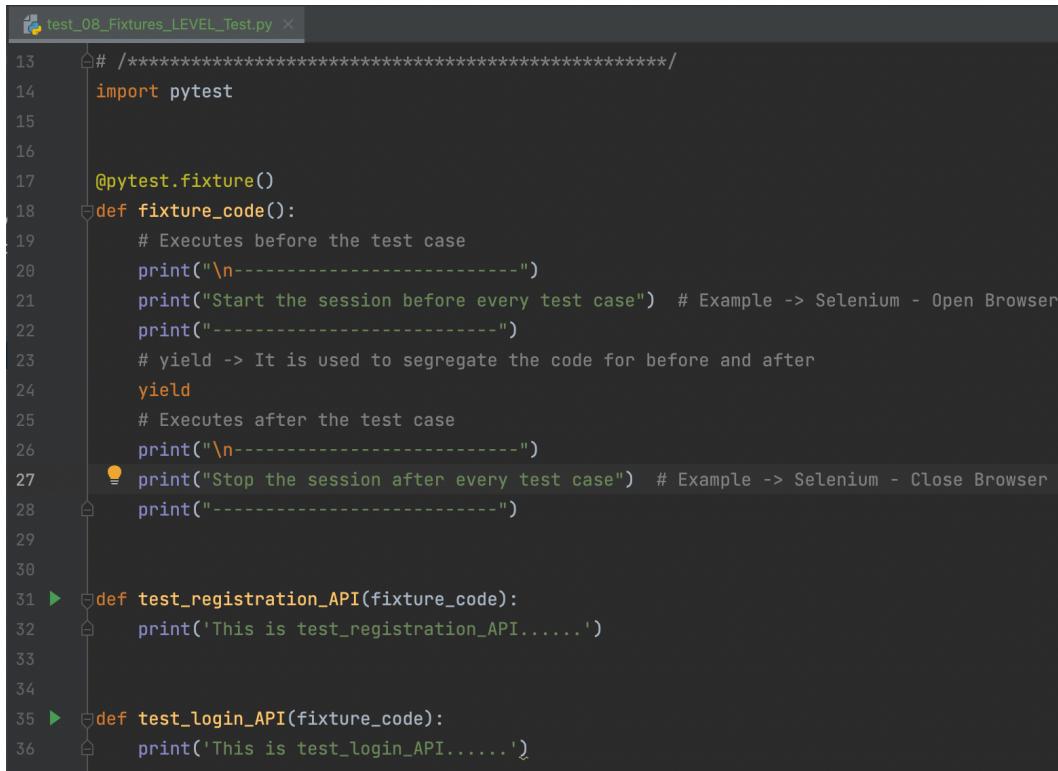
2. Execute something before and after each test case

3. Before -> Test 1 -> After

Before -> Test 2 -> After

Before -> Test 3 -> After

Before -> Test 4 -> After



The screenshot shows a code editor with a file named `test_08_Fixtures_LEVEL_Test.py`. The code defines a fixture named `fixture_code` that prints session start and stop messages. It then defines two test functions, `test_registration_API` and `test_login_API`, which both print their respective names.

```
# /*****  
13     import pytest  
14  
15  
16  
17     @pytest.fixture()  
18     def fixture_code():  
19         # Executes before the test case  
20         print("\n-----")  
21         print("Start the session before every test case") # Example -> Selenium - Open Browser  
22         print("-----")  
23         # yield -> It is used to segregate the code for before and after  
24         yield  
25         # Executes after the test case  
26         print("\n-----")  
27         print("Stop the session after every test case") # Example -> Selenium - Close Browser  
28         print("-----")  
29  
30  
31     def test_registration_API(fixture_code):  
32         print('This is test_registration_API.....')  
33  
34  
35     def test_login_API(fixture_code):  
36         print('This is test_login_API.....')
```

ii. Module-level

1. Module is 1 python file

2. Example:

a. You have 5 test cases inside an python file and you want fixture code to execute

i. Before -> Test1 -> Test2 -> Test3 -> Test4 -> After

3. `@pytest.fixture(scope="module")`

```

13  # ****
14  import pytest
15
16  @pytest.fixture(scope="module")
17  def fixture_code():
18      # Executes before the test case
19      print("\n-----")
20      print("Start the session before 1st test case") # Example -> Start DB connection
21      print("-----")
22      # yield -> It is used to segregate the code for before and after
23      yield
24      # Executes after the test case
25      print("\n-----")
26      print("Stop the session after last test case") # Example -> Stop DB connection
27      print("-----")
28
29
30
31 > def test_registration_API(fixture_code):
32     print('This is test_registration_API.....')
33
34
35 > def test_login_API(fixture_code):
36     print('This is test_login_API.....')

```

4.

-----94. Reporting in Pytest -----

1. Default Reporting:

- a. No extra plugins are required
- b. This is an advanced HTML report
- c. High-level execution status in tabular format
- d. Display Graphical trend by comparing execution status with the previous build
- e. Low level information at the Test case level
- f. Clear information for the reason of failure

2. **pip install pytest-html**

- a. To install **pytest-html** package using PIP

3. Execute the test cases

a. python3 -m pytest -v -s PyTest_Basics --html=report.html

4. Once we execute the tests,

- a. At the project level, we should see a report generated with the name: **report.html**
- b. Report:

report.html																										
Report generated on 29-Jun-2022 at 15:47:22 by pytest-html v3.1.1																										
Environment																										
<table border="1"><tr><td>JAVA_HOME</td><td>/Library/Java/JavaVirtualMachines/jdk-11.0.13.jdk/Contents/Home</td></tr><tr><td>Packages</td><td>{"pluggy": "1.0.0", "py": "1.11.0", "pytest": "7.1.2"}</td></tr><tr><td>Platform</td><td>macOS-12.4-arm64-arm-64bit</td></tr><tr><td>Plugins</td><td>{"html": "3.1.1", "metadata": "2.0.1"}</td></tr><tr><td>Python</td><td>3.10.5</td></tr></table>			JAVA_HOME	/Library/Java/JavaVirtualMachines/jdk-11.0.13.jdk/Contents/Home	Packages	{"pluggy": "1.0.0", "py": "1.11.0", "pytest": "7.1.2"}	Platform	macOS-12.4-arm64-arm-64bit	Plugins	{"html": "3.1.1", "metadata": "2.0.1"}	Python	3.10.5														
JAVA_HOME	/Library/Java/JavaVirtualMachines/jdk-11.0.13.jdk/Contents/Home																									
Packages	{"pluggy": "1.0.0", "py": "1.11.0", "pytest": "7.1.2"}																									
Platform	macOS-12.4-arm64-arm-64bit																									
Plugins	{"html": "3.1.1", "metadata": "2.0.1"}																									
Python	3.10.5																									
Summary																										
19 tests ran in 0.03 seconds.																										
(Un)check the boxes to filter the results.																										
<input type="checkbox"/> 18 passed, <input checked="" type="checkbox"/> 3 skipped, <input checked="" type="checkbox"/> 1 failed, <input type="checkbox"/> 0 errors, <input type="checkbox"/> 0 expected failures, <input type="checkbox"/> 0 unexpected passes																										
Results																										
<a>Show all details / Hide all details																										
<table border="1"><thead><tr><th>Result</th><th>Test</th><th>Duration</th></tr></thead><tbody><tr><td>Failed (hide details)</td><td>PyTest_Basics/test_07_Assertions.py::test_registration_API</td><td>0.00</td></tr><tr><td>No log output captured.</td><td></td><td></td></tr><tr><td>Skipped (hide details)</td><td>PyTest_Basics/test_03_Decorator_skip_SpecificTestCase.py::test_logout_API</td><td>6.06</td></tr><tr><td>No log output captured.</td><td></td><td></td></tr><tr><td>Skipped (hide details)</td><td>PyTest_Basics/test_04_Decorator_skipif_SpecificTestCase.py::test_logout_API</td><td>4.84</td></tr><tr><td>No log output captured.</td><td></td><td></td></tr><tr><td>Skipped (hide details)</td><td>PyTest_Basics/test_07_Assertions.py::test_logout_API</td><td>4.85</td></tr></tbody></table>			Result	Test	Duration	Failed (hide details)	PyTest_Basics/test_07_Assertions.py::test_registration_API	0.00	No log output captured.			Skipped (hide details)	PyTest_Basics/test_03_Decorator_skip_SpecificTestCase.py::test_logout_API	6.06	No log output captured.			Skipped (hide details)	PyTest_Basics/test_04_Decorator_skipif_SpecificTestCase.py::test_logout_API	4.84	No log output captured.			Skipped (hide details)	PyTest_Basics/test_07_Assertions.py::test_logout_API	4.85
Result	Test	Duration																								
Failed (hide details)	PyTest_Basics/test_07_Assertions.py::test_registration_API	0.00																								
No log output captured.																										
Skipped (hide details)	PyTest_Basics/test_03_Decorator_skip_SpecificTestCase.py::test_logout_API	6.06																								
No log output captured.																										
Skipped (hide details)	PyTest_Basics/test_04_Decorator_skipif_SpecificTestCase.py::test_logout_API	4.84																								
No log output captured.																										
Skipped (hide details)	PyTest_Basics/test_07_Assertions.py::test_logout_API	4.85																								

i.

1. https://pytest-html.readthedocs.io/en/latest/user_guide.html

1. To connect:

- a. <https://www.linkedin.com/in/rajat-v-3b0685128/>
- b. <https://github.com/rajatt95>
- c. <https://rajatt95.github.io/>

THANK YOU!

