Python UNIT Testing Study Material

DURGASOFT, # 202, 2nd Floor, HUDA Maitrivanam, Ameerpet, Hyderabad - 500038,
☎ 88 85 25 26 27, 72 07 21 24 27/28 | www.durgasoftonline.com
Maii: durgasoftonline@gmail.com

Python UNIT Testing

There are mainly 2 types of testings.

1) **UNIT Testing**
2) **Integration Testing**

1) UNIT Testing:

- **The process of testing whether a particular unit is working properly or not, is called UNIT Testing.**
- **Developer is responsible to perform unit testing.**

2) Integration Testing:

- **The process of testing total application (end-to-end testing) whether it is working properly or not, is called Integration Testing.**
- **Separate QA Team (Testing Team) is responsible to perform Integration Testing.**

Test Scenario vs TestCase vs Test Suite:

Test Scenario (What to test?):

Test scenario describes what we have to test.
It describes test condition or requirment or functionality which need to validate.
Test Scenario also known as Test Condition or Test possibility.

Eg 1: Identify various Test scenarios in google home page

TS01: Checking functionality of Google Search Button
TS02: Checking functionality of Sign in Button
TS03: Checking functionality of Gmail hyperlink
TS04: Checking functionality of images hyperlink
etc

Eg 2: Test Scenarios for Banking application

TS01: Check the login functionality
TS02: Check Transfer funds functionality
TS03: Check account statement functionality
etc

DURGASOFT, # 202, 2nd Floor, HUDA Maitrivanam, Ameerpet, Hyderabad - 500038,
☎ 88 85 25 26 27, 72 07 21 24 27/28 | www.durgasoftonline.com
Maii: durgasoftonline@gmail.com

Test Case (How to Test?):

After identifying Test Scenarios, we have to prepare test cases.

A Test case is a set of actions executed to verify a particular feature or functionality of our application.

Eg: Identify all possible test cases for Test Scenario: Login validation

TC01: Verify login functionity with valid data
TC02: Verify login functionity with invalid data
TC03: Verify login functionity without any data

Test Suite:

It contains a group of test cases. The main advantage of Test Suite is once we execute test suite automatically all test cases will be executed and we are not required to execute test cases individually.

How to perform UNIT Testing in Python:

We have to use unittest module. It is Python's inbuilt module. This module contains TestCase class. We have to create child class for this TestCase class.

In the child class we have to override the following methods.

1) **setUp():**
   This method will be executed before every test

2) **test():**
   In this method we have to define the logic for our required testing. The method name can be anything, but should be prefixed with test.
   Eg: test(), test1(), test2(), test_method1(), test_method2() etc

3) **tearDown():**
   This method will be executed after every test.
   Usually we have to write cleanup activities like closing the browser in this method.

DURGASOFT, # 202, 2nd Floor, HUDA Maitrivanam, Ameerpet, Hyderabad - 500038,
☎ 88 85 25 26 27, 72 07 21 24 27/28 | www.durgasoftonline.com
Maii: durgasoftonline@gmail.com

unittest Class Level Methods:

unittest framework contains the following 2 class level methods.

1) setUpClass():

This method will be executed only once before all test methods execution.

We have to declare this method with @classmethod decorator as follows

```
@classmethod
 def setUpClass(cls):
      body
```

2) tearDownClass():

This method will be executed only once after all test methods execution.
We have to declare this method with @classmethod decorator as follows

```
@classmethod
 def tearDownClass(cls):
   body
```

Demo Application-1:

```
1)  import unittest
2)  class TestCaseDemo(unittest.TestCase):
3)      def setUp(self):
4)          print('setUp method execution')
5)      def test(self):
6)          print('test method execution')
7)      def tearDown(self):
8)          print('tearDown method execution')
9)
10) unittest.main()
```

Output

```
D:\durgaclasses>py test.py
setUp method execution
test method execution
tearDown method execution
.
----------------------------------------------------------------------
Ran 1 test in 0.016s

OK
```


DURGASOFT, # 202, 2nd Floor, HUDA Maitrivanam, Ameerpet, Hyderabad - 500038,
☎ 88 85 25 26 27, 72 07 21 24 27/28 | www.durgasoftonline.com
Maii: durgasoftonline@gmail.com

Demo Application-2:

```
1)  import unittest
2)  class TestCaseDemo(unittest.TestCase):
3)      def setUp(self):
4)          print('setUp method execution')
5)      def test(self):
6)          print('test method execution')
7)          print(10/0)
8)      def tearDown(self):
9)          print('tearDown method execution')
10)
11) unittest.main()
```

Output

```
D:\durgaclasses>py test.py
setUp method execution
test method execution
tearDown method execution
E
======================================================================
ERROR: test (__main__.TestCaseDemo)
----------------------------------------------------------------------
Traceback (most recent call last):
  File "test.py", line 7, in test
    print(10/0)
ZeroDivisionError: division by zero
----------------------------------------------------------------------
Ran 1 test in 0.000s

FAILED (errors=1)
```

Demo Application-3:

```
1)  import unittest
2)  class TestCaseDemo(unittest.TestCase):
3)      def setUp(self):
4)          print('setUp method execution')
5)      def test_method1(self):
6)          print('test method1 execution')
7)
8)      def test_method2(self):
9)          print('test method2 execution')
10)
```


DURGASOFT, # 202, 2nd Floor, HUDA Maitrivanam, Ameerpet, Hyderabad - 500038,
☎ 88 85 25 26 27, 72 07 21 24 27/28 | www.durgasoftonline.com
Maii: durgasoftonline@gmail.com

```
11)    def tearDown(self):
12)        print('tearDown method execution')
13)
14) unittest.main()
```

Output

```
setUp method execution
test method1 execution
tearDown method execution
.setUp method execution
test method2 execution
tearDown method execution
.
----------------------------------------------------------------------
Ran 2 tests in 0.016s

OK
```

Demo Application-4:

```
1)  import unittest
2)  class TestCaseDemo(unittest.TestCase):
3)      @classmethod
4)      def setUpClass(cls):
5)          print('*'*70)
6)          print('setUpClass:This method will be executed only once before executing all t
    est methods')
7)          print('*'*70)
8)
9)      def setUp(self):
10)         print('setUp method execution')
11)     def test_method1(self):
12)         print('test method1 execution')
13)
14)     def test_method2(self):
15)         print('test method2 execution')
16)
17)     def tearDown(self):
18)         print('tearDown method execution')
19)
20)     @classmethod
21)     def tearDownClass(cls):
22)         print('*'*70)
```


DURGASOFT, # 202, 2nd Floor, HUDA Maitrivanam, Ameerpet, Hyderabad - 500038,
☎ 88 85 25 26 27, 72 07 21 24 27/28 | www.durgasoftonline.com
Maii: durgasoftonline@gmail.com

```
23)       print('tearDownClass:This method will be executed only once after executing a
      ll test methods')
24)       print('*'*70)
25)
26) unittest.main()
```

Output

```
D:\durgaclasses>py test.py
**********************************************************************
setUpClass:This method will be executed only once before executing all test methods
**********************************************************************
setUp method execution
test method1 execution
tearDown method execution
.setUp method execution
test method2 execution
tearDown method execution
.**********************************************************************
tearDownClass:This method will be executed only once after executing all test methods
**********************************************************************
----------------------------------------------------------------------
Ran 2 tests in 0.016s

OK
```

Executing Test Suite:

By using unittest framework we can execute a group of test cases,which is nothing but test suite.

testcase1.py:

```
1)  import unittest
2)  class TestCase1(unittest.TestCase):
3)      def setUp(self):
4)          print('TestCase1:setUp')
5)
6)      def test1(self):
7)          print('TestCase1:test1')
8)
9)      def test2(self):
10)         print('TestCase1:test2')
11)
12)     def tearDown(self):
13)         print('TestCase1:tearDown')
```


DURGASOFT, # 202, 2nd Floor, HUDA Maitrivanam, Ameerpet, Hyderabad - 500038,
☎ 88 85 25 26 27, 72 07 21 24 27/28 | www.durgasoftonline.com
Maii: durgasoftonline@gmail.com

testcase2.py:

```
1)  import unittest
2)  class TestCase2(unittest.TestCase):
3)      def setUp(self):
4)          print('TestCase2:setUp')
5)
6)      def test1(self):
7)          print('TestCase2:test')
8)
9)      def tearDown(self):
10)         print('TestCase2:tearDown')
```

testsuite.py:

```
1)  import unittest
2)  from testcase1 import *
3)  from testcase2 import *
4)
5)  tc1=unittest.TestLoader().loadTestsFromTestCase(TestCase1)
6)  tc2=unittest.TestLoader().loadTestsFromTestCase(TestCase2)
7)
8)  ts=unittest.TestSuite([tc1,tc2])
9)  unittest.TextTestRunner().run(ts)
```

Output

```
D:\durgaclasses>py testsuite.py
TestCase1:setUp
TestCase1:test1
TestCase1:tearDown
.TestCase1:setUp
TestCase1:test2
TestCase1:tearDown
.TestCase2:setUp
TestCase2:test
TestCase2:tearDown
.
----------------------------------------------------------------------
Ran 3 tests in 0.016s

OK
```


DURGASOFT, # 202, 2nd Floor, HUDA Maitrivanam, Ameerpet, Hyderabad - 500038,
☎ 88 85 25 26 27, 72 07 21 24 27/28 | www.durgasoftonline.com
Maii: durgasoftonline@gmail.com

UNIT TESTING By using Selenium Automation Tool:

Selenium is functional testing automation tool. ie we test functionality of web application by using selenium tool.

How to install Selenium?

pip install selenium

webdriver: selenium package contains webdriver module, which contains several functions which are helpful to test functinality of webapplication.

How to Launch FireFox Browser:

We have to download the following Browser driver from the link:
https://www.seleniumhq.org/download/

file: geckodriver.exe

place this exe file inside D:\library folder.

The following is the code to launch Firefox browser

```
from selenium import webdriver
driver=webdriver.Firefox(executable_path='D:\library\geckodriver.exe')
```

Browser Interaction and Navigation of Web Pages:

We can perform browser interaction and navigation by using the following methods:

1) driver.get(url)
To open specified url

2) driver.maximize_window()
Maximizes current window that webdriver is using

3) driver.title
Returns the title of the current page

4) driver.current_url
url of the current loaded page

5) driver.refresh()
refresh current page

DURGASOFT, # 202, 2nd Floor, HUDA Maitrivanam, Ameerpet, Hyderabad - 500038,
☎ 88 85 25 26 27, 72 07 21 24 27/28 | www.durgasoftonline.com
Maii: durgasoftonline@gmail.com

6) driver.get(driver.current_url)
refresh current page

7) driver.back()
Goes one step backward in the browser history

8) driver.forward()
Goes one step forward in the browser history

9) driver.close()
To close current window

10) driver.quit()
To close all associated windows

Demo Program for Browser Navigation:

```
1) from selenium import webdriver
2) driver=webdriver.Firefox(executable_path='D:\library\geckodriver.exe')
3) driver.get('https://www.facebook.com/')
4) driver.maximize_window()
5) print('Title:',driver.title)
6) print('Current URL:',driver.current_url)
7) driver.get('http://durgasoftvideos.com/')
8) print('Title:',driver.title)
9) print('Current URL:',driver.current_url)
10) driver.back()
11) print('After Back current url:',driver.current_url)
12) driver.forward()
13) print('After Forward current url:',driver.current_url)
14) driver.close()
```

Output

```
D:\durgaclasses>py test.py
Title: Facebook – log in or sign up
Current URL: https://www.facebook.com/
Title: DurgaSoftware Training Youtube Videos by Industry Experts | Durgasoft
Current URL: http://durgasoftvideos.com/
After Back current url: https://www.facebook.com/
After Forward current url: http://durgasoftvideos.com/
```


DURGASOFT, # 202, 2nd Floor, HUDA Maitrivanam, Ameerpet, Hyderabad - 500038,
☎ 88 85 25 26 27, 72 07 21 24 27/28 | www.durgasoftonline.com
Maii: durgasoftonline@gmail.com

How to locate/find Web Elements:

Once we opened web page, we have to locate web elements like text box, links, submit buttons etc. We can identify each web element by using the locators like name, id, link text, class, xpath etc

We can find elements by using the following methods:
driver.find_element_by_id()
driver.find_element_by_name()
driver.find_element_by_xpath()
driver.find_element_by_css_selector()
driver.find_element_by_link_text()
etc

Alternatively we can use the following more convenient methods also.

driver.find_element(By.ID,'id')
driver.find_element(By.NAME,'name')
driver.find_element(By.LINK_TEXT,'text')
driver.find_element(By.CSS_SELECTOR,'css')
driver.find_element(By.XPATH,'xpath')

Testing Google Search Functionality by using unittest Framework:

```
1)  import unittest
2)  from selenium import webdriver
3)  import time
4)  class GoogleSearch(unittest.TestCase):
5)
6)      def setUp(self):
7)          global driver
8)          driver=webdriver.Firefox(executable_path='D:\\library\\geckodriver.exe')
9)          driver.get('http://google.co.in')
10)         driver.maximize_window()
11)
12)     def test(self):
13)         driver.find_element_by_name('q').send_keys('Mahesh Babu')
14)         time.sleep(5)
15)         driver.find_element_by_name('btnK').click()
16)         driver.find_element_by_class_name('LC20lb').click()
17)
18)     def tearDown(self):
19)         time.sleep(10)
```


DURGASOFT, # 202, 2nd Floor, HUDA Maitrivanam, Ameerpet, Hyderabad - 500038,
☎ 88 85 25 26 27, 72 07 21 24 27/28 | www.durgasoftonline.com
Maii: durgasoftonline@gmail.com

```
20)       driver.close()
21) unittest.main()
```

Testing HMS Login and Logout Functionality by using unittest Framework:

```
1)  from selenium import webdriver
2)  from selenium.webdriver.common.by import By
3)  import time
4)  import unittest
5)  class HMSLoginLogout(unittest.TestCase):
6)      @classmethod
7)      def setUpClass(cls):
8)          print('setUpClass method execution...')
9)          global driver
10)         driver=webdriver.Firefox(executable_path='D:\\library\\geckodriver.exe')
11)         driver.get('http://seleniumbymahesh.com')
12)         driver.maximize_window()
13)
14)     def test_login(self):
15)         print('test_login method execution...')
16)         driver.find_element(By.LINK_TEXT,'HMS').click()
17)         driver.find_element(By.NAME,'username').send_keys('admin')
18)         driver.find_element(By.NAME,'password').send_keys('admin')
19)         driver.find_element(By.NAME,'submit').click()
20)         time.sleep(10)
21)
22)     def test_logout(self):
23)         print('test_logout method execution...')
24)         driver.find_element(By.LINK_TEXT,'Logout').click()
25)
26)     @classmethod
27)     def tearDownClass(cls):
28)         print('tearDownClass method execution...')
29)         time.sleep(10)
30)         driver.quit()
31)
32) unittest.main()
```


DURGASOFT, # 202, 2nd Floor, HUDA Maitrivanam, Ameerpet, Hyderabad - 500038,
☎ 88 85 25 26 27, 72 07 21 24 27/28 | www.durgasoftonline.com
Maii: durgasoftonline@gmail.com

Demo Application to demonstrate Test Method Execution Order:

```
1)  import unittest
2)  class TestCaseDemo2(unittest.TestCase):
3)
4)      def test_C(self):
5)          print('test_C method execution from TestCaseDemo2')
6)      def test_B(self):
7)          print('test_B method execution from TestCaseDemo2')
8)      def test_A(self):
9)          print('test_A method execution from TestCaseDemo2')
10)
11) unittest.main()
```

Output

```
test_A method execution from TestCaseDemo2
.test_B method execution from TestCaseDemo2
.test_C method execution from TestCaseDemo2
.
----------------------------------------------------------------------
Ran 3 tests in 0.000s

OK
```

Note: In unittesting all test methods will be executed in alphabetical order.

Limitations of unittesting:

1) Test Results will be displayed to the console only and it is not possible to generate reports.
2) unittest framework always executes test methods in alphabetical order only and it is not possible to customize execution order.
3) As the part of batch execution (TestSuite), all test methods from specified TestCase classes will be executed and it is not possible to specify only particular test methods.
4) In unittesting only limited setUp and tearDown methods available.
   - setUpClass() → Before executing all test methods inside TestCase class
   - tearDownClass() → After executing all test methods inside TestCase class
   - setUp() → Before executing every test method
   - tearDown() → After executing every test method

If we want to perform certain activity before executing testsuite and after executing testsuite, unittest framework does not define any methods.
To overcome these limitations we should go for PyTest.

DURGASOFT, # 202, 2nd Floor, HUDA Maitrivanam, Ameerpet, Hyderabad - 500038,
☎ 88 85 25 26 27, 72 07 21 24 27/28 | www.durgasoftonline.com
Maii: durgasoftonline@gmail.com

PyTest Framework:

It is the advanced version of unittest framework.
This framework built on top of unittest framework.
It is not available by default with python. We have to install seperately by using pip command.

pip install pytest

PyTest Naming Rules:

While developing testscripts by using PyTest, compulsory we should follow naming conventions.

1) **File Name should Starts OR Ends with 'test'**
   Eg: test_google_search.py
   google_search_test.py

2) **Class Name should Starts with 'Test'**
   Eg: TestGoogleSearch
   TestDemoClass

3) **Method name should starts with 'test_'**
   Eg: test_method1()
   test_method2()

Demo Program:

pytest_demo1_test.py:

```
1)  import pytest
2)  def test_methodA():
3)      print('test_methodA execution')
4)
5)  def test_methodB():
6)      print('test_methodB execution')
```


DURGASOFT, # 202, 2nd Floor, HUDA Maitrivanam, Ameerpet, Hyderabad - 500038,
☎ 88 85 25 26 27, 72 07 21 24 27/28 | www.durgasoftonline.com
Maii: durgasoftonline@gmail.com

How to Run pytest Script:

We have to execute pytest scripts by using some special command py.test

D:\durgaclasses\pytest_scripts>py.test

It will execute all test methods present in all pytest testscripts of pytest_scripts folder.

If we want to execute a particular test script then we have to use command as follows

D:\durgaclasses\pytest_scripts>py.test pytest_demo1_test.py

Note: Bydefault pytest won't print any print() statements output to the console. If we want then we should use -s option. For getting verbose output we can use -v option

-v meant for verbose output
-s meant for print statements output

D:\durgaclasses\pytest_scripts>py.test -v -s pytest_demo1_test.py

How to implement setUp Mechanism:

In Pytest, we are not having any special methods like setUp(),tearDown(),setUpClass() and tearDownClass(). We can implement these methods by using @pytest.fixture() decorator.

We can declare any method with this decorator

```
@pytest.fixture()
def setUp():
        print('This method will be executed before every test method')
```

We can use any method name and need not be setUp()

We have to pass this method name as argument to test method.

```
def test_methodA(setUp):
```

Demo Program:

```
1)  import pytest
2)  @pytest.fixture()
3)  def setUp():
4)      print('This method will be executed before every test method')
5)
6)  def test_methodA(setUp):
```

```
7)     print('test_methodA execution')
8)
9)   def test_methodB(setUp):
10)    print('test_methodB execution')
11)
12) def test_methodC():
13)    print('test_methodC execution')
```

Ouput Flow:

setUp()
test_methodA()
setUp()
test_methodB()
test_methodC()

How to implement tearDown() Functionality:

We can implement both setUp and tearDown functionality by using @pytest.yield_fixture() decorator

Syntax:

```
@pytest.yield_fixture()
def setUptearDown():
        setUpactivity
        yield
        tearDownactivity
```

Note:

@pytest.fixture() → Meant for only setUp() activity
@pytest.yield_fixture() → Meant for both setUp() and tearDown() activities

Demo Program:

```
1)  import pytest
2)
3)  @pytest.yield_fixture()
4)  def setUptearDown():
5)     print('setUp activity')
6)     yield
7)     print('tearDown activity')
8)
9)  def test_methodA(setUptearDown):
10)    print('test_methodA execution')
```


DURGASOFT, # 202, 2nd Floor, HUDA Maitrivanam, Ameerpet, Hyderabad - 500038,
☎ 88 85 25 26 27, 72 07 21 24 27/28 | www.durgasoftonline.com
Maii: durgasoftonline@gmail.com

```
11) def test_methodB(setUptearDown):
12)     print('test_methodB execution')
13)
14) def test_methodC():
15)     print('test_methodC execution')
```

Output Flow:

```
setUp activity
test_methodA execution
tearDown activity

setUp activity
test_methodB execution
tearDown activity

test_methodC execution
```

How to implement setUpClass() and tearDownClass() Functionality:

We can use the same @pytest.yield_fixture() decorator. But we have to specify the scope attribute.

```
@pytest.yield_fixture(scope='module')
```

module scope means for all test methods only once
The default value for scope attribute is function, means it is applicable for every test method.

Demo Program:

```
1)  import pytest
2)
3)  @pytest.yield_fixture(scope='module')
4)  def setUptearDownClass():
5)      print('setUp activity')
6)      yield
7)      print('tearDown activity')
8)
9)  def test_methodA(setUptearDownClass):
10)     print('test_methodA execution')
11)
12) def test_methodB(setUptearDownClass):
```


DURGASOFT, # 202, 2nd Floor, HUDA Maitrivanam, Ameerpet, Hyderabad - 500038,
☎ 88 85 25 26 27, 72 07 21 24 27/28 | www.durgasoftonline.com
Maii: durgasoftonline@gmail.com

```
13)    print('test_methodB execution')
14)
15) def test_methodC(setUptearDownClass):
16)    print('test_methodC execution')
```

Output Flow:

setUp activity
test_methodA execution
test_methodB execution
test_methodC execution
tearDown activity

How to implement setUp and tearDown at Function Level and Module Level simultaneously:

```
1)  import pytest
2)
3)  @pytest.yield_fixture()
4)  def setUpTearDown():
5)     print('setUp method')
6)     yield
7)     print('tearDown method')
8)
9)  @pytest.yield_fixture(scope='module')
10) def setUpTearDownClass():
11)    print('setUpClass method')
12)    yield
13)    print('tearDownClass method')
14)
15) def test_method1(setUpTearDownClass,setUpTearDown):
16)    print('test_method1 execution...')
17)
18) def test_method2(setUpTearDownClass,setUpTearDown):
19)    print('test_method2 execution...')
```

Output

setUpClass method
setUp method
test_method1 execution...
tearDown method

setUp method
test_method2 execution...

DURGASOFT, # 202, 2nd Floor, HUDA Maitrivanam, Ameerpet, Hyderabad - 500038,
☎ 88 85 25 26 27, 72 07 21 24 27/28 | www.durgasoftonline.com
Maii: durgasoftonline@gmail.com

tearDown method
tearDownClass method

Note:
def test_method1(setUpTearDownClass,setUpTearDown):
The order of these arguments is not important

conftest.py:

If several modules required same setUp and tearDown functionality then it is not recommended to define that in every module seperately. We have to configure such common functionality inside a special file, which is nothing but conftest.py

If we define any setUp and tearDown functionality inside conftest.py file, then that functionality is bydefault available for every test module.

Demo Application:

conftest.py:

```
1)  import pytest
2)
3)  @pytest.yield_fixture()
4)  def setUpTearDown():
5)      print('setUp method')
6)      yield
7)      print('tearDown method')
8)
9)  @pytest.yield_fixture(scope='module')
10) def setUpTearDownClass():
11)     print('setUpClass method')
12)     yield
13)     print('tearDownClass method')
```

test.py:

```
1)  def test_method1(setUpTearDownClass,setUpTearDown):
2)      print('test:test_method1 execution...')
3)
4)  def test_method2(setUpTearDownClass,setUpTearDown):
5)      print('test:test_method2 execution...')
```

test1.py:

```
1)  def test_method1(setUpTearDownClass,setUpTearDown):
2)      print('test1:test_method1 execution...')
3)
4)  def test_method2(setUpTearDownClass,setUpTearDown):
5)      print('test1:test_method2 execution...')
```

Output

```
D:\durgaclasses>py.test -v -s test.py test1.py
setUpClass method
setUp method
test:test_method1 execution...
tearDown method

setUp method
test:test_method2 execution...
tearDown method
tearDownClass method

setUpClass method
setUp method
test1:test_method1 execution...
tearDown method

setUp method
test1:test_method2 execution...
tearDown method
tearDownClass method
```

Various Possible Ways to Run pytest Test Scripts:

1) py.test -v -s

To run all test methods in every test script of current working directory

2) py.test -v -s test1.py

To run all test methods of a particular test script

3) py.test -v -s test1.py test2.py test3.py

To run multiple test scripts

DURGASOFT, # 202, 2nd Floor, HUDA Maitrivanam, Ameerpet, Hyderabad - 500038,
☎ 88 85 25 26 27, 72 07 21 24 27/28 | www.durgasoftonline.com
Maii: durgasoftonline@gmail.com

4) py.test -v -s test1.py::test_method2

To run a particular test method

How to Customize Order of Tests in pytest:

In Pytest bydefault all test methods will be executed from top to bottom (but not alphabetical order)

Demo Program:

```
1) import pytest
2) def test_method3():
3)     print('test_method3 execution...')
4)
5) def test_method1():
6)     print('test_method1 execution...')
7)
8) def test_method2():
9)     print('test_method2 execution...')
```

Output

```
test_method3 execution...
test_method1 execution...
test_method2 execution...
```

But we can customize test methods execution based on our requirement. For this we have to use extra plugin:
pytest-ordering

We have to install by using the following command

pip install pytest-ordering

We have to specify the order by using the following decorator

@pytest.mark.run(order=n)

DURGASOFT, # 202, 2nd Floor, HUDA Maitrivanam, Ameerpet, Hyderabad - 500038,
☎ 88 85 25 26 27, 72 07 21 24 27/28 | www.durgasoftonline.com
Maii: durgasoftonline@gmail.com

Demo Program:

```
1)  import pytest
2)
3)  @pytest.mark.run(order=3)
4)  def test_methodC():
5)      print('test_methodC execution...')
6)
7)  @pytest.mark.run(order=1)
8)  def test_methodA():
9)      print('test_methodA execution...')
10)
11) @pytest.mark.run(order=2)
12) def test_methodB():
13)     print('test_methodB execution...')
14)
15) @pytest.mark.run(order=4)
16) def test_methodD():
17)     print('test_methodD execution...')
```

Output

```
test_methodA execution...
test_methodB execution...
test_methodC execution...
test_methodD execution...
```

How to generate Test Results in HTML Form:

We have to install pytest-html module.

```
pip install pytest-html
```

While executing test script we have to pass the required html file name as command line argument.

```
py.test -v -s test.py --html=results.html
```

results.html file will be generated in the current working directory.

DURGASOFT, # 202, 2nd Floor, HUDA Maitrivanam, Ameerpet, Hyderabad - 500038,
☎ 88 85 25 26 27, 72 07 21 24 27/28 | www.durgasoftonline.com
Maii: durgasoftonline@gmail.com

Testing Google Search Functionality by using pytest Framework:

```
1)  import pytest
2)  from selenium import webdriver
3)  import time
4)  class TestGoogleSearch:
5)      @pytest.yield_fixture()
6)      def setUpTearDown(self):
7)          global driver
8)          driver=webdriver.Firefox(executable_path='D:\\library\\geckodriver.exe')
9)          driver.get('http://google.co.in')
10)         driver.maximize_window()
11)         yield
12)         time.sleep(10)
13)         driver.close()
14)     def test_search(self,setUpTearDown):
15)         driver.find_element_by_name('q').send_keys('Mahesh Babu')
16)         time.sleep(5)
17)         driver.find_element_by_name('btnK').click()
18)         driver.find_element_by_class_name('LC20lb').click()
19)
20) py.test -s -v test.py
```

Testing HMS Login and Logout Functionality by using pytest Framework:

```
1)  import pytest
2)  from selenium import webdriver
3)  import time
4)  from selenium.webdriver.common.by import By
5)  class TestHMSLoginLogout:
6)      @pytest.yield_fixture(scope='module')
7)      def setUpTearDownClass(self):
8)          global driver
9)          driver=webdriver.Firefox(executable_path='D:\\library\\geckodriver.exe')
10)         driver.get('http://seleniumbymahesh.com')
11)         driver.maximize_window()
12)         yield
13)         time.sleep(10)
14)         driver.close()
15)     def test_login(self,setUpTearDownClass):
```


DURGASOFT, # 202, 2nd Floor, HUDA Maitrivanam, Ameerpet, Hyderabad - 500038,
☎ 88 85 25 26 27, 72 07 21 24 27/28 | www.durgasoftonline.com
Maii: durgasoftonline@gmail.com

```
16)       driver.find_element(By.LINK_TEXT,'HMS').click()
17)       driver.find_element(By.NAME,'username').send_keys('admin')
18)       driver.find_element(By.NAME,'password').send_keys('admin')
19)       driver.find_element(By.NAME,'submit').click()
20)       time.sleep(10)
21)    def test_logout(self,setUpTearDownClass):
22)       driver.find_element(By.LINK_TEXT,'Logout').click()
```

Django Application Testing:

Django provides inbuilt testing framework (django.test), which is developed based on unittest framework. This framework contains TestCase class, which is the child class of unittest.TestCase.

Demo Application:

models.py:

```
1)  from django.db import models
2)
3)  # Create your models here.
4)  class Employee(models.Model):
5)      eno=models.IntegerField()
6)      ename=models.CharField(max_length=64)
7)      esal=models.FloatField()
8)      eaddr=models.CharField(max_length=64)
9)      def get_salary(self):
10)         return self.esal
```

tests.py:

```
1)  from django.test import TestCase
2)  from testapp.models import Employee
3)
4)  # Create your tests here.
5)  class EmployeeDataSearch(TestCase):
6)      def setUp(self):
7)          print('setUp activity')
8)          Employee.objects.create(eno=100,ename='durga',esal=10000,eaddr='Hyd')
9)       Employee.objects.create(eno=200,ename='sunny',esal=20000,eaddr='Mumbai')
10)
11)     def test_employee_salaries(self):
12)         print('testing employee salaries')
```


DURGASOFT, # 202, 2nd Floor, HUDA Maitrivanam, Ameerpet, Hyderabad - 500038,
☎ 88 85 25 26 27, 72 07 21 24 27/28 | www.durgasoftonline.com
Maii: durgasoftonline@gmail.com

```
13)     global e1,e2
14)     e1=Employee.objects.get(ename='durga')
15)     e2=Employee.objects.get(ename='sunny')
16)     self.assertEqual(e1.get_salary(),10000)
17)     self.assertEqual(e2.get_salary(),20000)
18)
19)   def tearDown(self):
20)     print('deleting employees..')
21)     e1.delete()
22)     e2.delete()
```

How to Run Test Scripts:

We have to use the following command
py manage.py test

Various Possible Ways to Run Test Scripts:

```
# Run all the tests in the animals.tests module
$ ./manage.py test animals.tests

# Run all the tests found within the 'animals' package
$ ./manage.py test animals

# Run just one test case
$ ./manage.py test animals.tests.AnimalTestCase

# Run just one test method
$ ./manage.py test animals.tests.AnimalTestCase.test_animals_can_speak
```

Various Possible assert Methods of unittest Framework:

assertEqual(a,b) → a==b
assertTrue(x) → bool(x) is True
assertFalse(x) → bool(x) is False
assertIs(a,b) → a is b
assertIsNone(x) → x is None
assertIn(a,b) → a in b
assertIsInstance(a,b) → isInstance(a,b)

Note: assertIs(),assertNone(),assertIn() and assertInstance() all have opposite methods like assertIsNot() etc

DURGASOFT, # 202, 2nd Floor, HUDA Maitrivanam, Ameerpet, Hyderabad - 500038,
☎ 88 85 25 26 27, 72 07 21 24 27/28 | www.durgasoftonline.com
Maii: durgasoftonline@gmail.com

Testing of DRF Application:

DRF provides its own testing framework rest_framework.test to perform unittesting. This framework class contains APITestCase class which acts as base class to define our own TestCase classes.

Demo Application (djproject):

models.py:

```
1)  from django.db import models
2)
3)  # Create your models here.
4)  class hydjobs(models.Model):
5)      date=models.DateField();
6)      company=models.CharField(max_length=100);
7)      title=models.CharField(max_length=100);
8)      eligibility=models.CharField(max_length=100);
9)      address=models.CharField(max_length=100);
10)     email=models.EmailField();
11)     phonenumber=models.IntegerField()
```

testapp/api/views.py:

```
1)  from rest_framework import viewsets
2)  from testapp.models import hydjobs
3)  from testapp.api.serializers import HydJobsSerializer
4)  class HydJobsCRUDCBV(viewsets.ModelViewSet):
5)      serializer_class=HydJobsSerializer
6)      queryset=hydjobs.objects.all()
```

testapp/api/serializers.py:

```
1)  from rest_framework.serializers import ModelSerializer
2)  from testapp.models import hydjobs
3)  class HydJobsSerializer(ModelSerializer):
4)      class Meta:
5)          model=hydjobs
6)          fields='__all__'
```

testapp/api/urls.py:

```
1) from django.conf.urls import url,include
2) from rest_framework import routers
3) from testapp.api.views import HydJobsCRUDCBV
4) router=routers.DefaultRouter()
5) router.register('hydjobsinfo',HydJobsCRUDCBV)
6) urlpatterns = [
7)     url(r'', include(router.urls)),
8) ]
```

project level urls.py:

```
1) from django.conf.urls import url,include
2) from django.contrib import admin
3) from testapp import views
4)
5) urlpatterns = [
6)     url(r'^admin/', admin.site.urls),
7)     url(r'^$', views.index),
8)     url(r'^hydjobs/', views.hydjobs1),
9)     url(r'^blorejobs/', views.blorejobs),
10)    url(r'^punejobs/', views.punejobs),
11)    url(r'^chennaijobs/', views.chennaijobs),
12)    url(r'^api/', include('testapp.api.urls')),
13) ]
```

testapp/api/tests.py:

```
1) from rest_framework.test import APITestCase
2) from testapp.models import hydjobs
3) class JobsAPITestCase(APITestCase):
4)     def setUp(self):
5)         hydjobs.objects.create(date="2019-03-04",
6)                     company="IBMSOFT",title= "SoftwareEngineer", eligibility="B.Tech",
7)                         address="Maitrivanam,Hyderabad",
8)                         email= "durga@durgasoft.com",
9)                         phonenumber=9898989898)
10)    def test_get_method(self):
11)        url='http://127.0.0.1:8000/api/hydjobsinfo/'
12)        response=self.client.get(url)
13)        print(response.status_code)
14)        self.assertEqual(response.status_code,200)
```


DURGASOFT, # 202, 2nd Floor, HUDA Maitrivanam, Ameerpet, Hyderabad - 500038,
☎ 88 85 25 26 27, 72 07 21 24 27/28 | www.durgasoftonline.com
Maii: durgasoftonline@gmail.com

```
15)     qs=hydjobs.objects.filter(company="IBMSOFT")
16)     self.assertEqual(qs.count(),1)
17)   def test_post_method_success(self):
18)     url='http://127.0.0.1:8000/api/hydjobsinfo/'
19)     data={
20)     'date': "2018-08-24",
21)     'company': "DURGASOFT",
22)     'title': "Software Engineer",
23)     'eligibility': "B.Tech",
24)     'address': "Maitrivanam,Hyderabad",
25)     'email': "durga@durgasoft.com",
26)     'phonenumber': 9898989898
27)     }
28)     response=self.client.post(url,data,format='json')
29)     print(response.status_code)
30)     self.assertEqual(response.status_code,201)
31)   def test_post_method_fail(self):
32)     url='http://127.0.0.1:8000/api/hydjobsinfo/'
33)     data={
34)     'eligibility': "B.Tech",
35)     'address': "Maitrivanam,Hyderabad",
36)     'email': "durga@durgasoft.com",
37)     'phonenumber': 9898989898
38)     }
39)     response=self.client.post(url,data,format='json')
40)     print(response.status_code)
41)     self.assertEqual(response.status_code,400)
42)   def test_delete_method_success(self):
43)     url='http://127.0.0.1:8000/api/hydjobsinfo/1/'
44)     response=self.client.delete(url)
45)     print(response.status_code)
46)     self.assertEqual(response.status_code,204)
```

py manage.py test testapp.api.tests

DURGASOFT, # 202, 2nd Floor, HUDA Maitrivanam, Ameerpet, Hyderabad - 500038,
☎ 88 85 25 26 27, 72 07 21 24 27/28 | www.durgasoftonline.com
Maii: durgasoftonline@gmail.com