PyTest

What is pytest?

- Pytest is a unit testing framework which allows us to write test codes using python.

Unit testing:

- **UNIT TESTING** is a level of software testing where individual units/ components of a software are tested.
- Unit testing of software applications is done during the development (coding) of an application. The objective of Unit Testing is to isolate a section of code and verify its correctness. In procedural programming, a unit may be an individual function or procedure.
- Unit testing is usually performed by the developer.

Note:

- Basically "PyTest" is used to perform unit testing, that is for every customer requirement develop develops business class, before giving the build to testing team, developers need to ensure that the source code is working properly. For this he will do unit testing, but testing every business class manually is time consuming because of this reason, developer creates "test class" which will test the respective business class. While developing the test class developer has to perform the following steps.
    1. Call the methods of business class.
    2. Compare expected and actual results.
    3. Generate the report.
    4. Executes all the test classes.
    5. Re-execute failed test classes after debugging etc.
- In order to do above steps developer uses "**PyTest**" because PyTest has readymade features to perform the above mentioned steps and the same steps need to be done in selenium also, hence we use "PyTest" in automation.

Introduction of PyTest

- Pytest is a framework that makes building simple and scalable tests easy. Tests are expressive and readable

It can be used for unit, functional, integration, and end-to-end testing

Why use PyTest?

Some of the advantages of pytest are

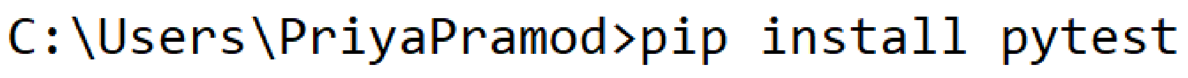
- Very easy to start with because of its simple and easy syntax.

- Can run tests in parallel.
- Can run a specific test or a subset of tests.
- Automatically detect tests.
- Skip tests.
- Dependency test.
- Grouping of test methods.
- Provide baseline for test functions, classes, modules and sessions.
- Generates report.

Installing PyTest

Installing the PyTest is very simple.

Run the following command in your command line:

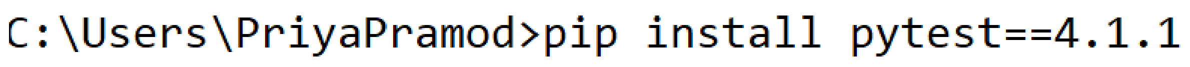


```
C:\Users\PriyaPramod>pip install pytest
```

To check that you installed PyTest:

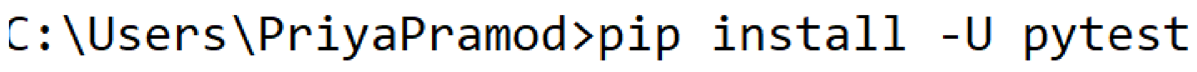
```
C:\Users\PriyaPramod>pytest --version
This is pytest version 4.1.1, imported from c:\users\priyapramod\appdata\local
\programs\python\python36-32\lib\site-packages\pytest.py
```

To install the specific version of PyTest:



```
C:\Users\PriyaPramod>pip install pytest==4.1.1
```

To Update the PyTest:



```
C:\Users\PriyaPramod>pip install -U pytest
```

PyTest Usage

- In order to develop the test function, test class, test module and test package. PyTest expects our test function/class/module/package name to be begin with "test_" or end with "_test".
- If we don't develop the function/class/module/package name begin or ending with test, than PyTest will never execute.

Note:

- If the "package name" is not starting or ending with "test_ or _test" than, anything which is present inside that module will be ignored by the PyTest.

- If the “module name” is not starting or ending with “test_ or _test” than, anything which is present inside that module (Example: Classes, methods) will be ignored by the PyTest.
- If the “Class name” is not starting or ending with “Test_ or _Test” than, anything which is present inside that class (functions) will be ignored by the PyTest.
- If the “Function Name” is not starting or ending with “test_ or _test” than, those functions will be ignored by the PyTest.

Important:

- Make sure every test package/module/class/functions are prefixed or suffixed with test.

Package name: example -> test_package or package_test

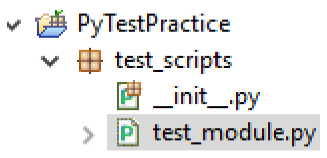
Module name: example -> test_module or module_test

Class Name: example -> TestClass or ClassTest

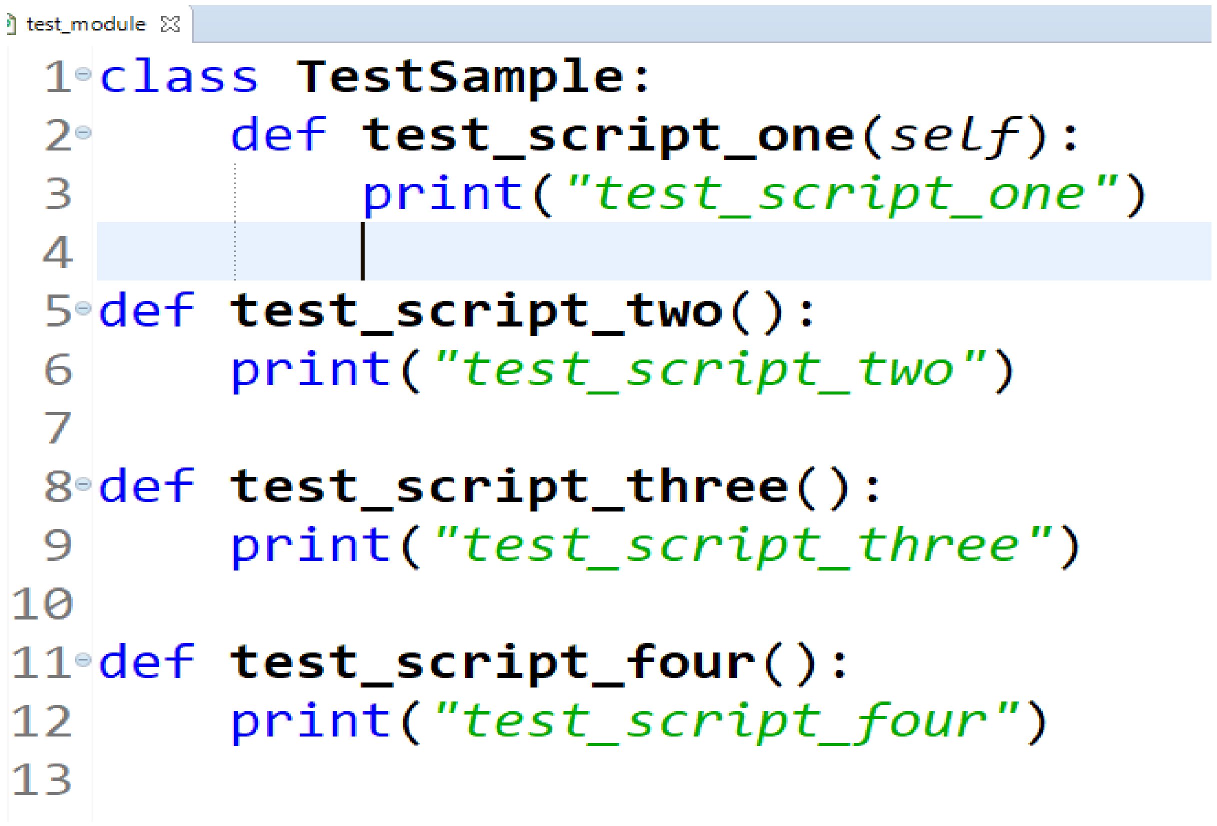
Note: In python, for class name, first character of every word should start with Upper Case. So does PyTest also follow the same naming convention?

Function name: example -> test_function or function_test

Simple folder structure of PyTest:



```
PyTestPractice
  test_scripts
    __init__.py
    test_module.py
```



test_module

```
class TestSample:
    def test_script_one(self):
        print("test_script_one")

def test_script_two():
    print("test_script_two")

def test_script_three():
    print("test_script_three")

def test_script_four():
    print("test_script_four")

```

How to run PyTest

There are 3 ways to run the PyTest.

1. From command prompt.
2. From eclipse/PyCharm.
3. From PYTEST MAIN method.

Command Prompt:

1. Using python command
2. Using pytest command

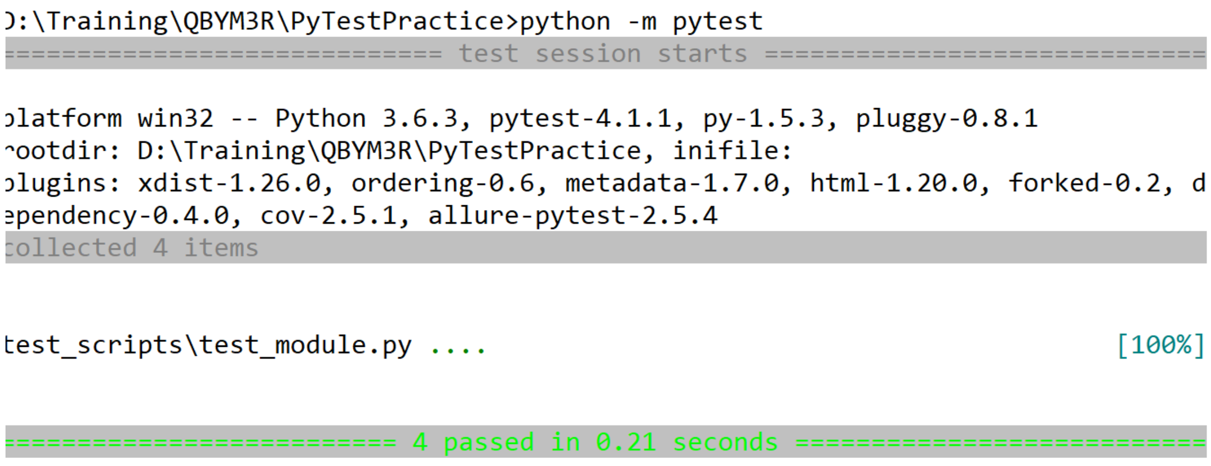
1. **Python command to run the pytest.**

- Open the command prompt and navigate to the location where you have created the pytest project.
- Write the below command and press enter button.

```
D:\Training\QBYM3R\PyTestPractice>python -m pytest
```

- As soon as you press the enter button. The above command will look into the project folder to check the packages which is starting with the “test_” or ending with the “_test”, if found, than it will look for the modules which is starting with the “test_” or ending with the “_test”, if found, it will search for the class and functions which is starting with the “test_” or ending with the “_test”, if found, pytest will run all the test functions.

And show the output as below,



```
:\Training\QBYM3R\PyTestPractice>python -m pytest
============================ test session starts =============================

latform win32 -- Python 3.6.3, pytest-4.1.1, py-1.5.3, pluggy-0.8.1
ootdir: D:\Training\QBYM3R\PyTestPractice, inifile:
lugins: xdist-1.26.0, ordering-0.6, metadata-1.7.0, html-1.20.0, forked-0.2, d
ependency-0.4.0, cov-2.5.1, allure-pytest-2.5.4
ollected 4 items


test_scripts\test_module.py ....                                        [100%]


========================= 4 passed in 0.21 seconds ==========================
```

Note:

- There were 4 test functions present in the test_module.py. All the four got executed successfully.

- Here, successful execution of the test method will be represented as dot ("."), failure will be represented as "F".
-

2. Using PyTest Command.

- Write the command "PyTest" in command prompt and click enter button.

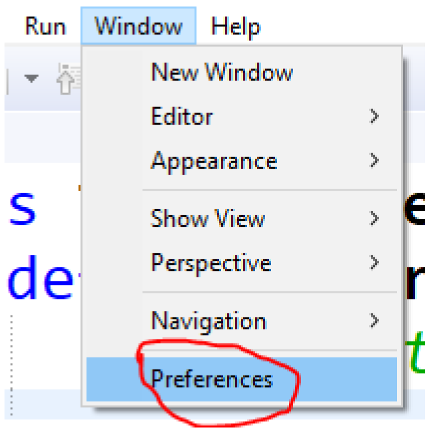
```
D:\Training\QBYM3R\PyTestPractice>pytest
```

The above command also executes the test functions and provides the same output.

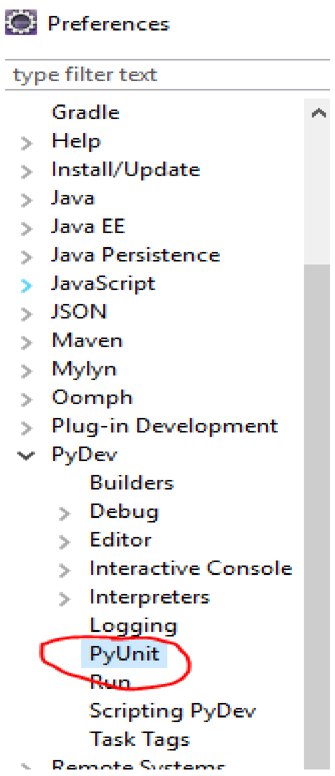
```
D:\Training\QBYM3R\PyTestPractice>pytest
============================= test session starts =============================

platform win32 -- Python 3.6.3, pytest-4.1.1, py-1.5.3, pluggy-0.8.1
rootdir: D:\Training\QBYM3R\PyTestPractice, inifile:
plugins: xdist-1.26.0, ordering-0.6, metadata-1.7.0, html-1.20.0, forked-0.2, d
ependency-0.4.0, cov-2.5.1, allure-pytest-2.5.4
collected 4 items


test_scripts\test_module.py ....                                         [100%]


========================== 4 passed in 0.09 seconds ===========================
```

Eclipse

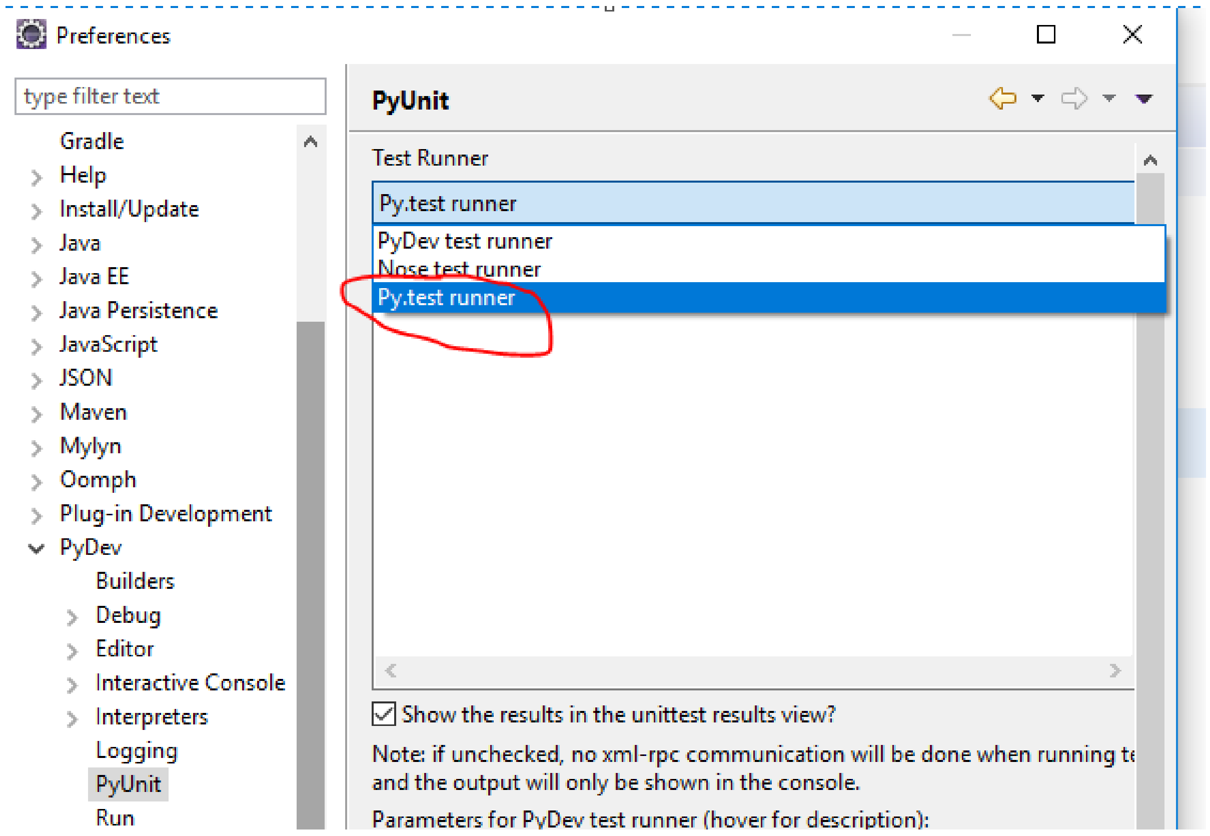
Step 1: Go to window -> Preference



Step 2: Go to PyDev -> PyUnit



Step 3: Select **"Py.test runner"** from the drop down.



Py.test runner

☑ Show the results in the unittest results view?

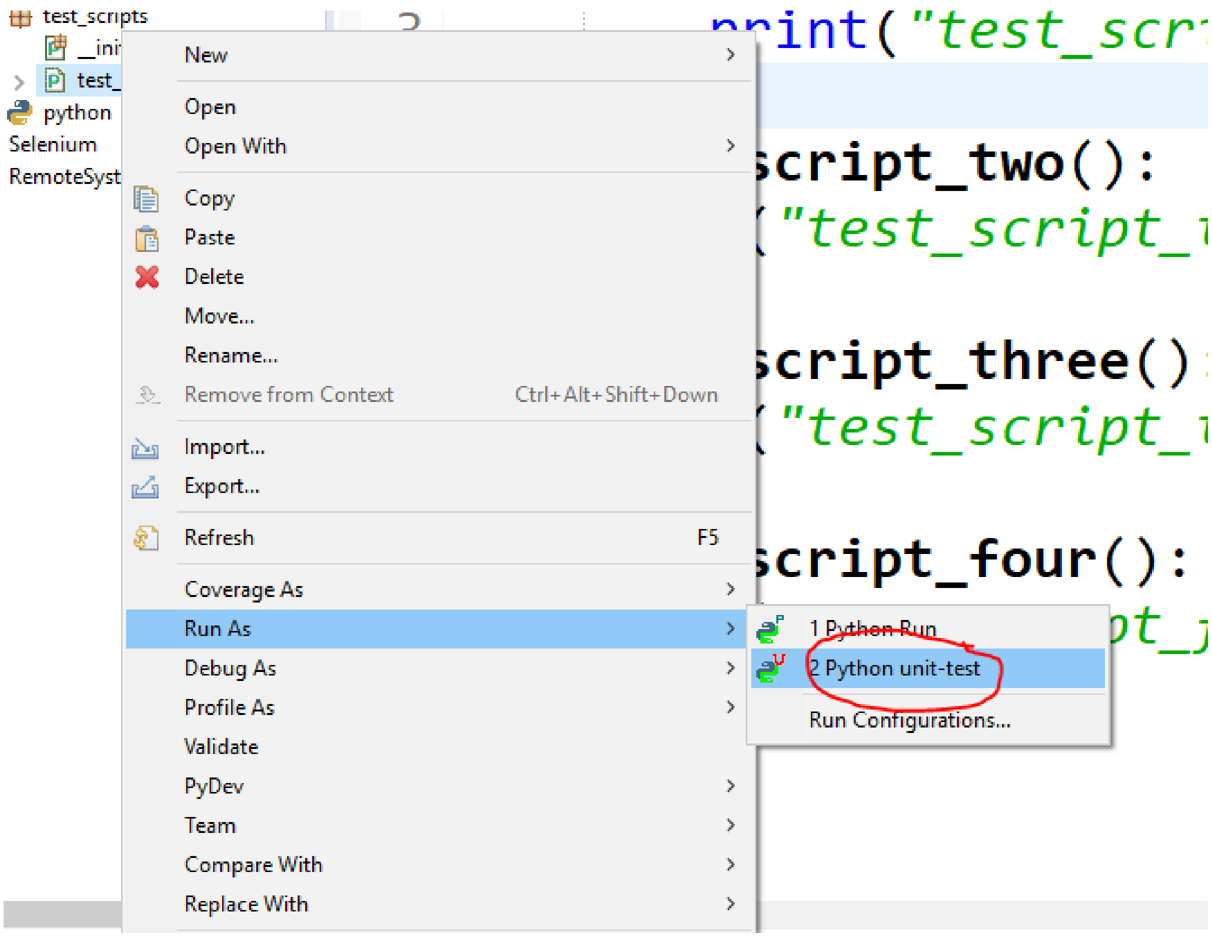
Note: if unchecked, no xml-rpc communication will be done when running te
and the output will only be shown in the console.

Parameters for PyDev test runner (hover for description):

Step 4: Click Apply -> Apply and close.

Step 5: Now, right click on the module which you want to run and select **"Python unit test"**

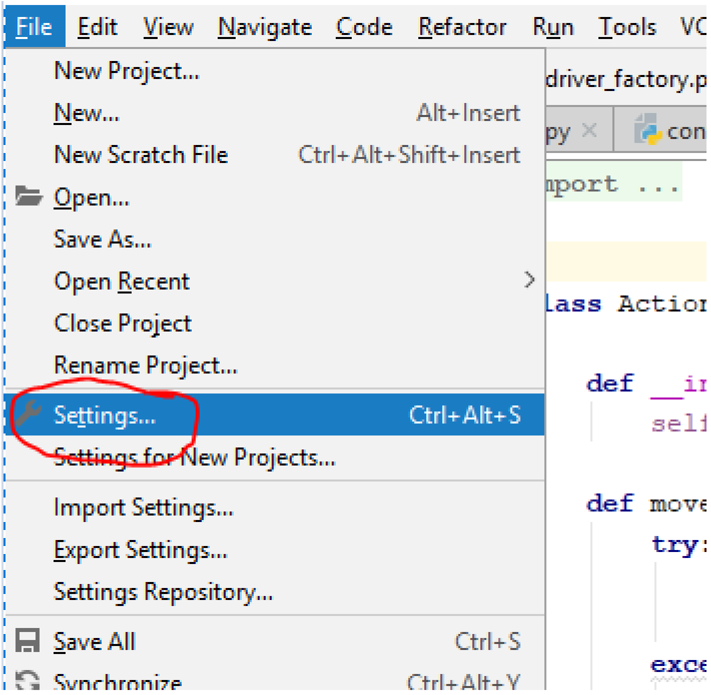
Pramod K S



```
print("test_scr
script_two():
("test_script_
script_three()
("test_script_
script_four():
pt_
```

PyCharm

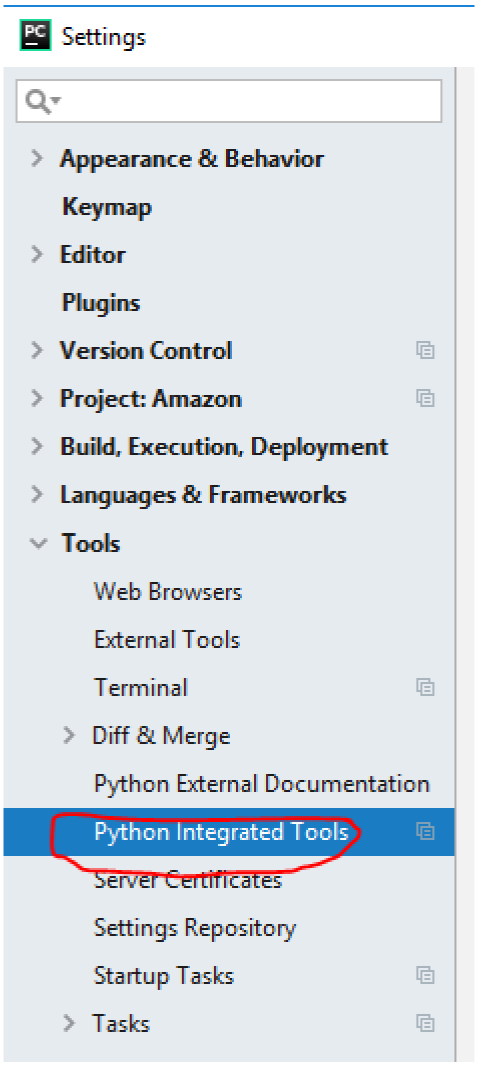
Step 1: Go to File -> Setting



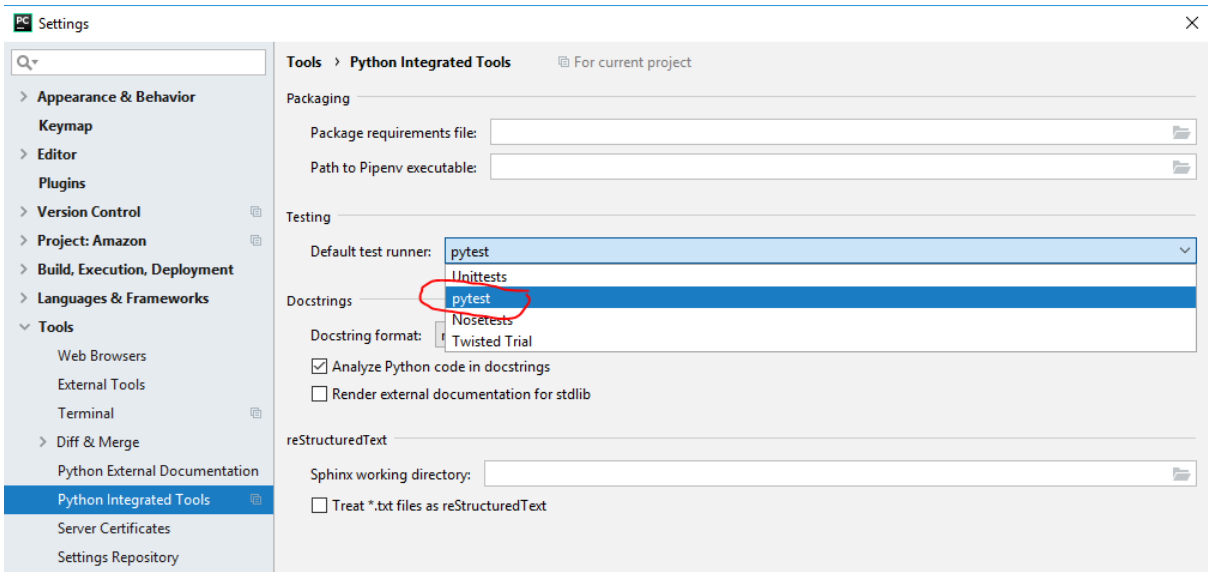
```
mport ...
lass Actio
    def __i
        sel
    def move
        try:
        exce
```

Step 2: Go to tools -> Python Integrated Tools

Pramod K S



Step 3: Select “PyTest” from the drop down.


Settings

Tools › Python Integrated Tools   For current project

Packaging

Package requirements file:

Path to Pipenv executable:

Testing

Default test runner: pytest

Unittests
pytest
Nosetests
Twisted Trial

Docstrings

Docstring format:

☑ Analyze Python code in docstrings

☐ Render external documentation for stdlib

reStructuredText

Sphinx working directory:

☐ Treat *.txt files as reStructuredText

Step 4: Right click on the module, which you want to run and select the run.

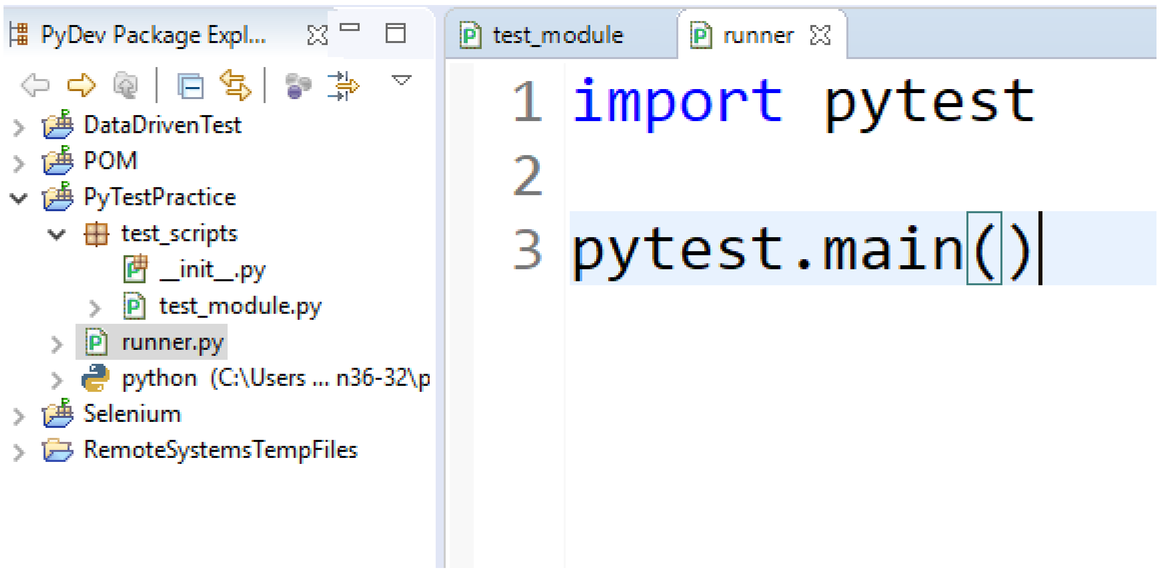
Using Pytest Main method

- Pytest provide the method called “main” method to execute the pytest test functions.
- Main method takes two arguments, commands and plugins.

Note: Will discuss the commands and plugins in the coming sessions.

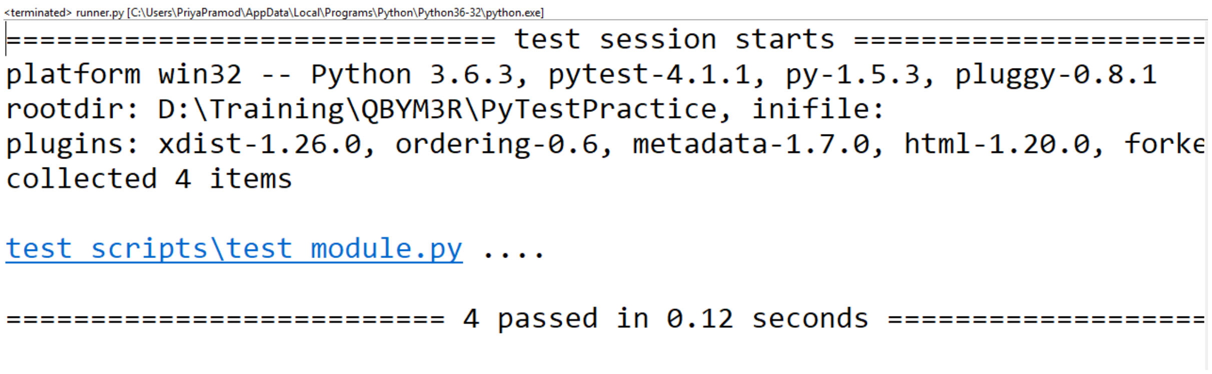
- Now, in order to use the main method, create one python file. And import the pytest and call the main method.
- Select the file and run as python run. Main method will execute the test functions.

Runner.py file.



```python
import pytest

pytest.main()
```

Output:



```
<terminated> runner.py [C:\Users\PriyaPramod\AppData\Local\Programs\Python\Python36-32\python.exe]
============================= test session starts =====================
platform win32 -- Python 3.6.3, pytest-4.1.1, py-1.5.3, pluggy-0.8.1
rootdir: D:\Training\QBYM3R\PyTestPractice, inifile:
plugins: xdist-1.26.0, ordering-0.6, metadata-1.7.0, html-1.20.0, forke
collected 4 items

test_scripts\test_module.py ....

========================== 4 passed in 0.12 seconds ===================
```