JENKINS uses PyTest

**pip install pytest** :-- command to install pytest

pytest –version :-- command to get pytest version

pytest –h :-- command to check pytest is working properly

**test\_filename.py || filename\_test.py** ;-- syntax for pytest file

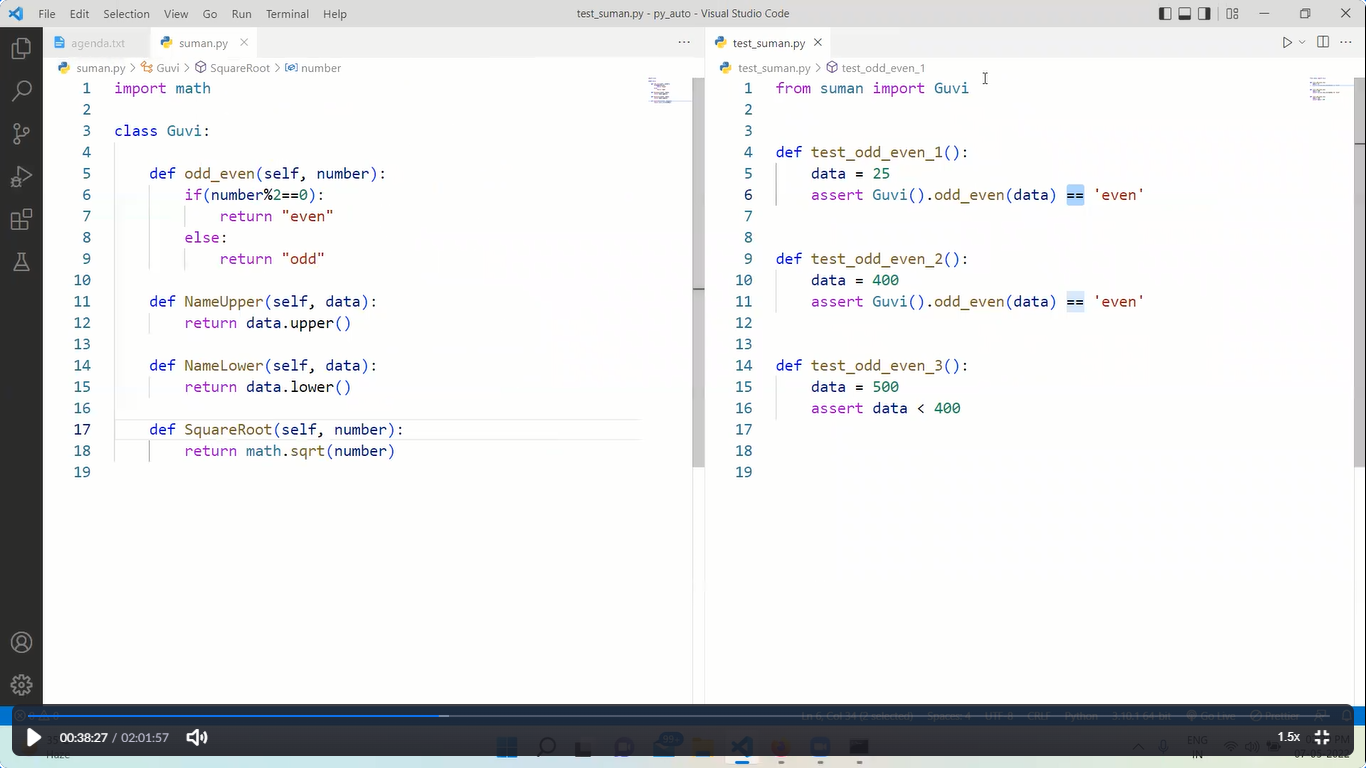
def test\_method\_name :--syntax to write the method in pytest file

pytest test\_filename.py :-- command to run pytest file

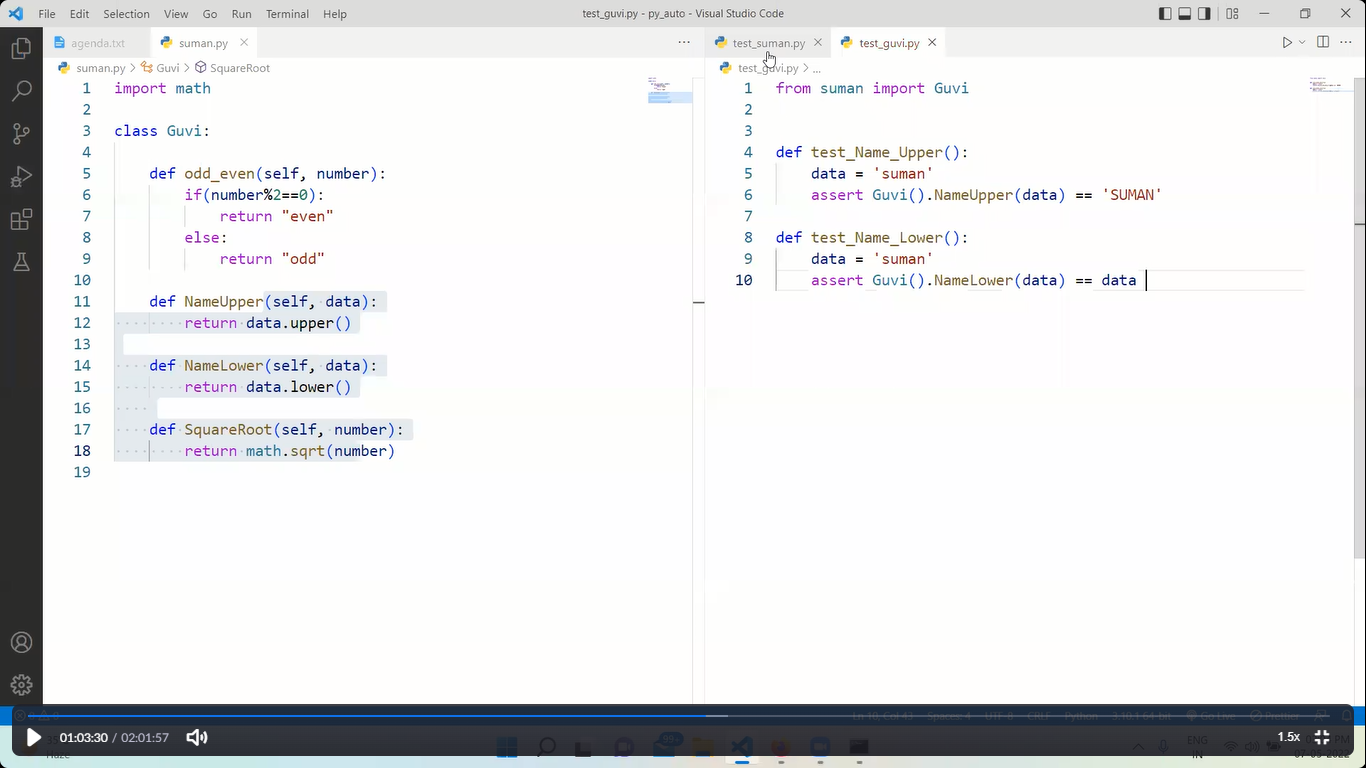
pytest –v test\_filename.py :-- command to run pytest with detailed report

pytest –v test\_file1.py test\_file2.py :-- command to run 2 test files at 1 go

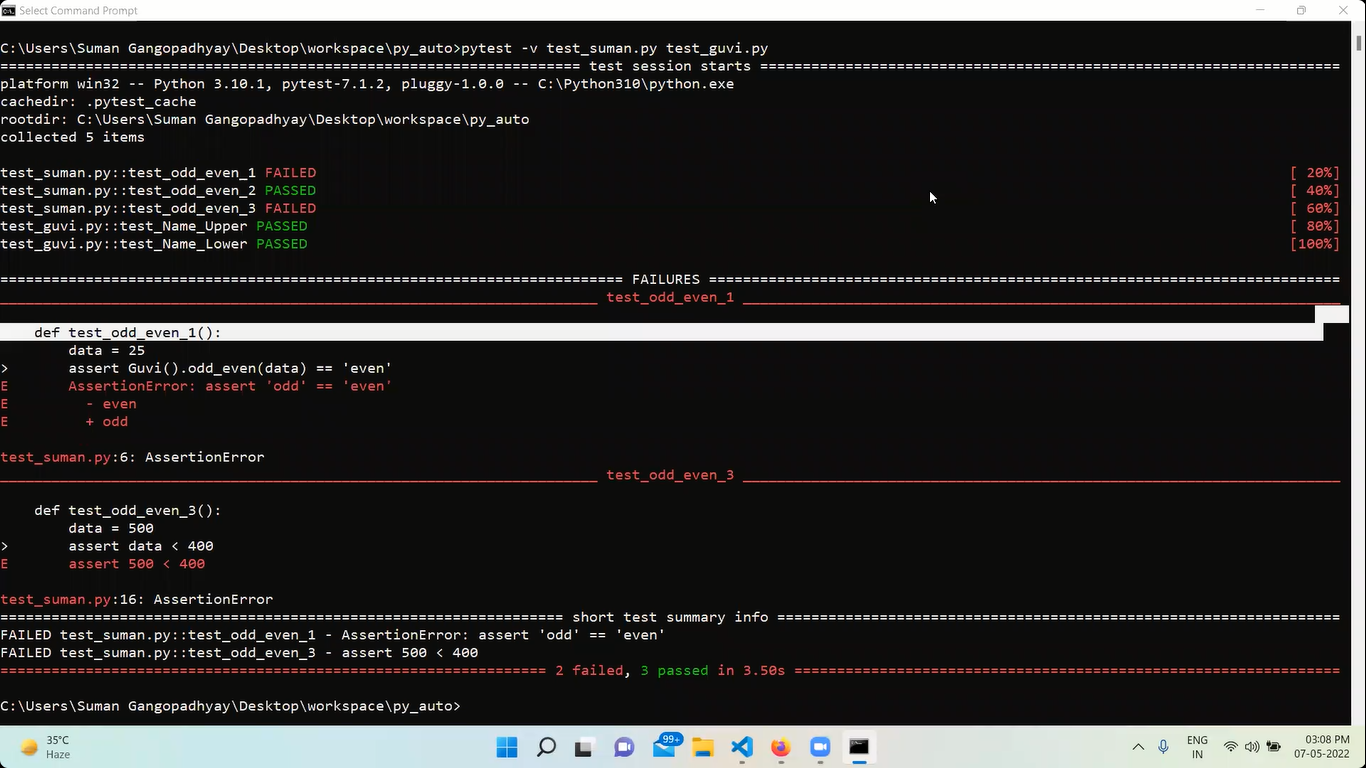
Here we created a suman.py & we wrote pytest for suman.py

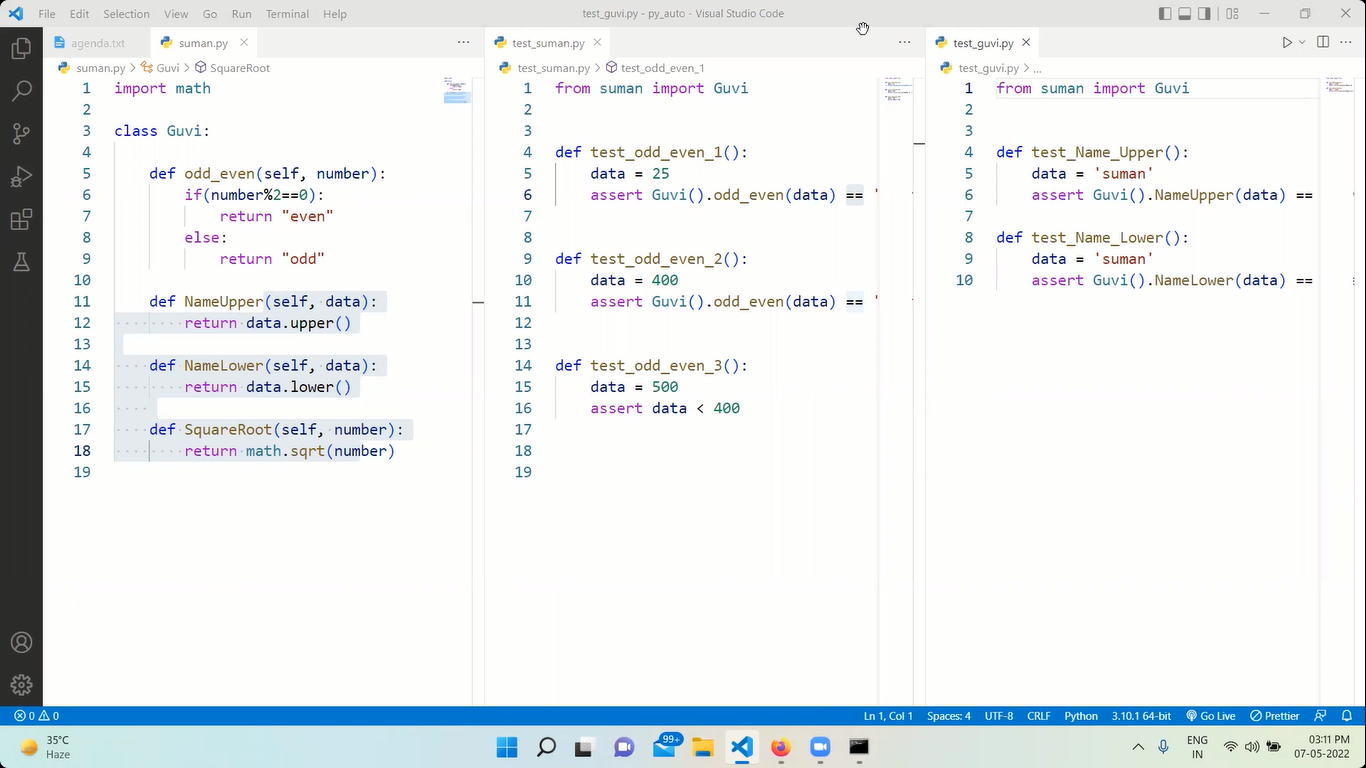


\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*



Above pic is to run 2 test files at 1 go





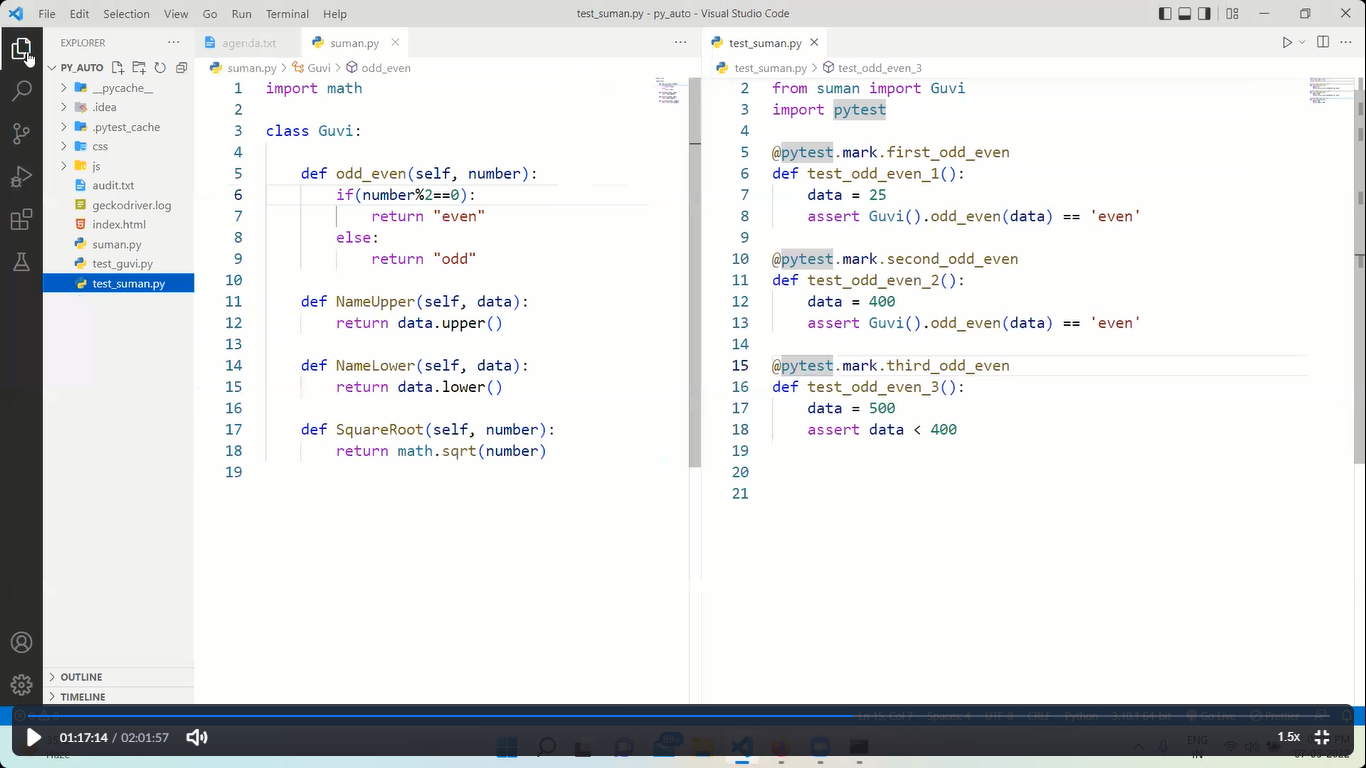
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

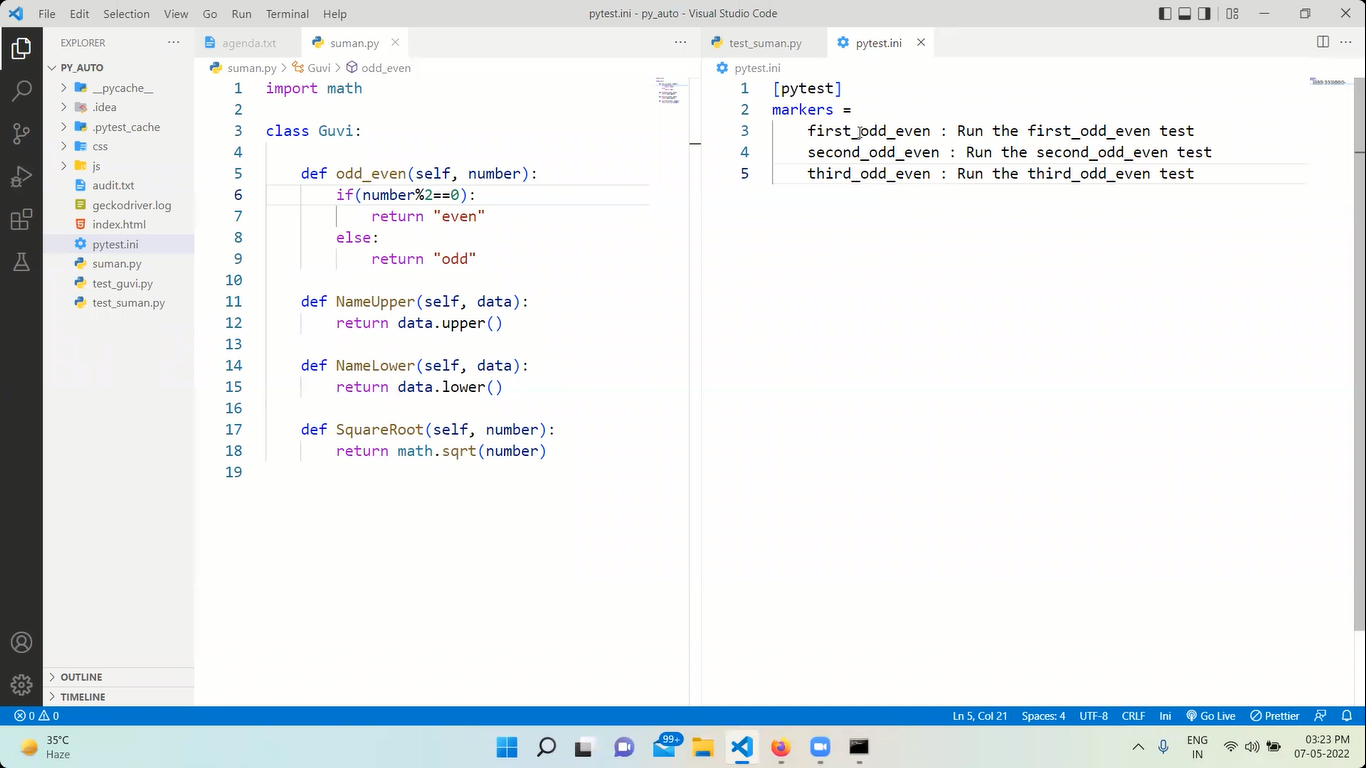
MARKERS:--

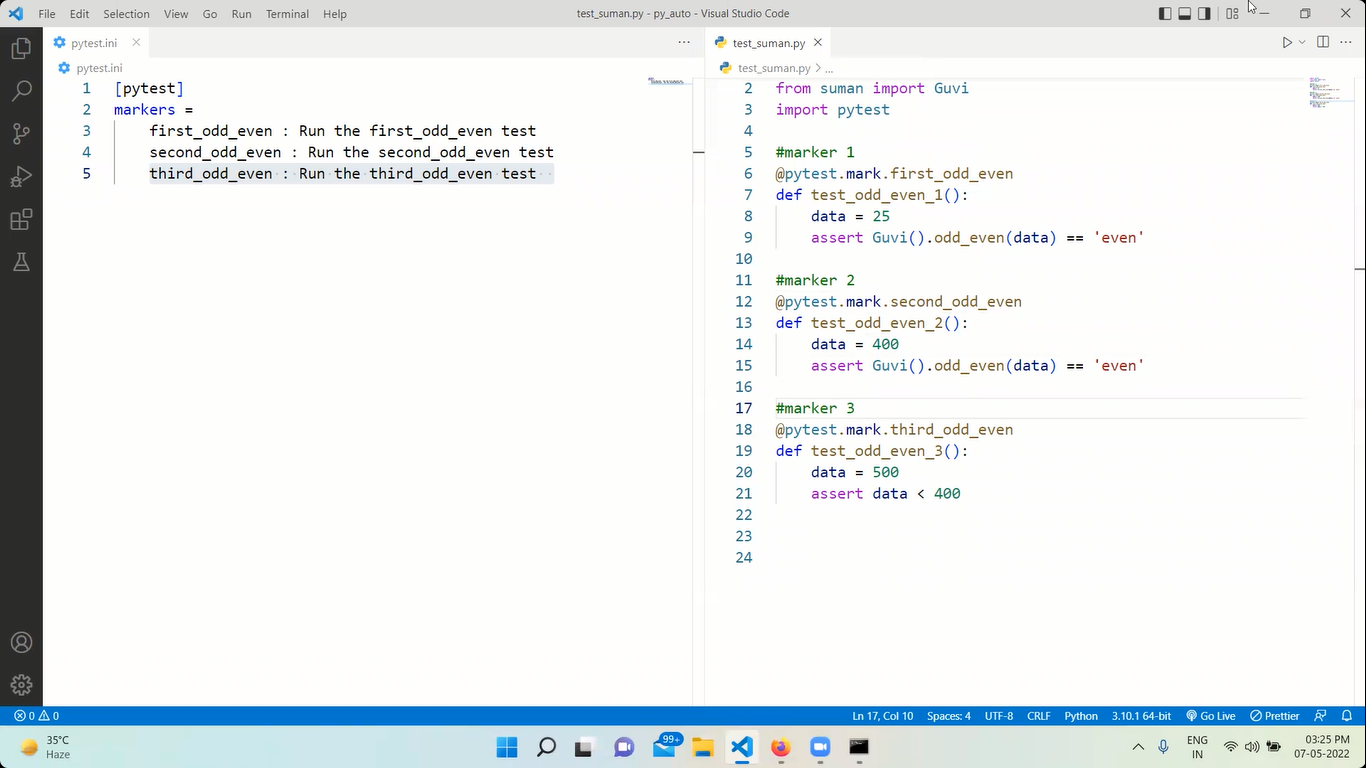
@pytest.mark.name\_of\_marker

To work with the markers we need to write import pytest

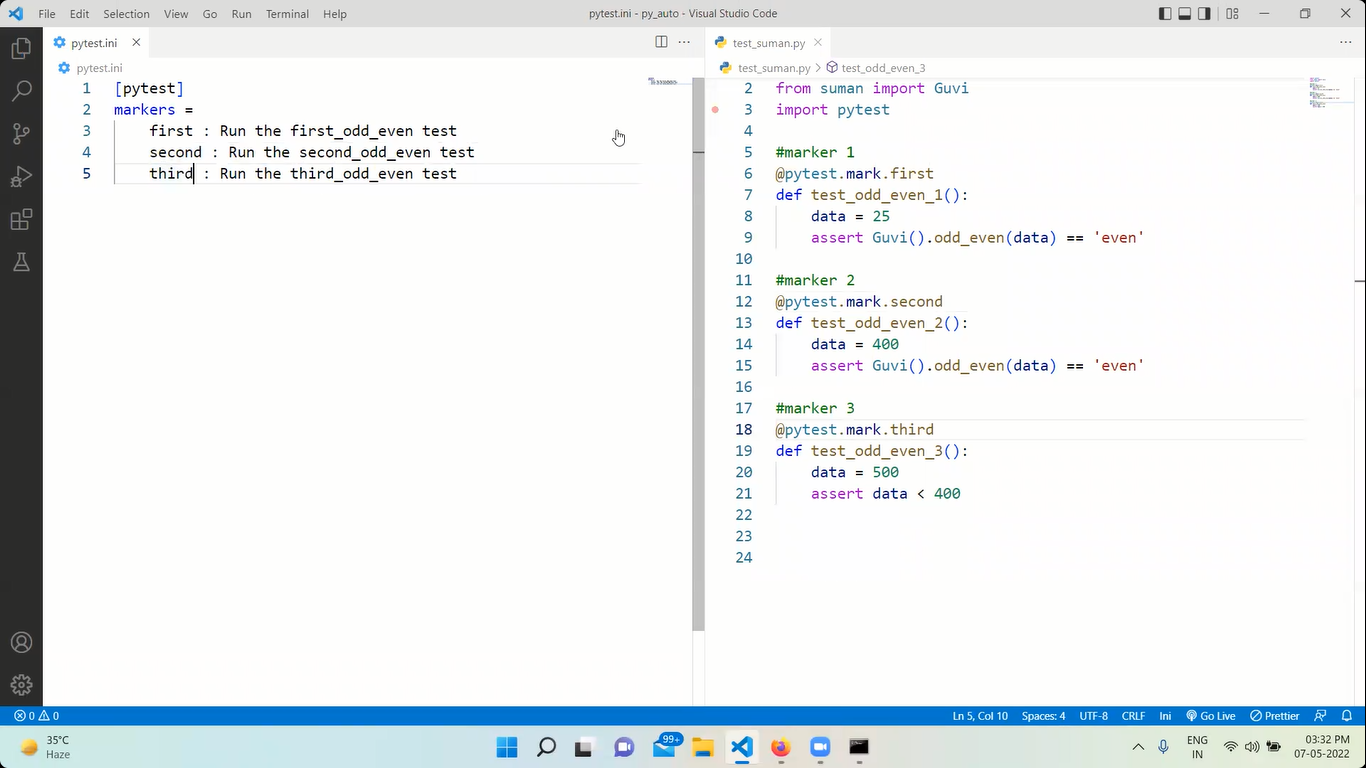
Pytest.ini :-- file should be created where the test file is created



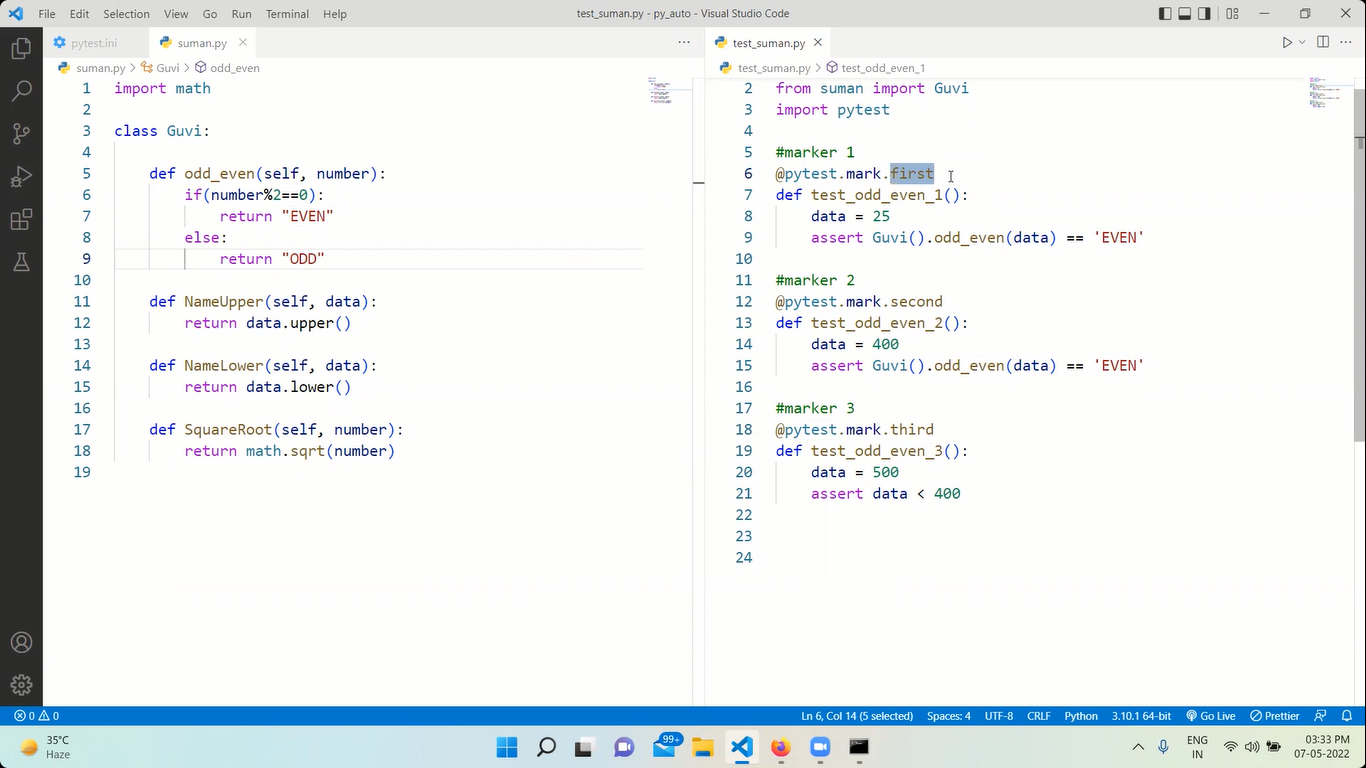




\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*



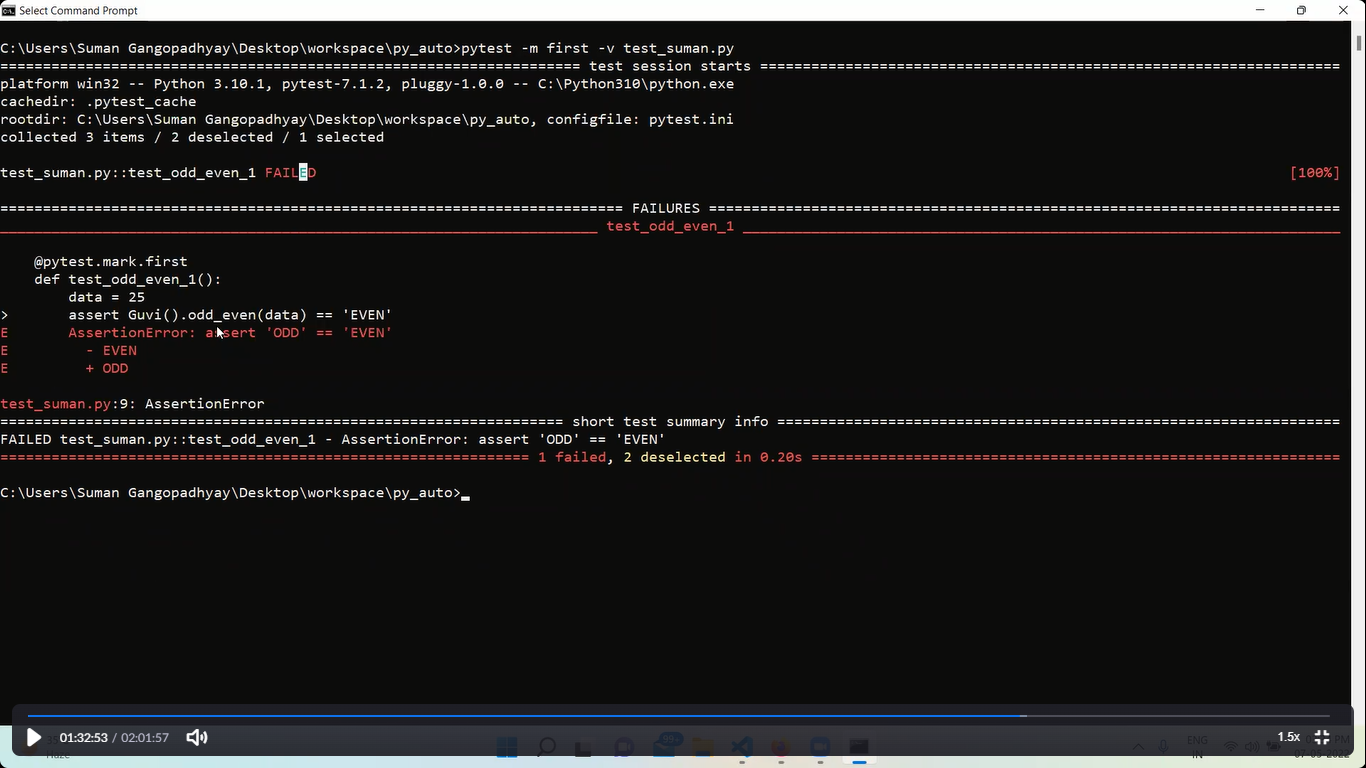
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

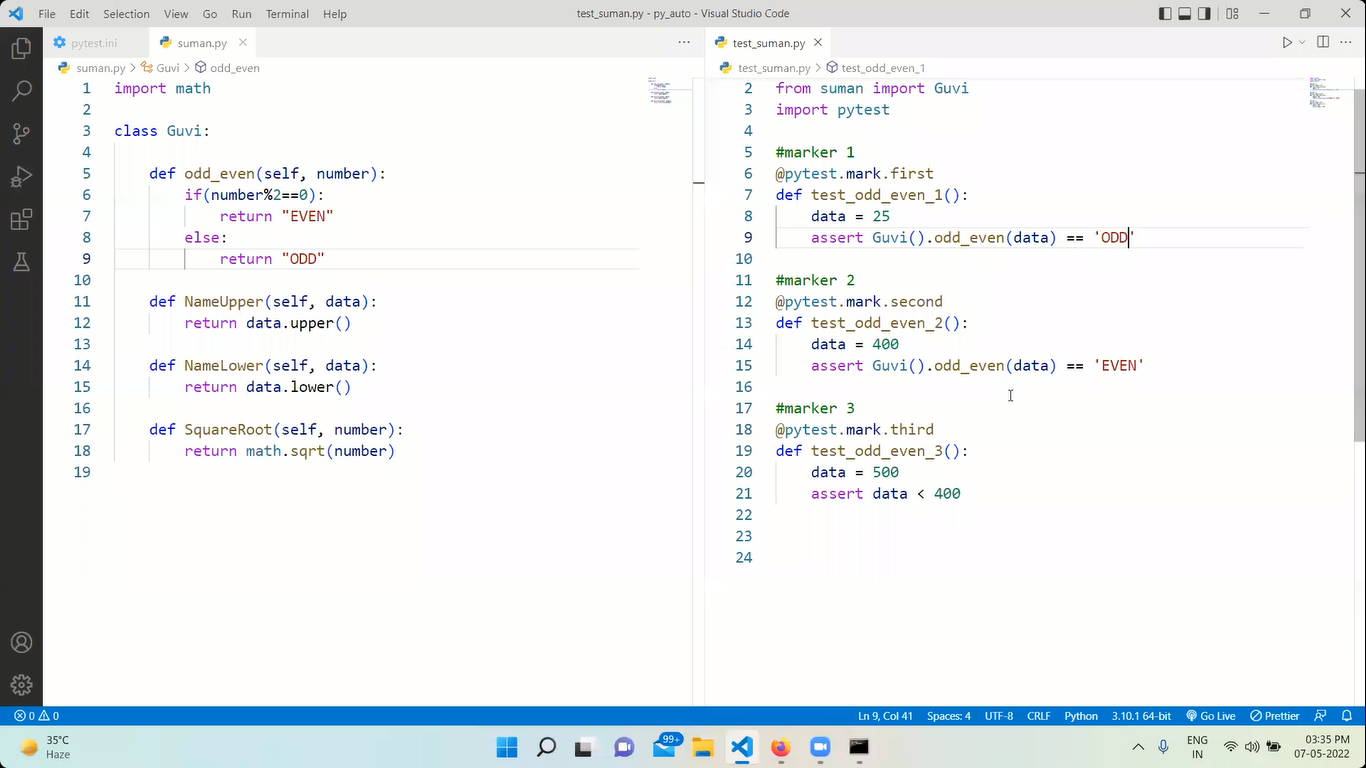


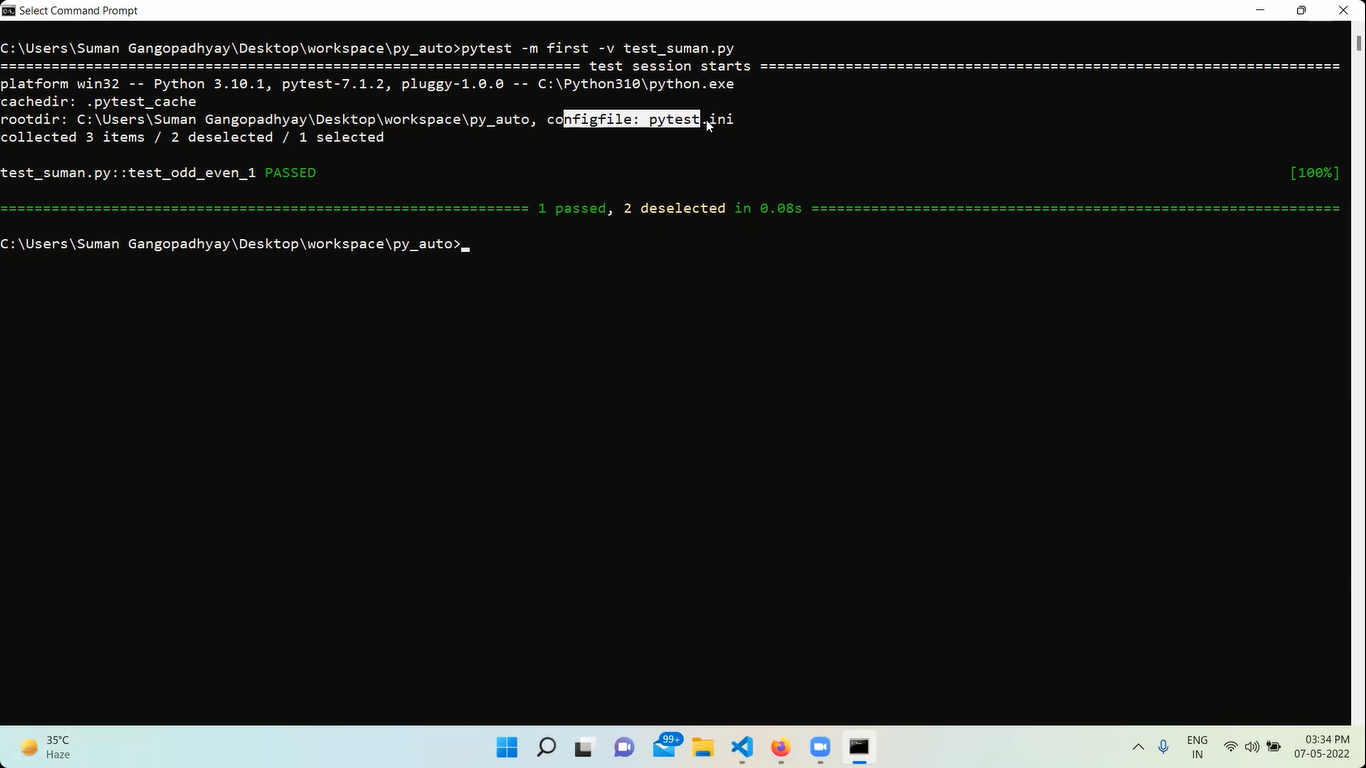
Above I have updated the suman.py and test\_sumaan.py file

Pytest –m marker\_name –v test\_file.py :-- command to run specific marker from test file

Pytest –m first –v test\_suman.py







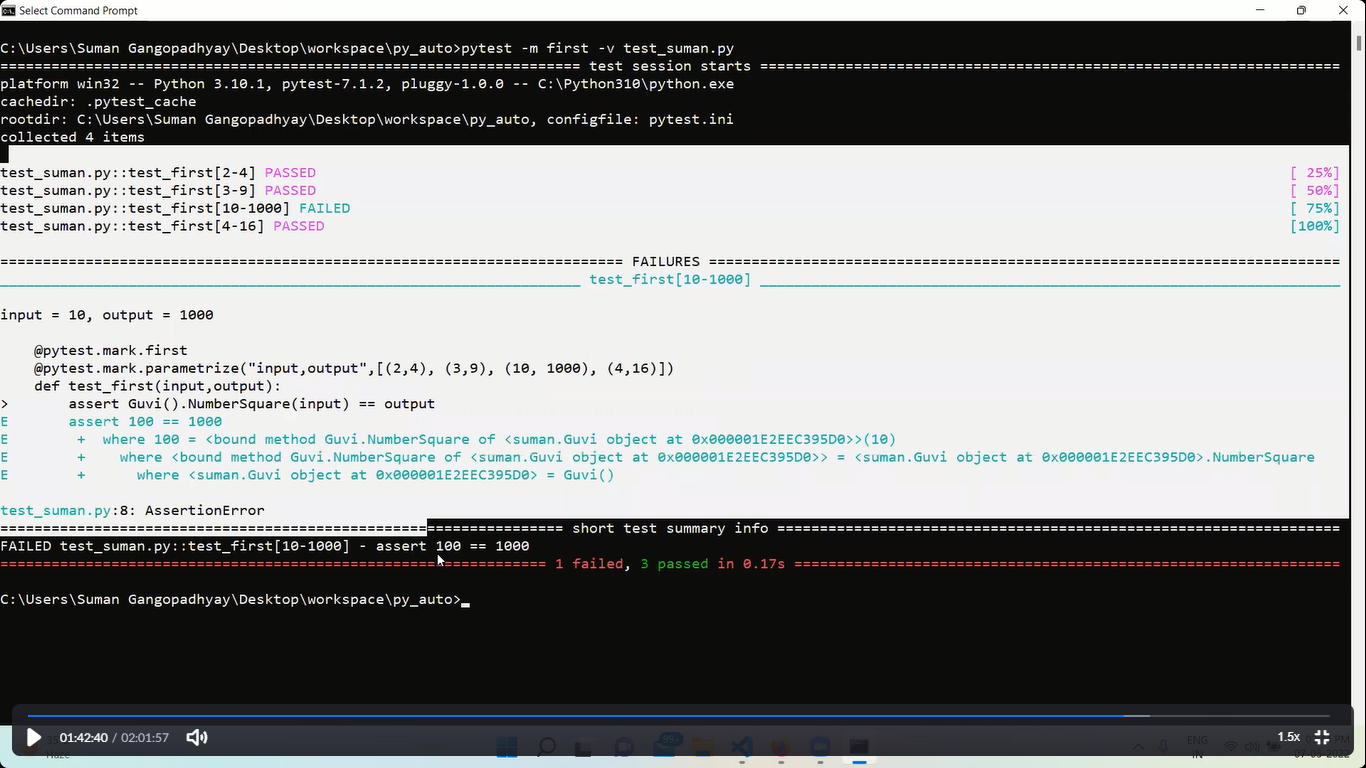
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

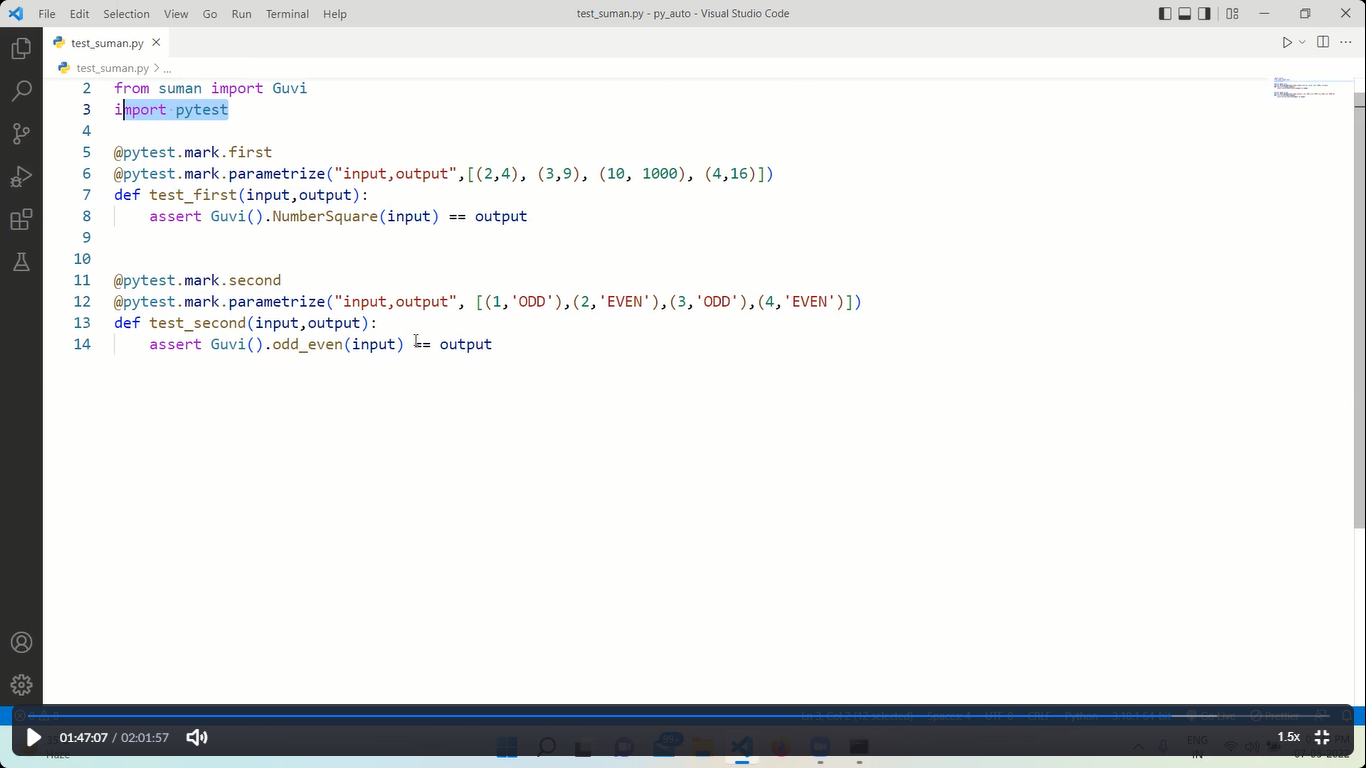
PARAMETERIZE :---

To run n number of test in a single function we need to parameterize it

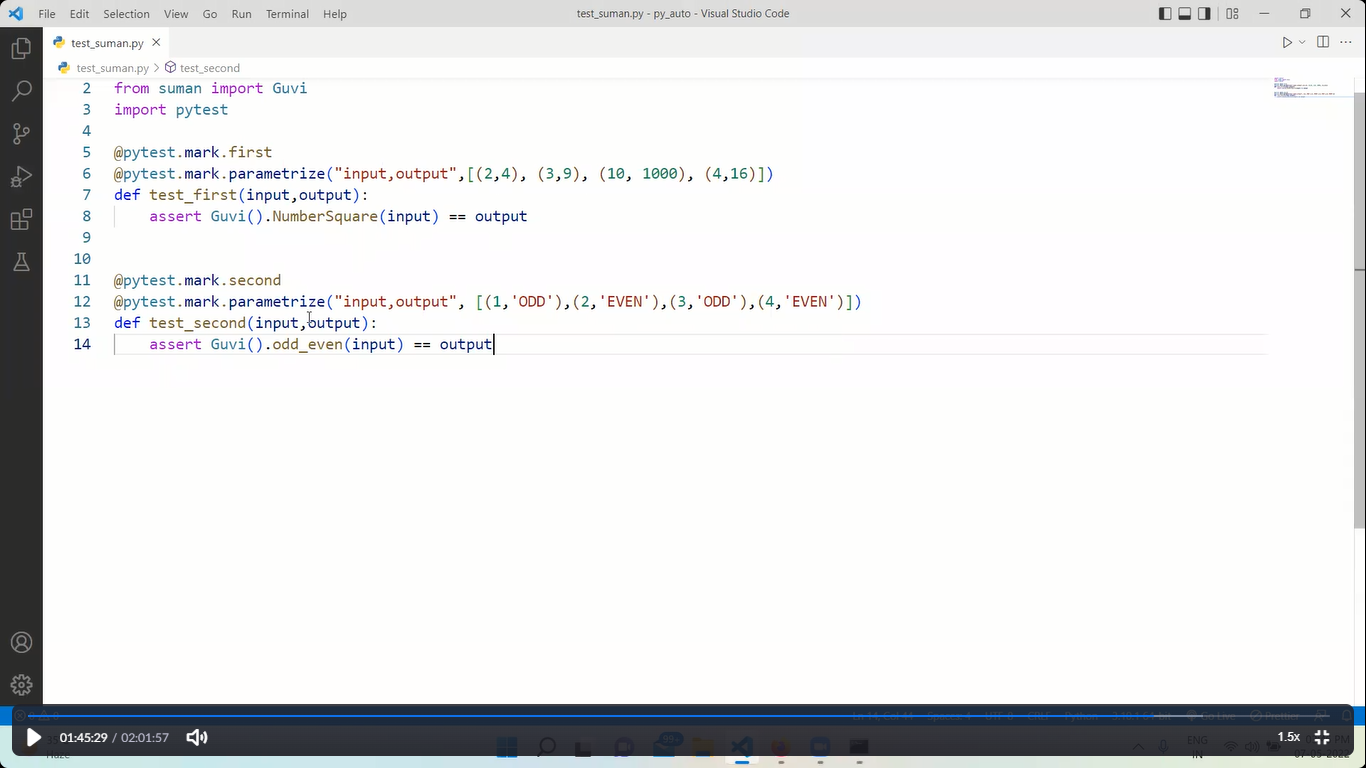
* A marker should be created
* Parameterize the marker
* Input & ouput || x & y || any variable is declared inside the parameterized
* In a list create a tuple & send the (input,output)
* 1st being the input & 2nd being the output
* Test function should be created & input , output should be passed as an arguments

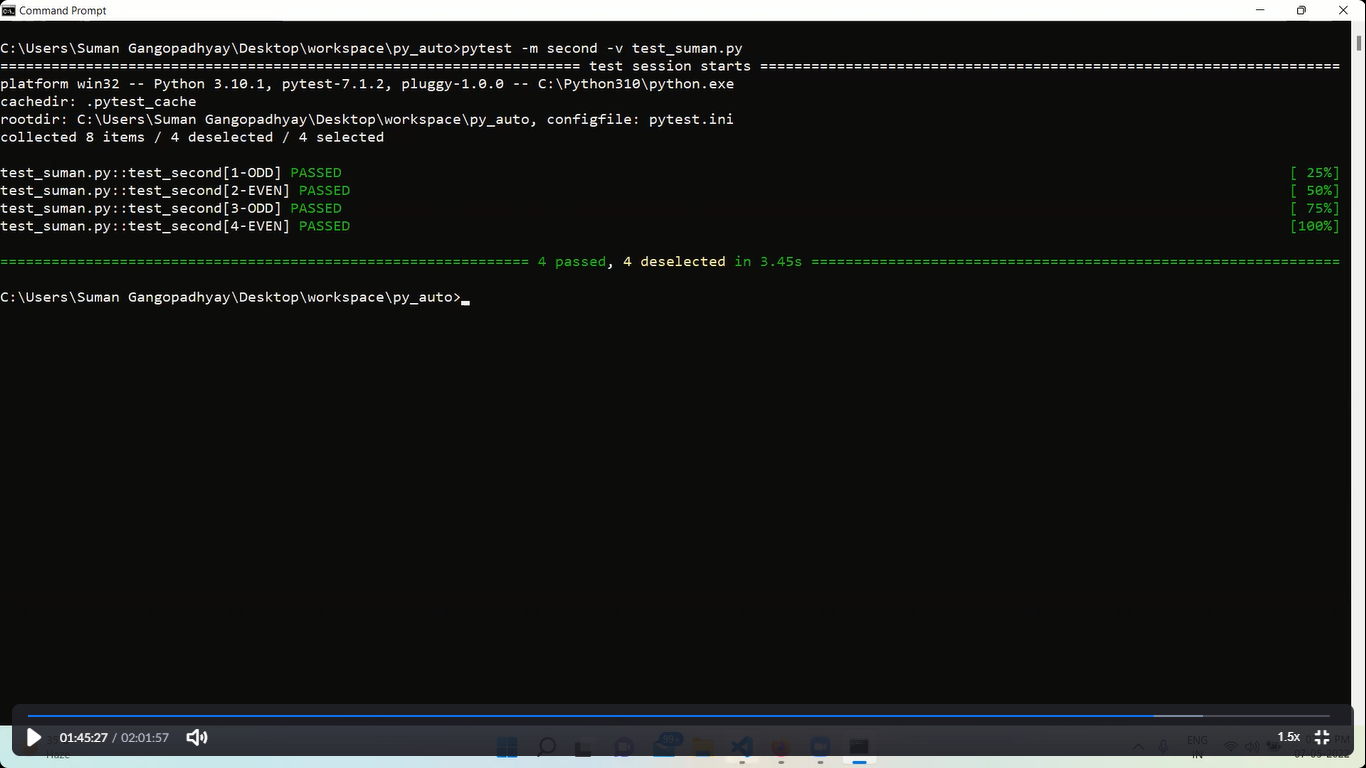






\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*





\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

GENERATING XML REPORT :--

Pytest –v test\_suman.py --junitxml=”report.xml” :-- command to generate the xml file for a python test file

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

CLASS TASK----------

1.) Use Python Selenium and Pytest for logging into your Instagram account finding your followersand generating a report for the same ?

2.) Use Python Selenium and Pytest for doing web-scrapping on Amazon and generating a report ?

Note # Kindly use external files for creating tests

**YOUTUBE PYTEST**

Pytest is a framework consist of classes

It is an open source – parallel mode – suite->set of testcases – pre conditions – post conditions – skip – fail – etc

pytest flags 🡪 search in google to get all flags present in the pytest

pytest --markers 🡪 search in cmd to get all the markers available in pytest

pytest --fixtures 🡪 search in cmd to get all the fixtures available in pytest

pytest <filename.py> -k <string> 🡪 command to run specific test contains the string in it from the test file

pytest <filename.py> -v 🡪 command to get detail report of the test runned i.e verbose

GENERATING REPORTS:--

XML REPORT :-- **pytest <test\_filename.py> --junitxml=”report.xml”**

HTML REPORT :-- initially we need to install **pip install pytest-html**

After installing **pytest <test\_filename.py> --html=Myhtmlreport.html**

ALLURE REPORT :--

**MARKERS**:--

pytest --markers 🡪 in cmd to get in built markers

pytest <test\_filename.py> -m <marker\_name>

pytest <test\_filename.py> -m “<marker\_1 or marker\_2>” 🡪 this command will run either marker1 or marker2. We can use “OR” “AND” etc

**pytest.ini** :-- is like a configuration file

all the markers which we created should be mentioned here

the pytest.ini file should be created in the same directory

[pytest]

markers =

marker1\_name : this is <smoke> scenario

marker2\_name : this is <regression> scenario

adopts = -rA --html=Automationreport.html # for inbuilt markers

**INBUILT MARKERS:--**

@pytest.mark.skip 🡪 this will skip the test for that function

@pytest.mark.skip

def test\_login():

    print("login done")

#skipif is used with condition as shown in below

@pytest.mark.skipif(sys.version\_info<(3,8),reason="python reason not supported")

def test\_addproduct():

    print("add product")

output:--

test\_markersdemo2.py::test\_login SKIPPED (unconditional skip) [ 33%]

test\_markersdemo2.py::test\_addproduct PASSED [ 66%]

test\_markersdemo2.py::test\_logout PASSED

#xfail is we expect the test to be failed

@pytest.mark.xfail

def test\_logout():

    assert True

    print("logout done")

collected 3 items

test\_markersdemo2.py::test\_login SKIPPED (unconditional skip) [ 33%]

test\_markersdemo2.py::test\_addproduct PASSED [ 66%]

test\_markersdemo2.py::test\_logout XPASS [100%]

#xfail is we expect the test to be failed

@pytest.mark.xfail

def test\_logout():

    assert False

    print("logout done")

collected 3 items

test\_markersdemo2.py::test\_login SKIPPED (unconditional skip) [ 33%]

test\_markersdemo2.py::test\_addproduct PASSED [ 66%]

test\_markersdemo2.py::test\_logout XFAIL [100%]

**PARAMETERIZE** :--

List of tuple as input

@pytest.mark.parametrize("username,password",

    [

        ("selenium","webdriver")

        ("python","pytest")

        ("santosh","otwani")

    ]

)

def test\_login(username,password):

    print(username)

    print(password)

**FIXTURES :--**

A test can be broken in to 4 parts

1. Arrange
2. Act
3. Assert
4. Cleanup

In general the process is like

def test\_1():

    print("start the browser")

    print("test 1 executed")

    print("close the browser")

def test\_2():

    print("start the browser")

    print("test 1 executed")

    print("close the browser")

def test\_3():

    print("start the browser")

    print("test 1 executed")

    print("close the browser")

but we can change this using fixtures as shown in below:-

import pytest

@pytest.fixture

def setup():

    print("start browser")

def test\_1(setup):

    print("test 1 executed")

    print("close the browser")

def test\_2(setup):

    print("test 1 executed")

    print("close the browser")

def test\_3(setup):

    print("test 1 executed")

    print("close the browser")

YIELD:--

Any tear down code for that fixture is placed after the yield

return is swapped out for yield

@pytest.fixture

def setup():

    print("start browser")

    yield

    print("close the browser")

def test\_1(setup):

    print("test 1 executed")

def test\_2(setup):

    print("test 1 executed")

def test\_3(setup):

    print("test 1 executed")

pytest <test\_filename> -s 🡪 command for in casse the yield is not displaying the result

fixtures example:--

from lib2to3.pgen2 import driver

import pytest

from webdriver\_manager.chrome import ChromeDriverManger

driver=None

@pytest.fixture

def setup():

    print("start browser")

    global driver

    driver = webdriver.Chrome(ChromeDriverManager().install())

    driver.maximize\_window()

    yield

    print("close the browser")

def test\_1(setup):

    driver.get("https://www.facebook.com")

    print("test 1 executed")

def test\_2(setup):

    driver.get("https://www.gmail.com")

    print("test 1 executed")

def test\_3(setup):

    driver.get("https://www.instagram.com")

    print("test 1 executed")

**to run the tests in parallel mode then** :---

1st install the package in cmd --- **pip install pytest-xdist**

2nd --- **pytest <test\_filename.py> -n 3** # here 3 are tests in our code

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*