# Pytest:

Why framework:

* Structure our test
* Added helpful features

Pytest framework:

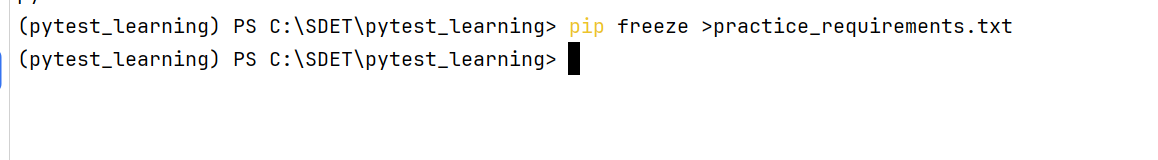
* Test framework based on python language
* Easy to write , execute and generate test reports
* Different types and level of testing

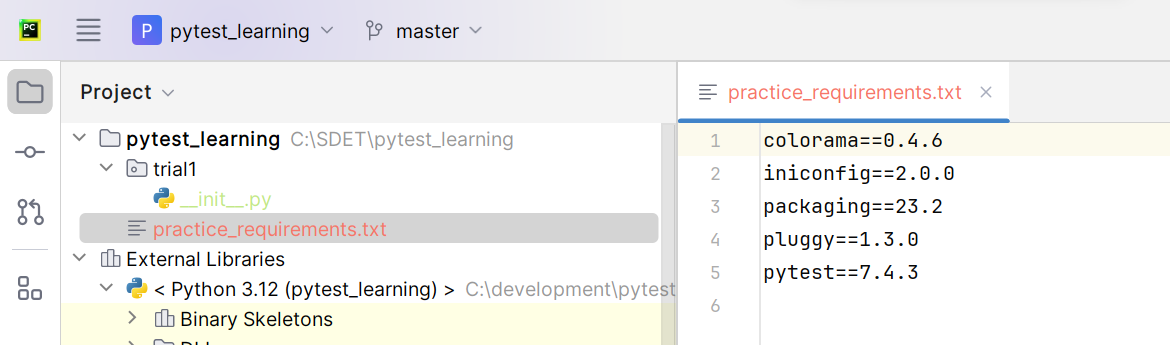
**Good:**

* Ease of use
* Auto detect tests
* Grouping and marking tests
* Uses pythons assert keyword
* Good features – fixtures, parameterization
* Good community support
* CI/CD integration

Pytest command to generate a requirements file which all packages/modules installed in our project

pytest freeze >file\_name.txt





* This one we used to do whenever we want to share our project with others
* Recivers after getting our project if they want to install all packages which is in out requirements file (generated using freeze command) they should use below command
* Pip install –r file\_name

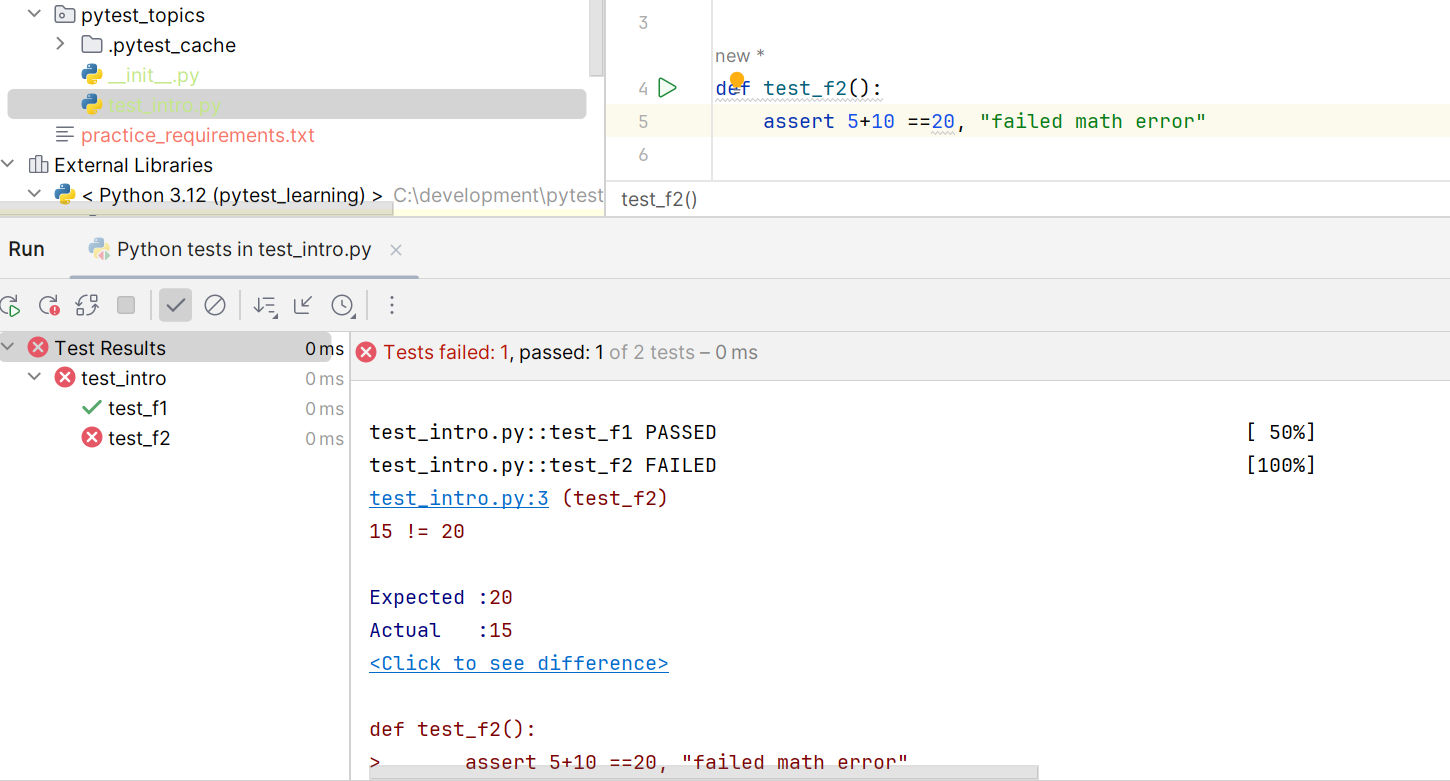
# File creation:

* While creating test file we should follow naming convention like either file shoukd start by test\_ or end by \_test
* Syntax: test\_<somename>.py

**Note: Pytest is going to search for the test files, in current directory and sub directories and run those test files**

**Note:Even the class and test methods should be named starting with test**

**Printing some failed text if its fails:**

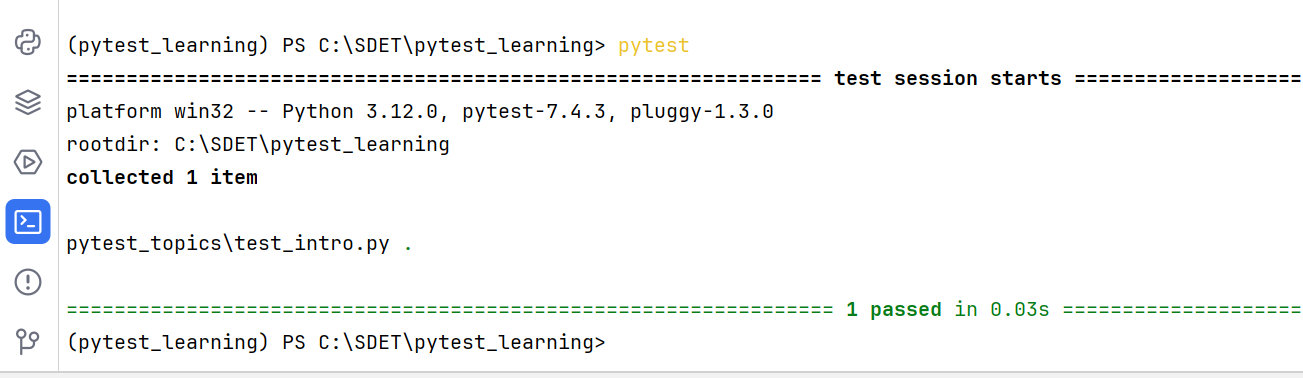
****

# Running tests:

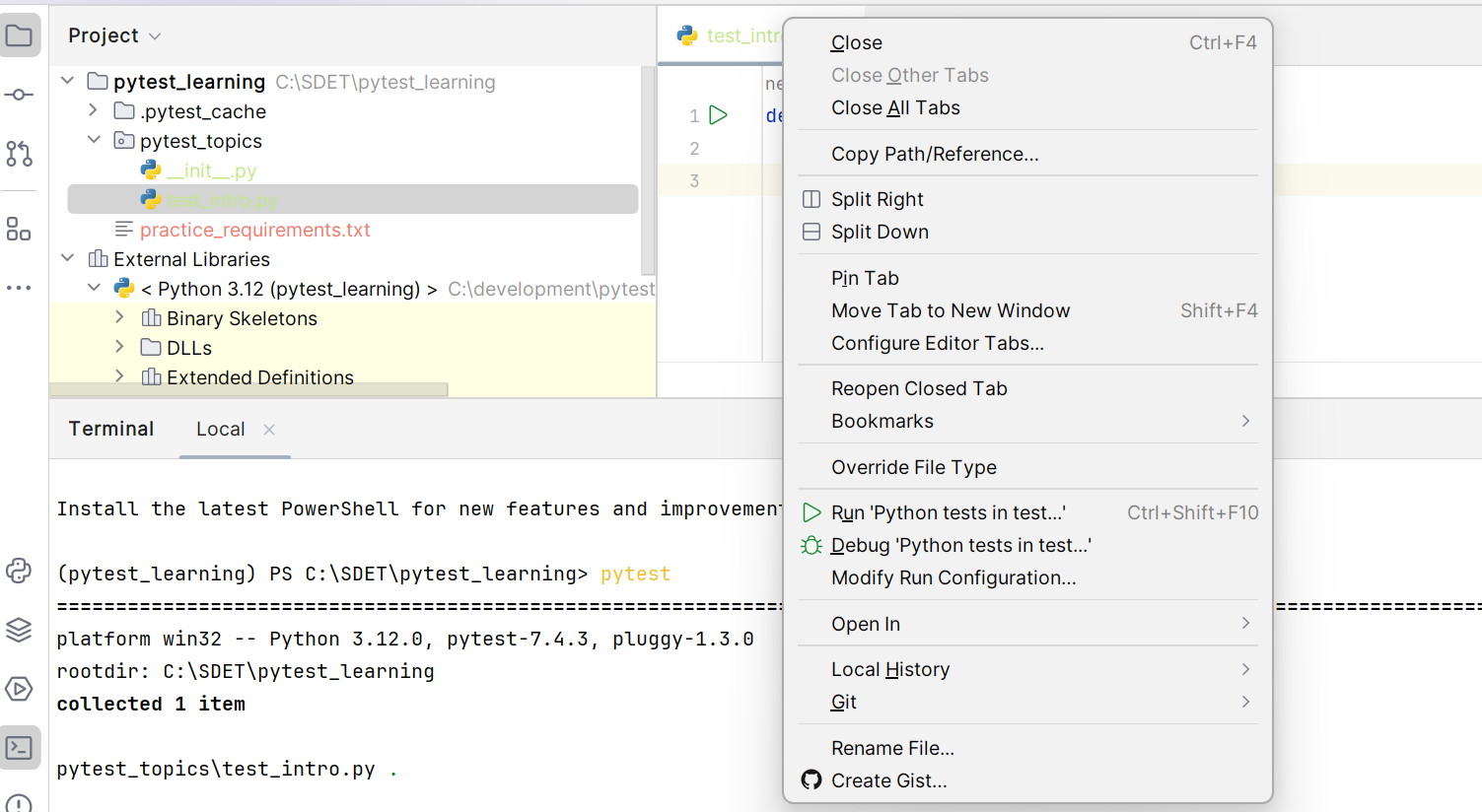
## Running test file:

* We can run the file from terminal
* We can run the file by right clicking on it
* We can run it from python console / command prompt

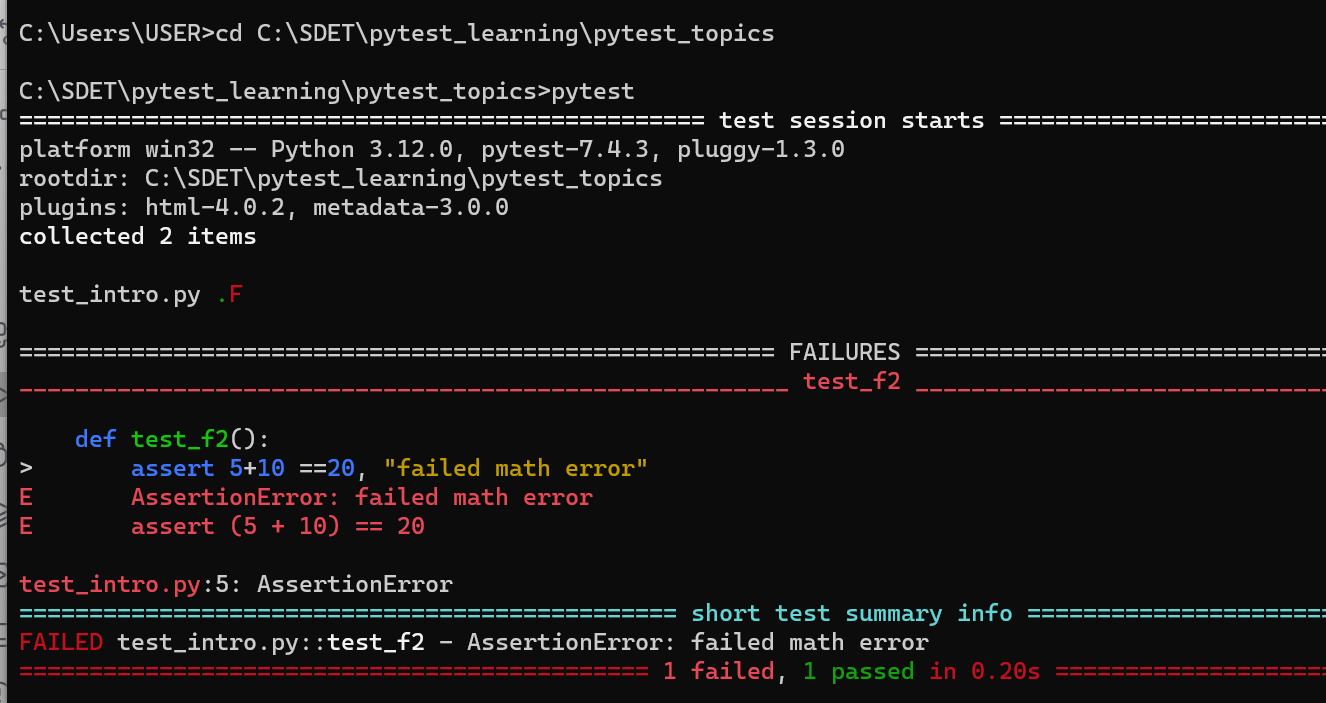
## From terminal:

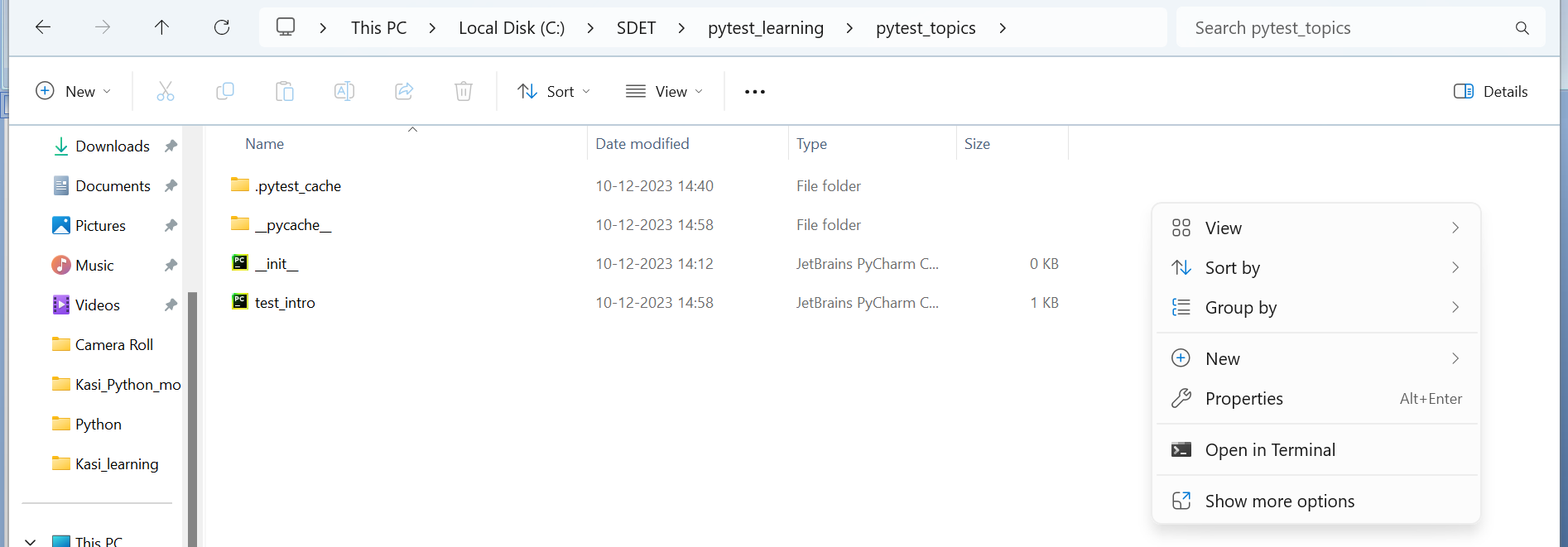
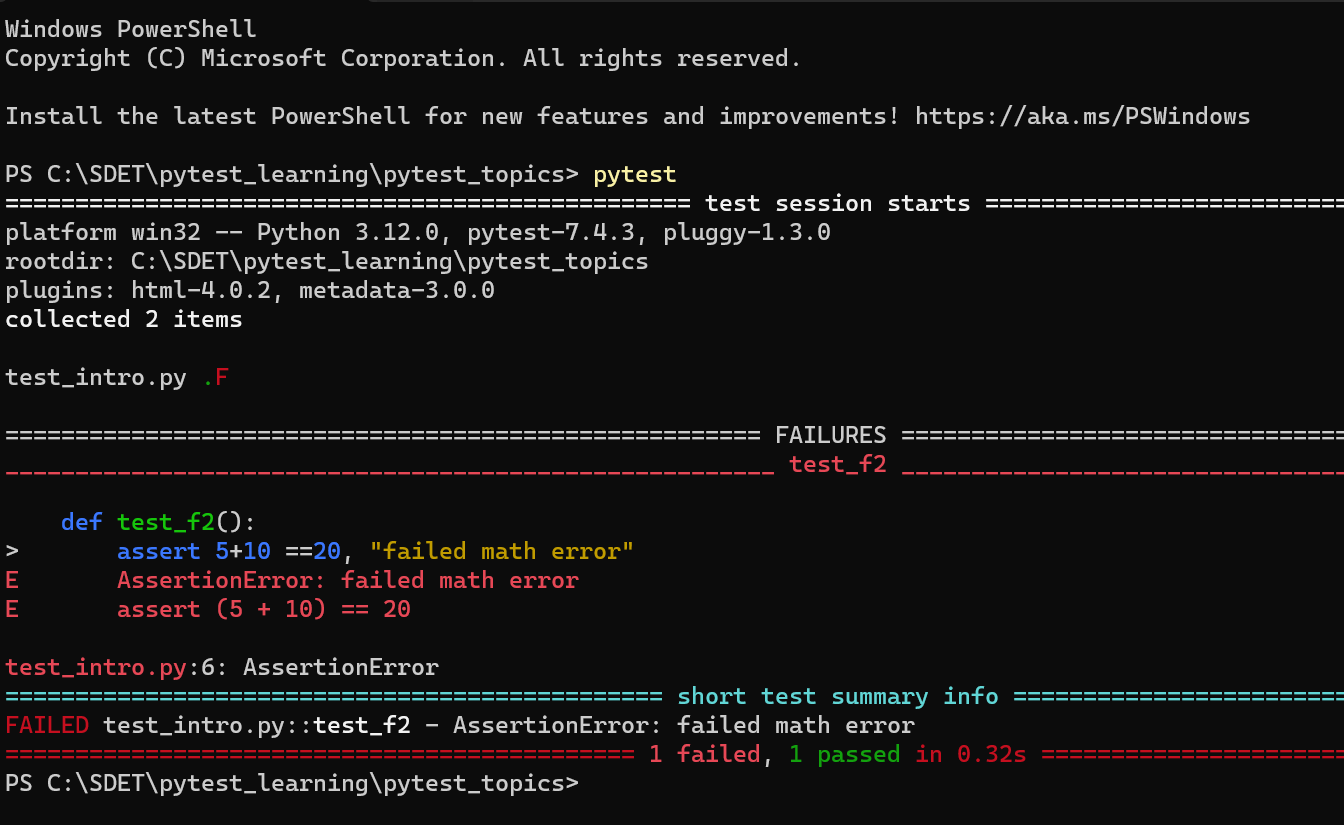


## Right clicking on file:



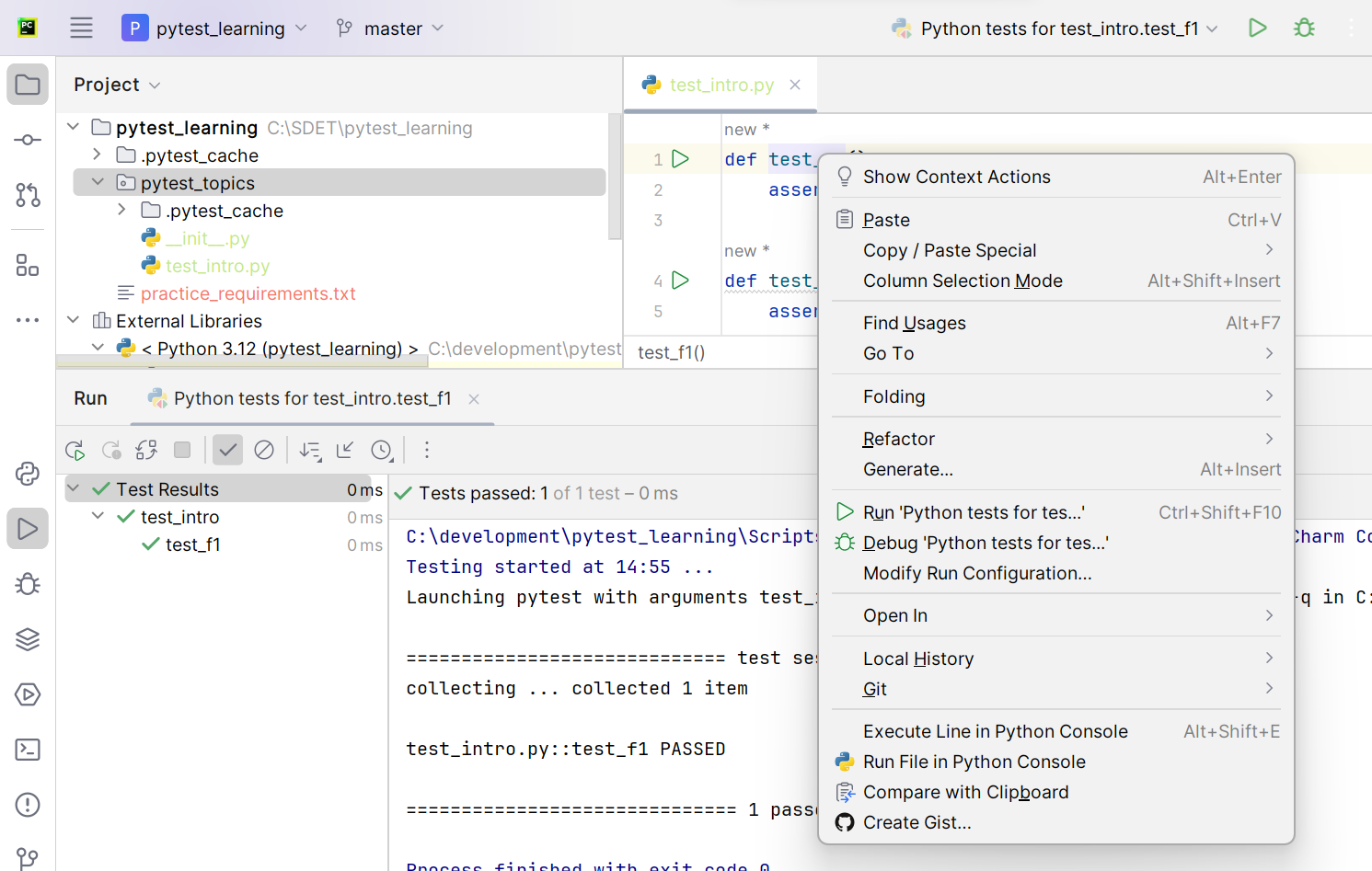
## Running from cmd:



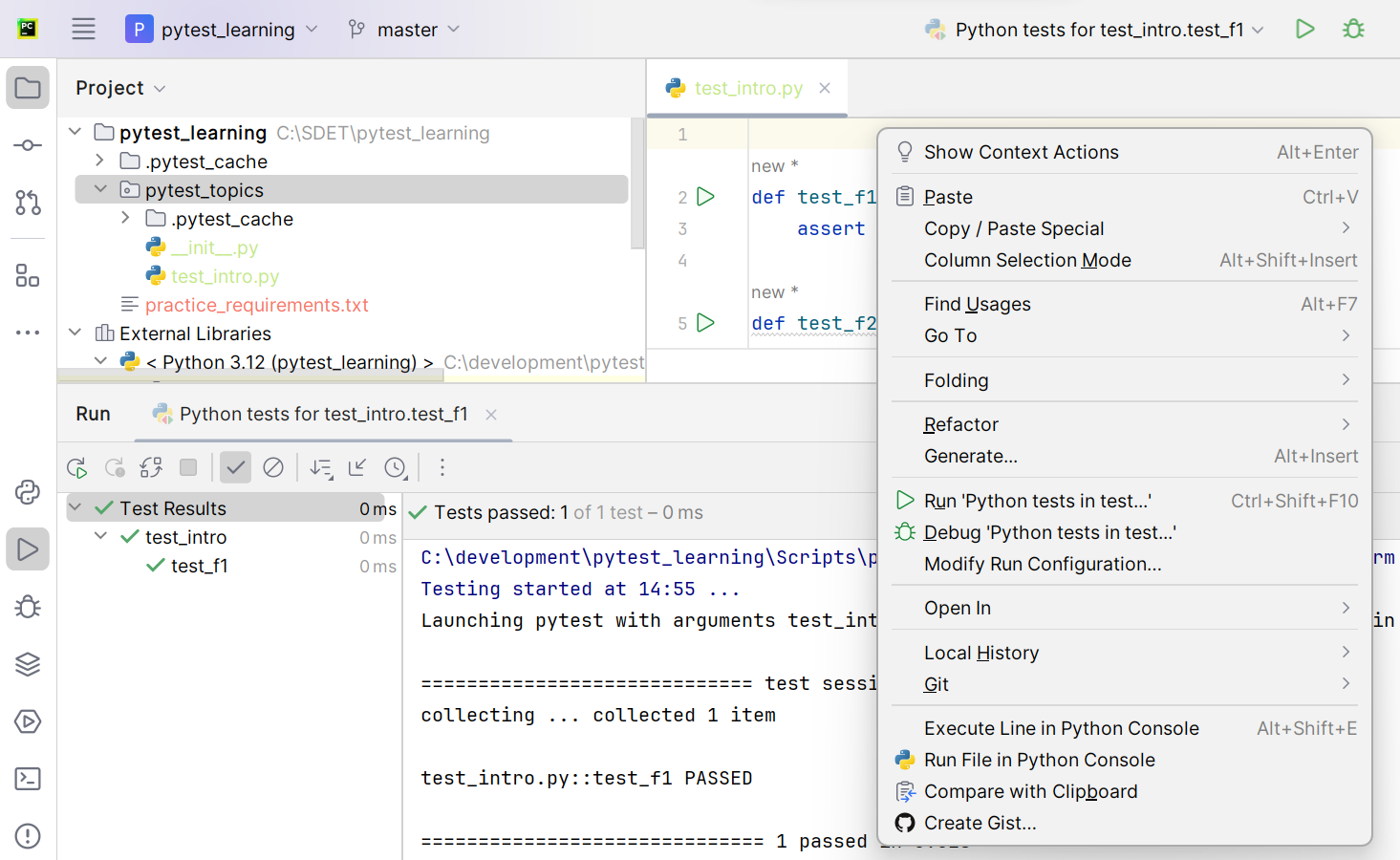
* Instead of cd changing to file directory, you can do the same by going to the project directory in file explorer and right click and open cmd
* 
* 

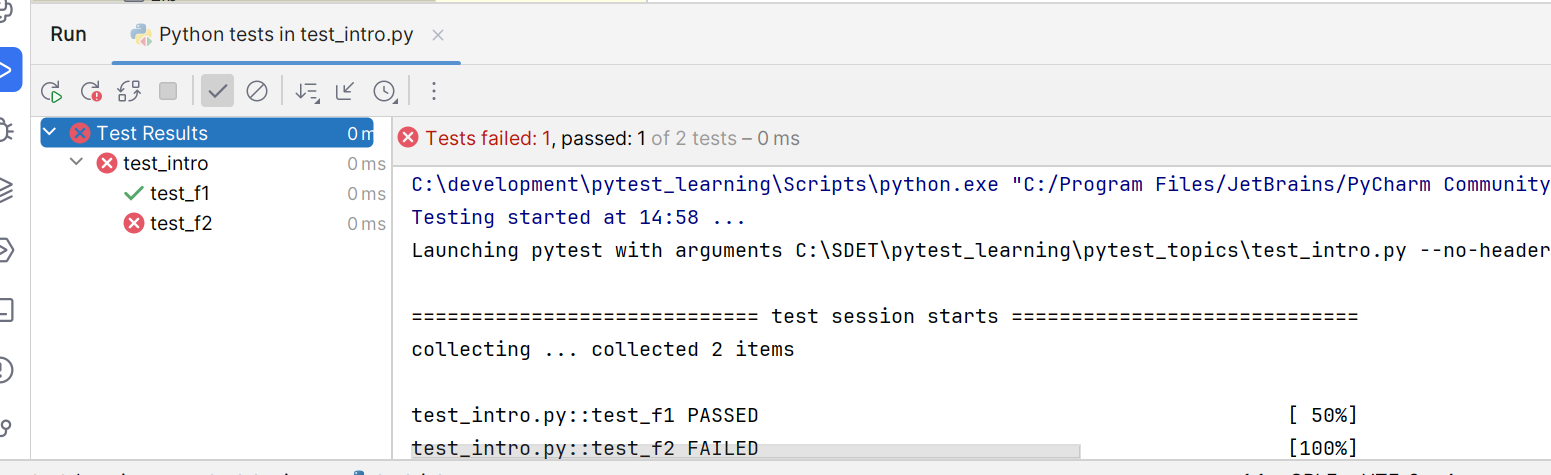
**Note: Good test practice is should not to add multiple asserts in one function/method**

**If you want to run particular test method in a file you can go to that method and right click on method by putting cursor there, it will run only that method**

****

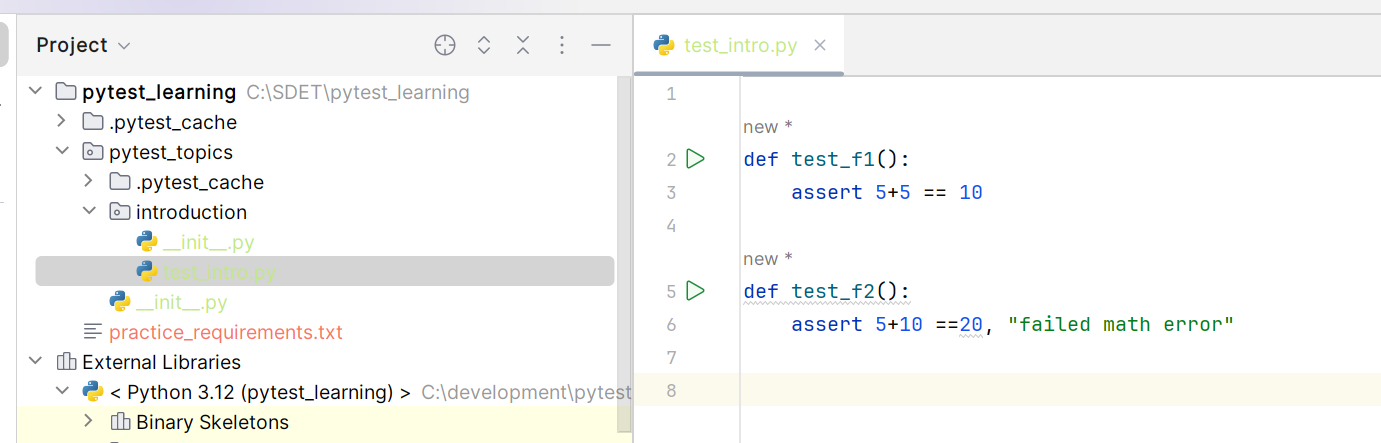
**If you want to run all methods, put the cursor at the top of file and right click and run**

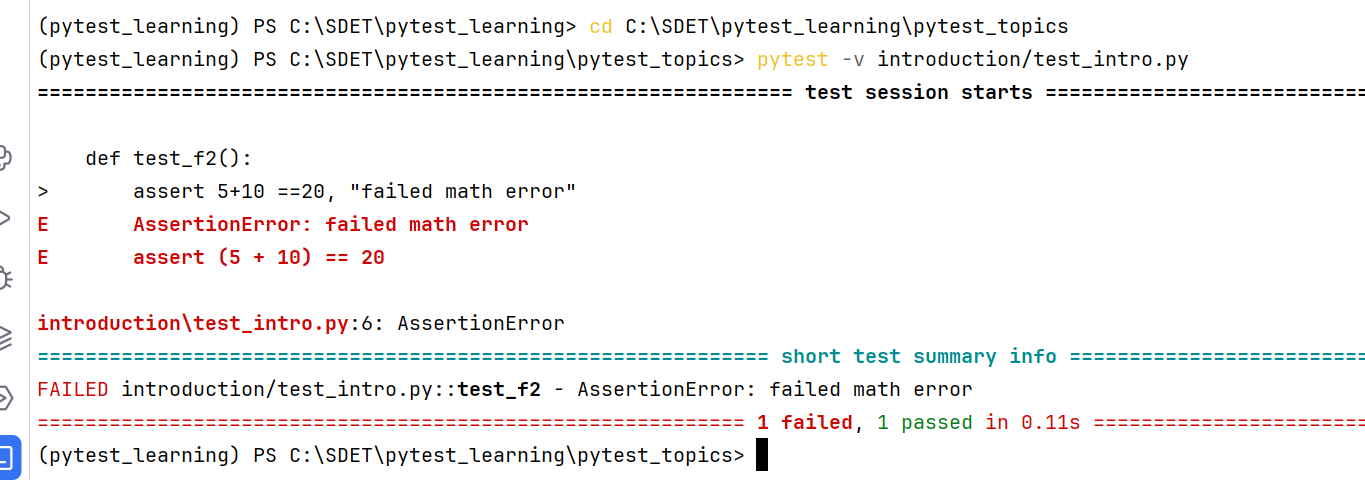
****

****

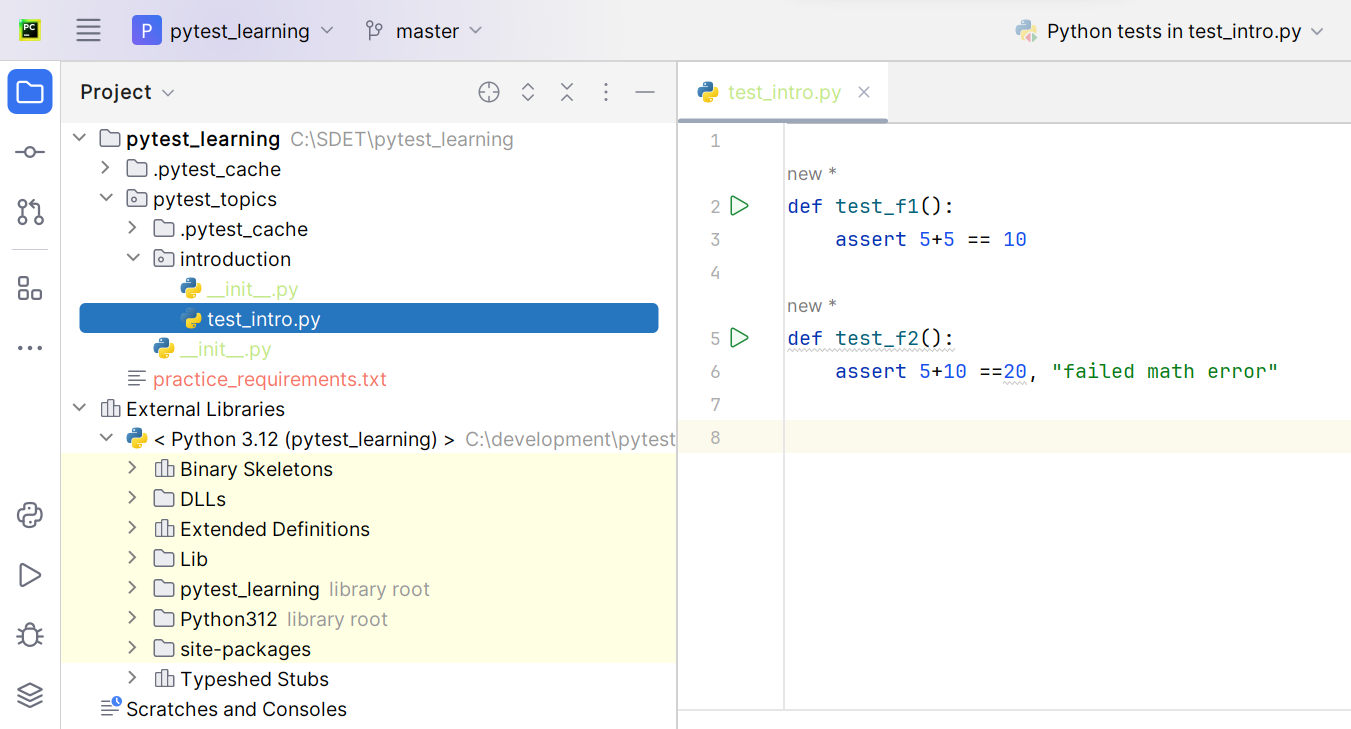
**Running particular module/particular function/particular class/particular method:**

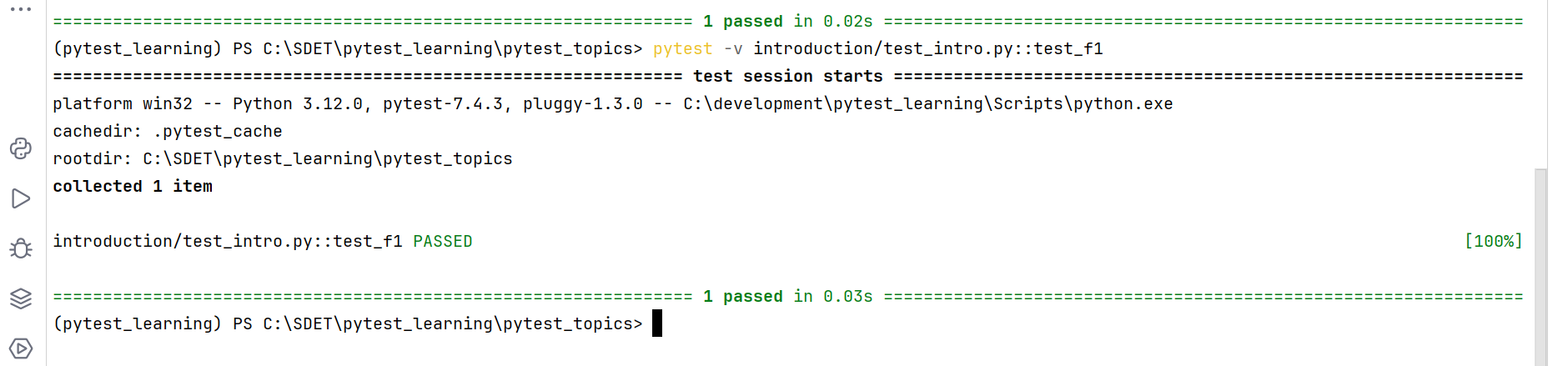
## Running only one module:

****

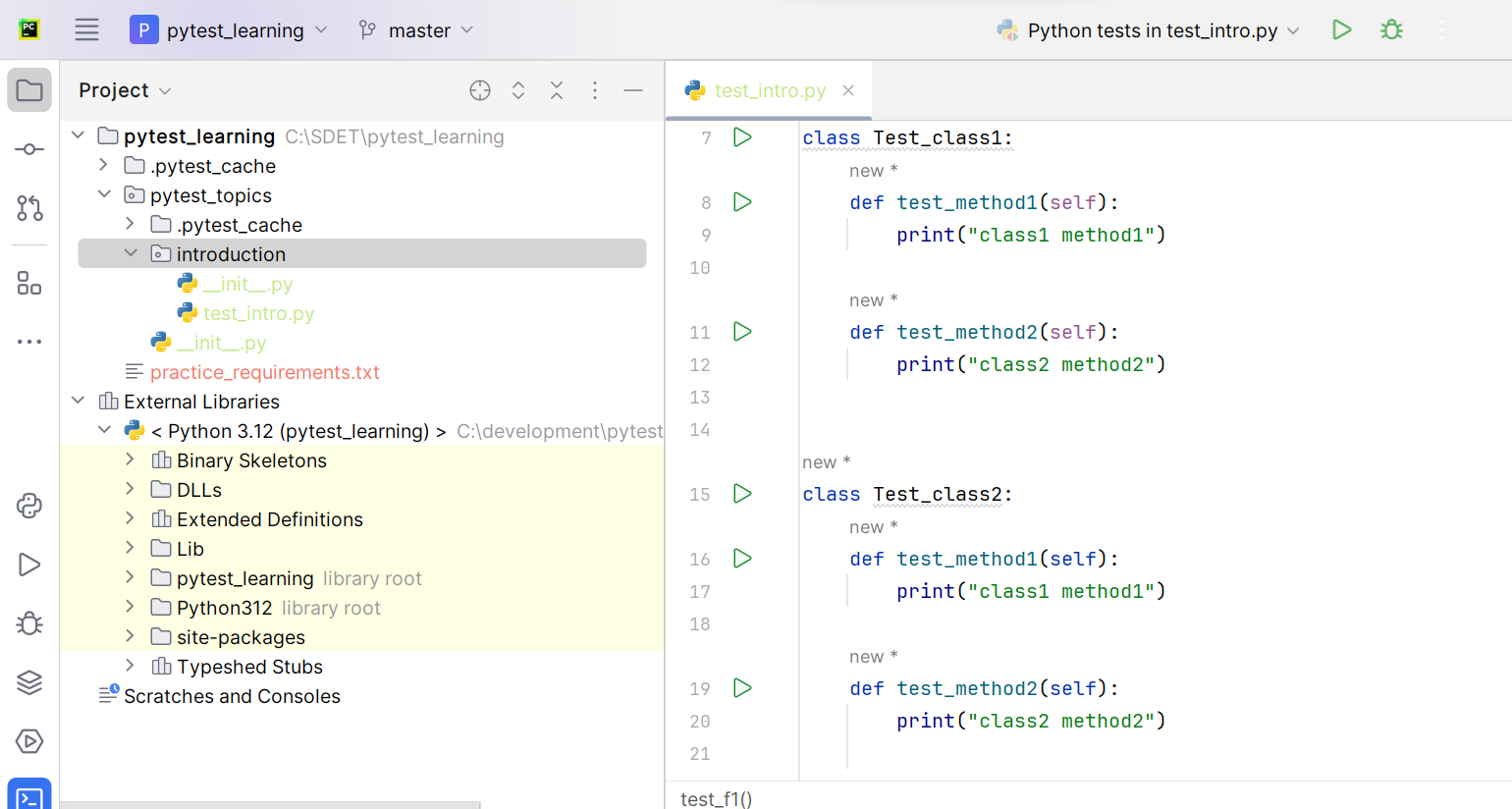
****

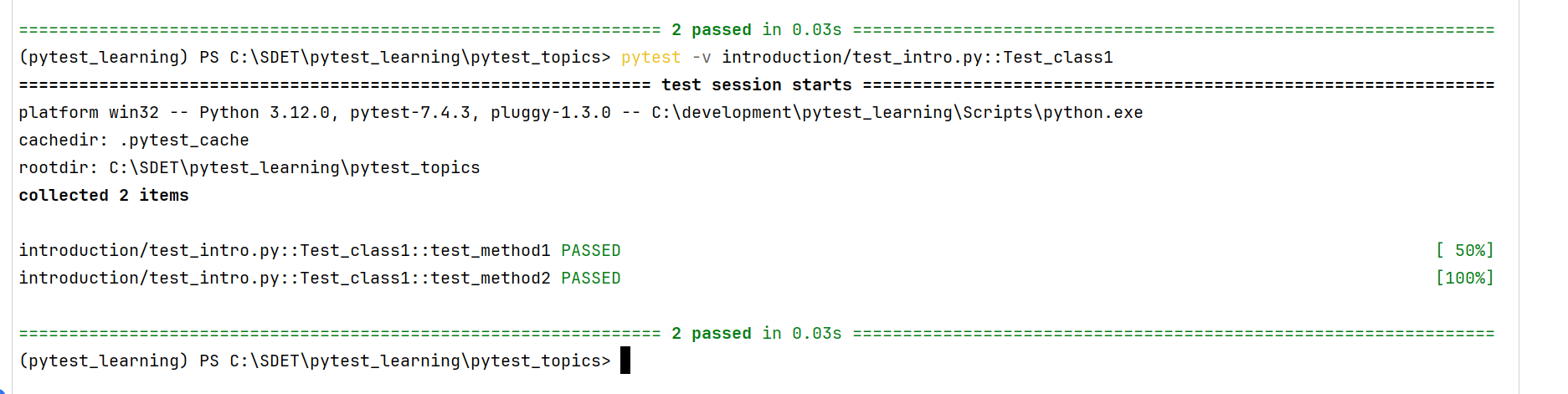
## Running only one function in a module:

****

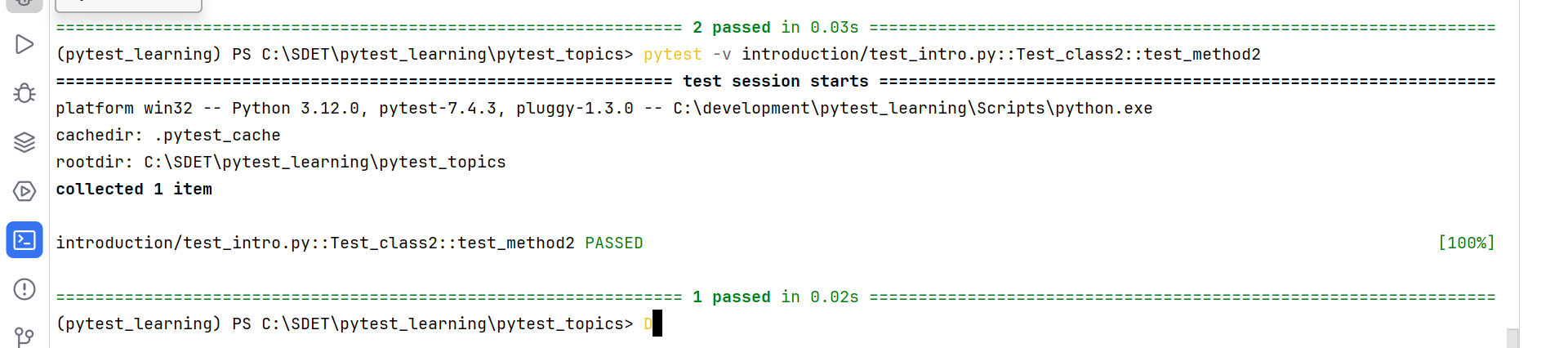
****

## Running only one class in a module:

****

****

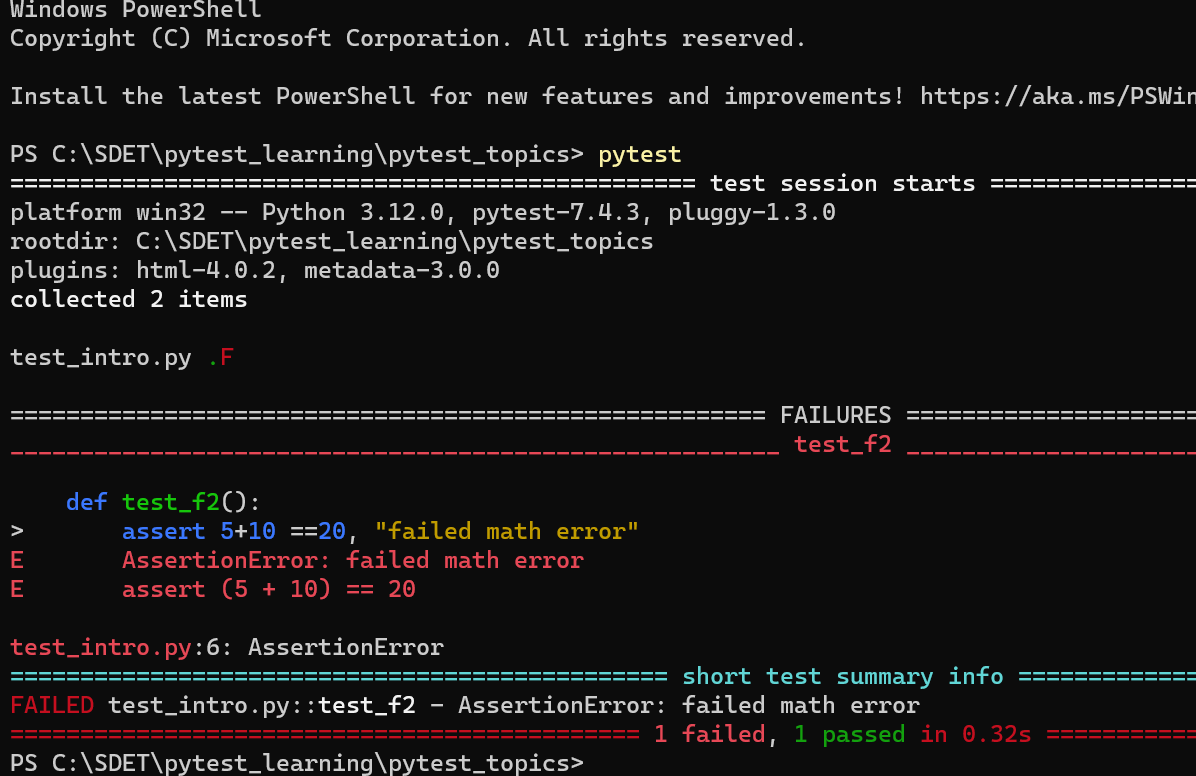
## Running a particular method in a class:

****

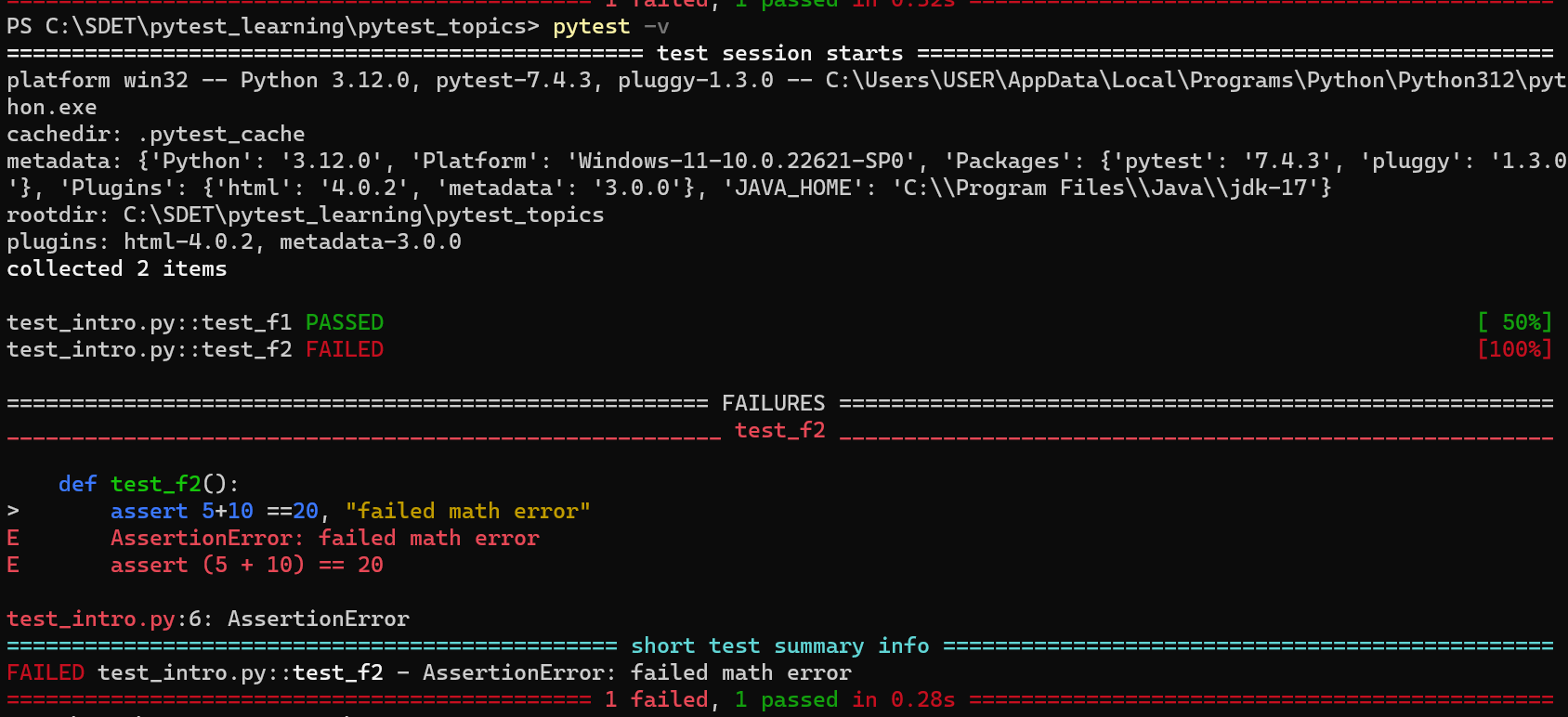
**///////////////////////////////////////////////////////////////////////////////////////////**

**Verbose mode: pytest –v**

**Without –v:**

****

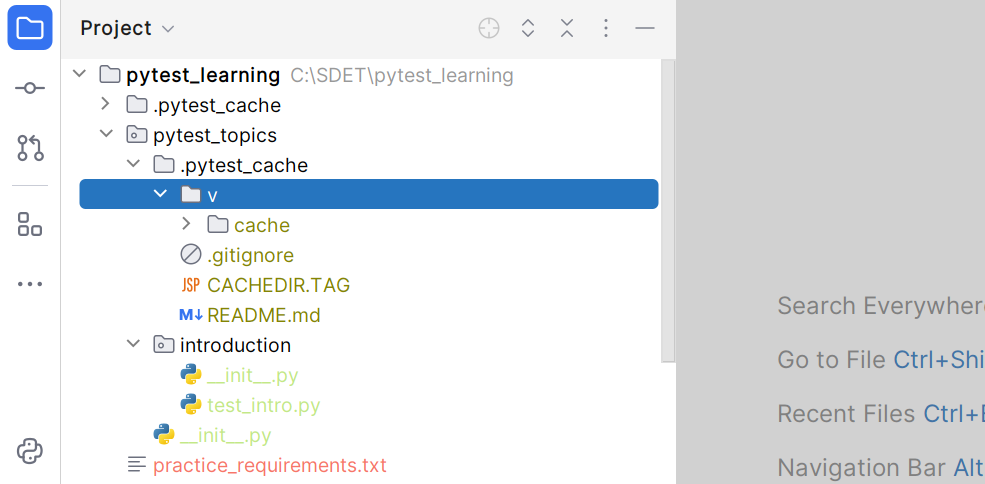
**With –v:**

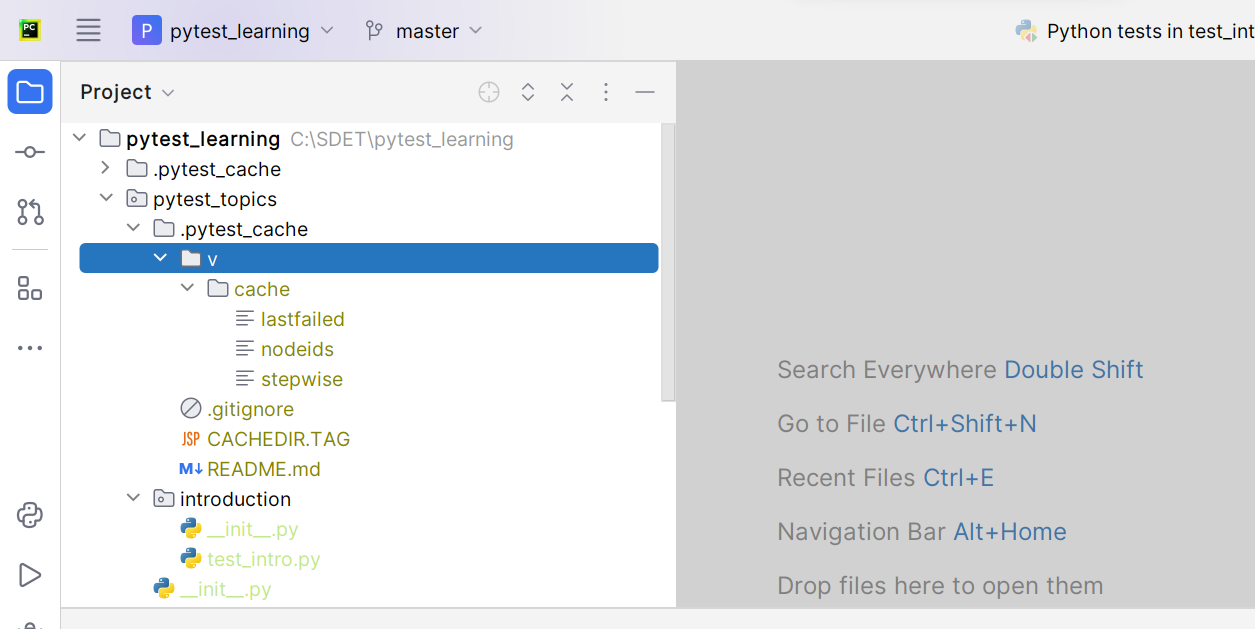
****

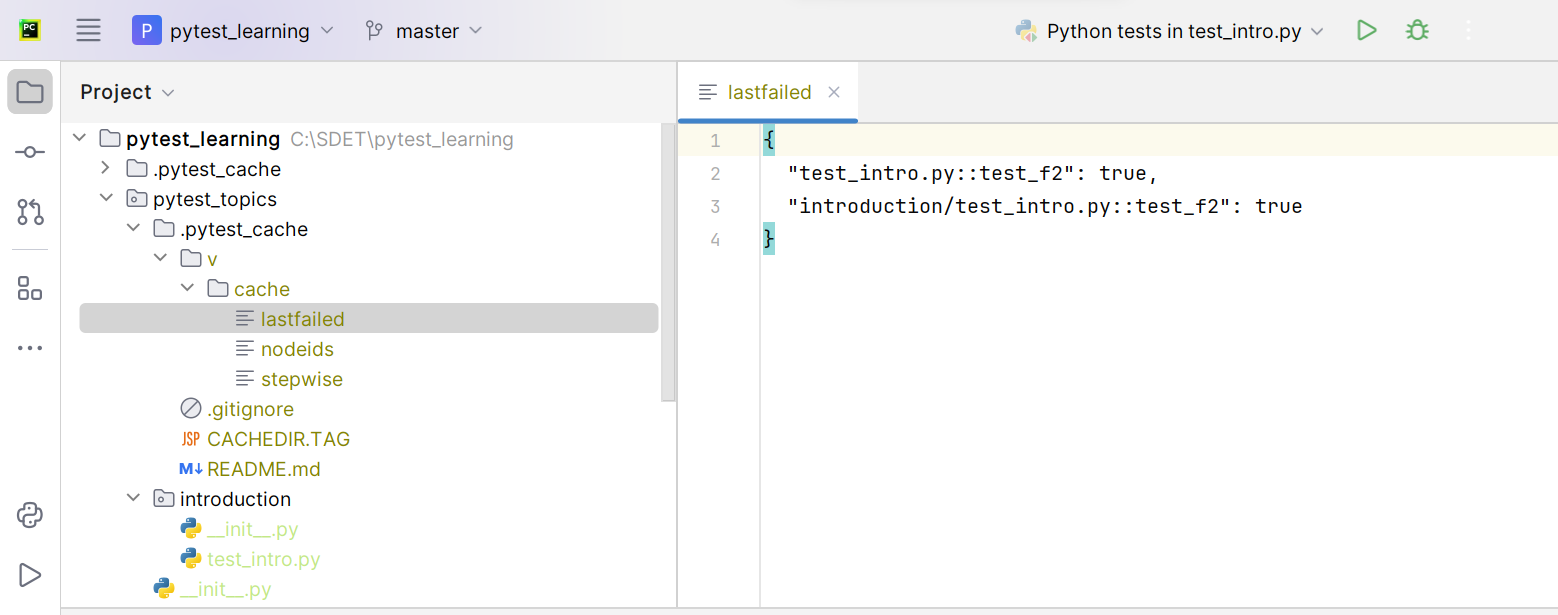
**Verbose mode provides proper clarity on test execution like which is passed and failed**

# Pytest cache:

* Cache stores the information about the last run

****

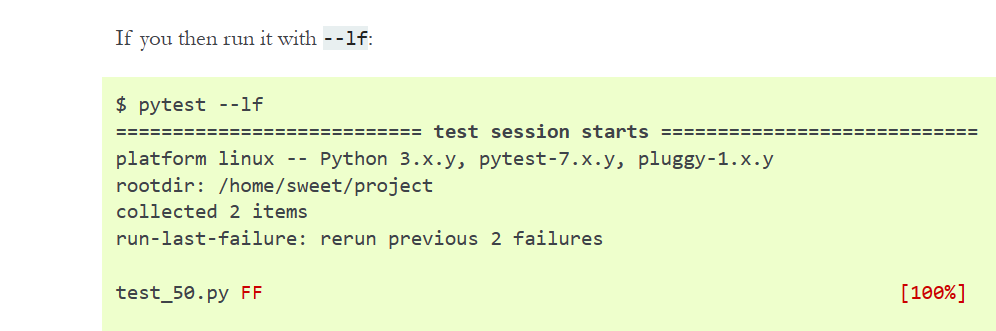
****

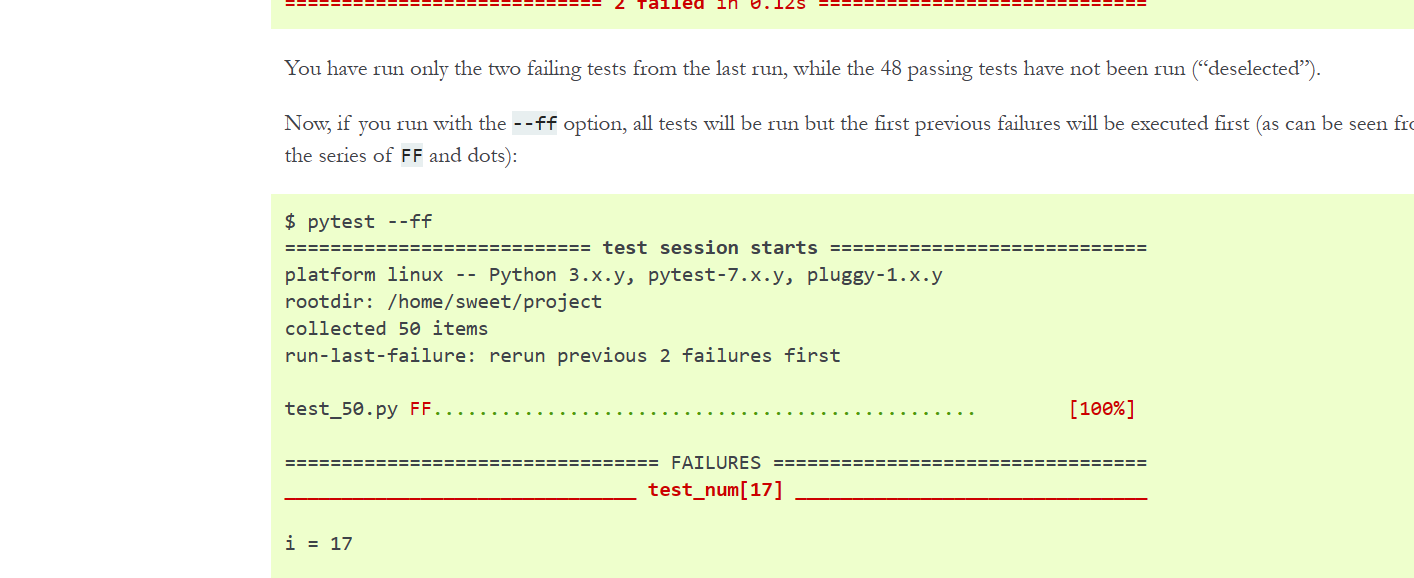
****

* **Here in lastfailed file the information about the last failed script is stored**

****

**When you run the –lf or –ff it will take the failed files from the lastfailed file pytestcache**

****

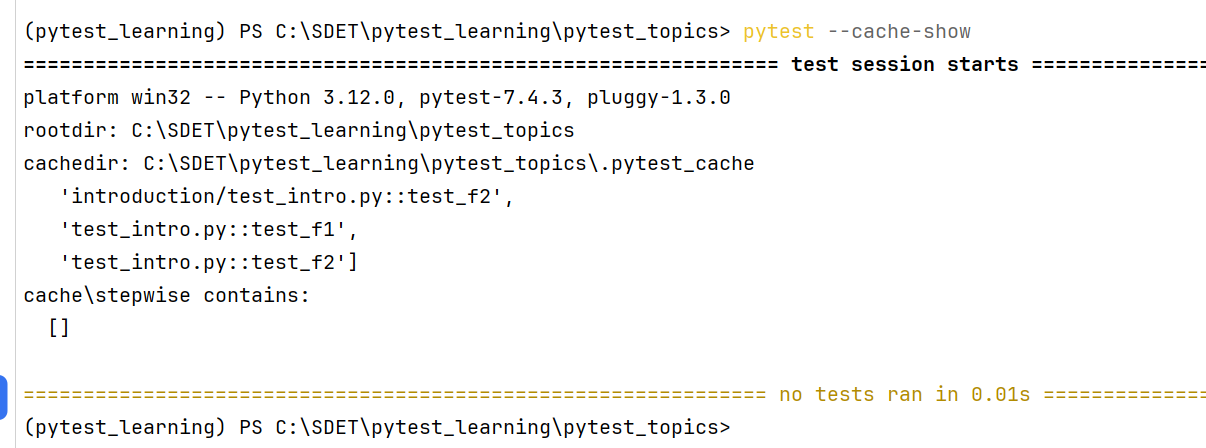
**’**

**--lf --last-failed 🡪 it will run the only last failed files**

**--ff --failed-first 🡪 it will run the failed first and remaining files later**

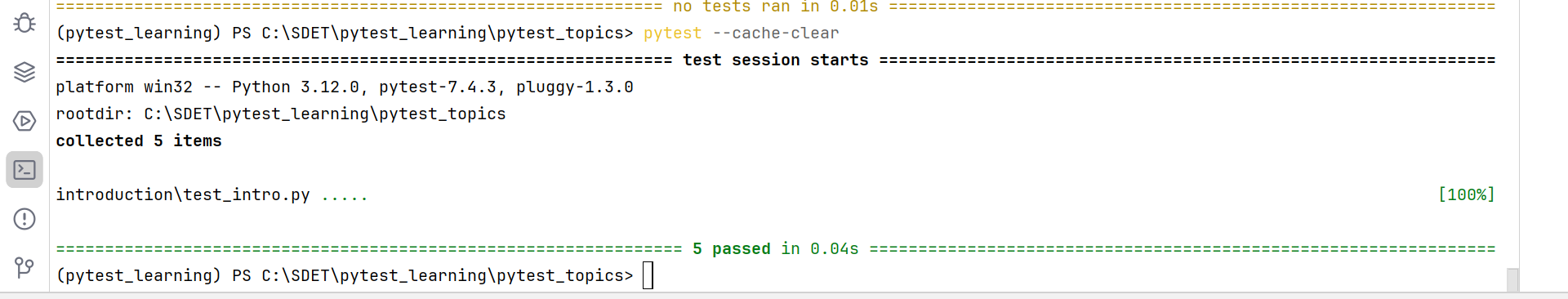
## If you want to see the pytest cache from the terminal instead of from folders:

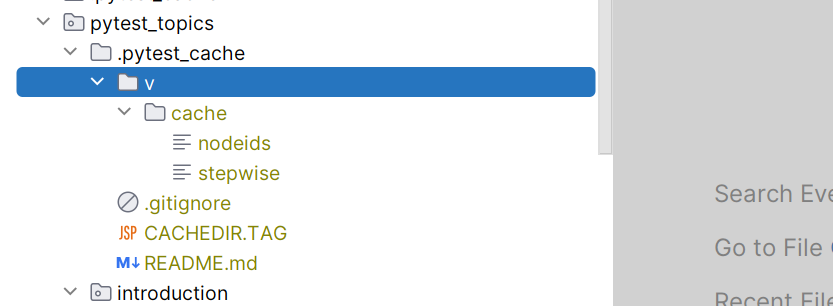
**Pytest --cache-show**

****

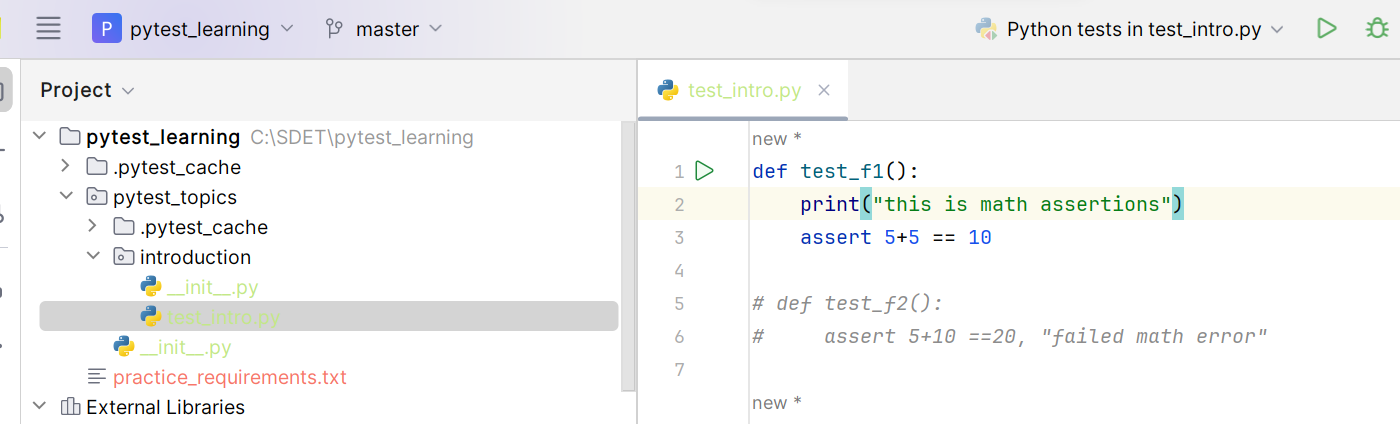
**Clearing cache:**

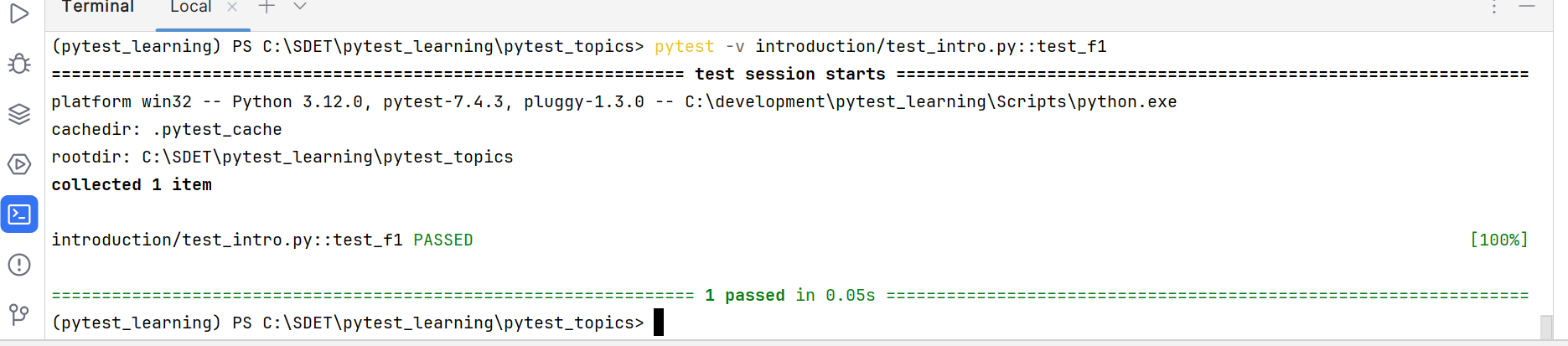
**Pytest –cache-clear**

****

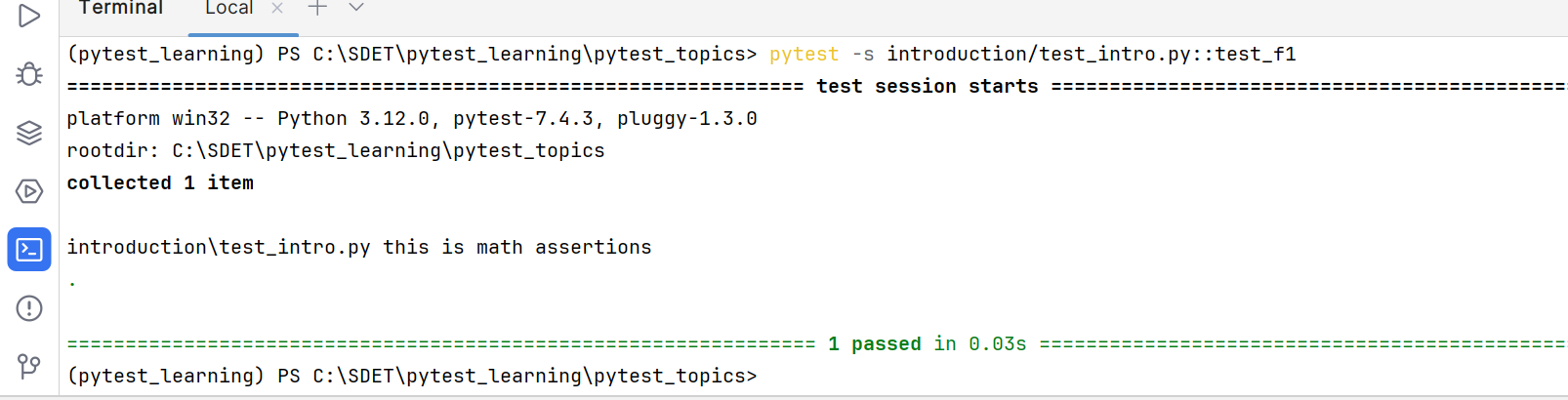
****

**Without –s in pytest command:**

****

****

**With –s in pytest commands:**

****

**Note: you can see the difference like –s printing print statements in terminal outputs**

# Assertions:

For assertions operation in python we do have assert keyword

## assert:

In Python, the assert statement is used to continue the execute if the given condition evaluates true, if the given condition evaluates to False, then it raises the Assertion Error exception with the specified messege.

Syntax: assert condition [, “Error messege”]

Examples:

def test\_a1():  
 assert 5+5 == 10  
  
string1 = 'Mohana'  
def test\_f2():  
 assert string1.lower() == 'mohana'  
  
def test\_f3():  
 assert 'i' in 'kasi'  
  
def test\_f4():  
 assert 1,2 in [1,3,4,2]  
  
def test\_f5():  
 assert 4<5

# Test class:

class Test\_asserts:  
  
 def test\_type(self):  
 assert type(1) == int  
  
 def test\_str(self):  
 assert 'mohana'.capitalize() == 'Mohana'

\_\_init\_\_.py: It describes the desired folder is a python package and allows us to have duplicate files with same names

Assertion handling:

import pytest  
string\_ = 'Mohana'  
def test\_1():  
 with pytest.raises(IndexError):  
 print(string\_[10]) *#as it gives index error*def test\_2():  
 with pytest.raises(ZeroDivisionError):  
 assert 1/0  
def test\_3():  
 with pytest.raises(TypeError) as set\_key\_error:  
 set\_ = {1,45, 'kasi'}  
 print(set\_[2])  
 print(str(set\_key\_error)) *#prints the exception messege*

skipping tests:

import pytest  
from pytest import mark  
import sys  
const = 9/5  
def cen\_to\_fh(cent=0):  
 fah = (cent\*const) + 32  
 return fah  
  
"mark.skip will simply skips the test method without chekcing any condition"  
@mark.skip("skiping typecheck as already know") *#skip the test & returns the messege*def test\_type\_simple\_skip():  
 assert type(cen\_to\_fh()) == float  
  
"skipif checks for a condition and returns the mentioned reason"  
@mark.skipif(sys.version\_info < (3,10), reason='chekcing skip if')  
def test\_validate1():  
 assert cen\_to\_fh() == 32  
  
def test\_validate2():  
 assert cen\_to\_fh(38) == 100.4  
  
"""examples of skipif with sys module"""  
@mark.skipif(sys.version\_info > (3,2), reason='chekcing skip if')  
def test\_validate\_version():  
 assert cen\_to\_fh() == 32  
  
@mark.skipif(100>1, reason="checking skipif")  
def test\_skip\_condition():  
 assert cen\_to\_fh(100) == 212  
  
  
@mark.skipif(sys.platform == 'win32', reason="checking skipif")  
def test\_validate\_platoform():  
 assert cen\_to\_fh(100) == 212  
  
@mark.skipif(pytest.\_\_version\_\_ < '6.4.6', reason='checking if with pytest version')  
def test\_pytest\_version\_less\_then\_check():  
 assert cen\_to\_fh(140) == 284  
  
@mark.skipif(pytest.\_\_version\_\_ > '7.3.2', reason='checking if with pytest version')  
def test\_pytest\_version\_check():  
 assert cen\_to\_fh(140) == 284