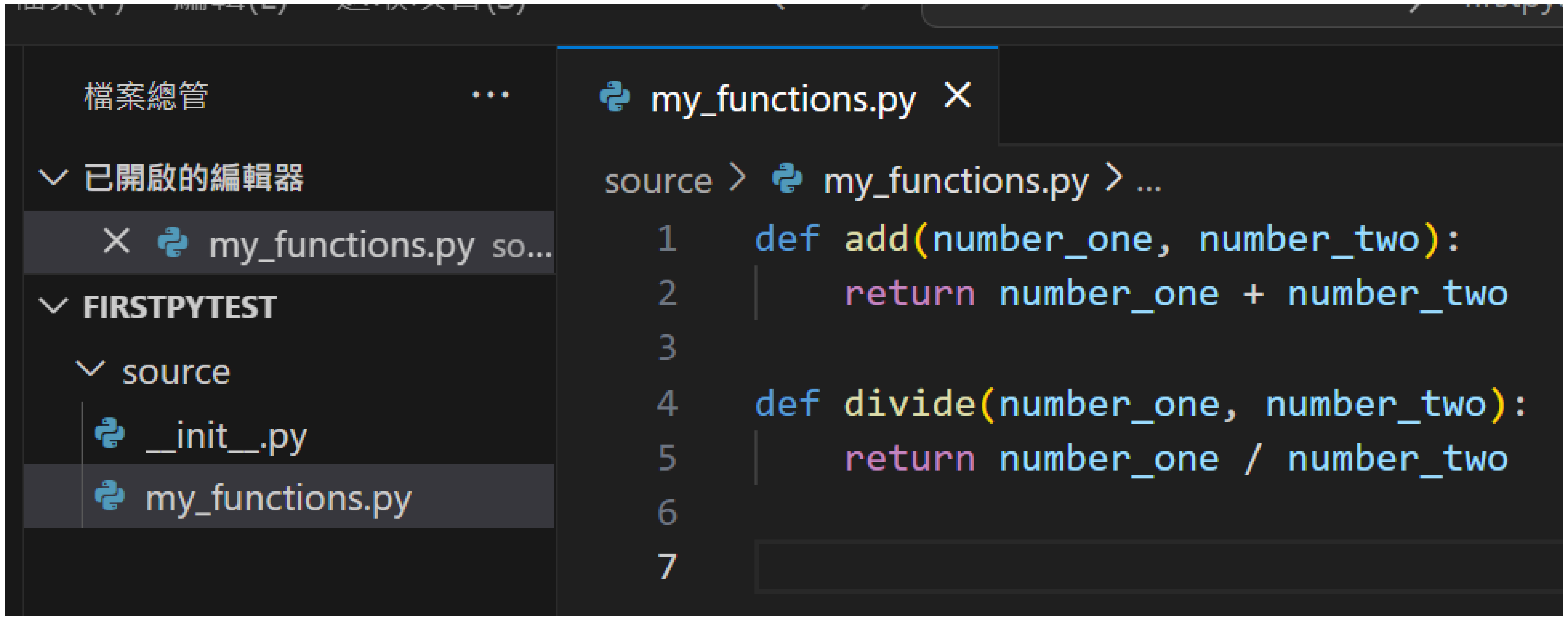
Pytest Tutorial

郭益華

目錄

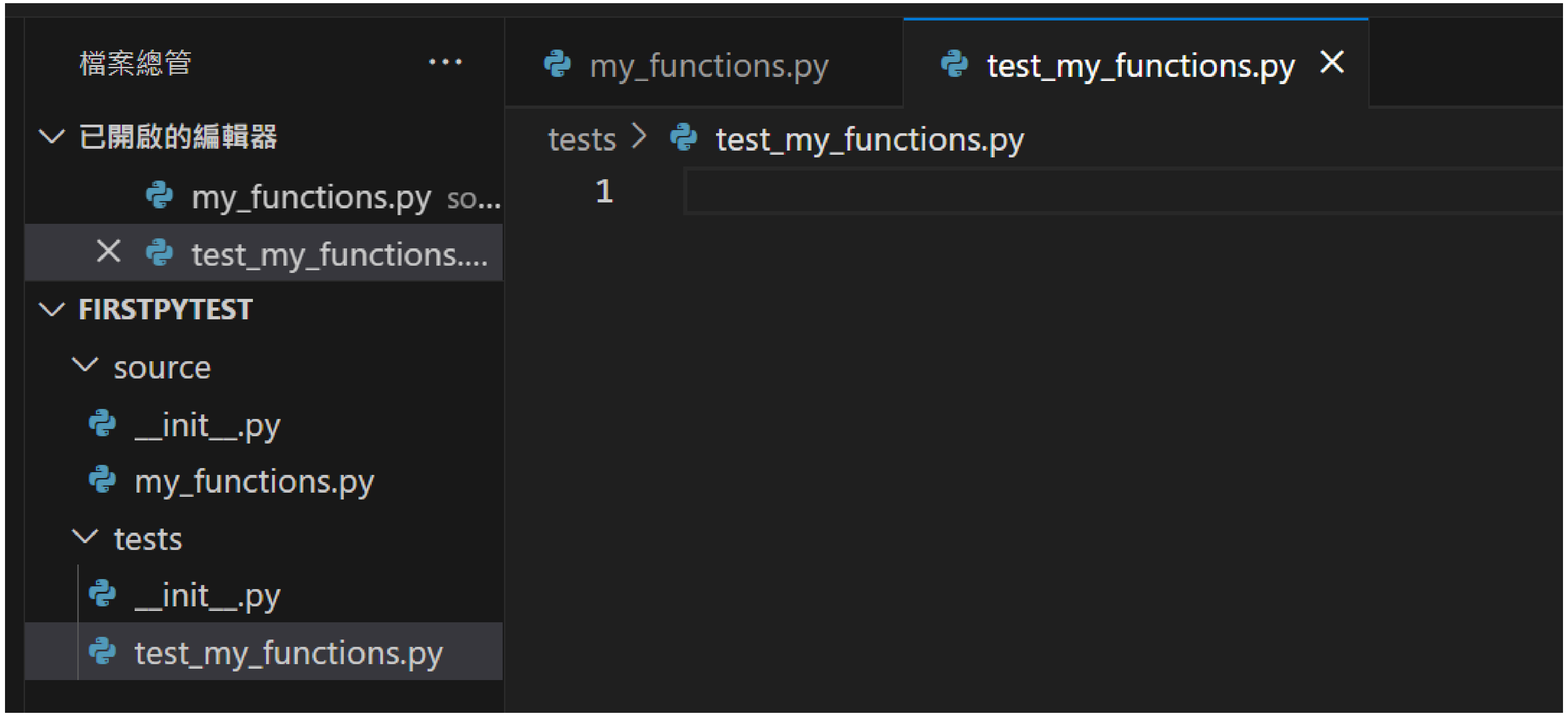
1. 基礎測試 function()

撰寫基本的兩數相加及相除的函式



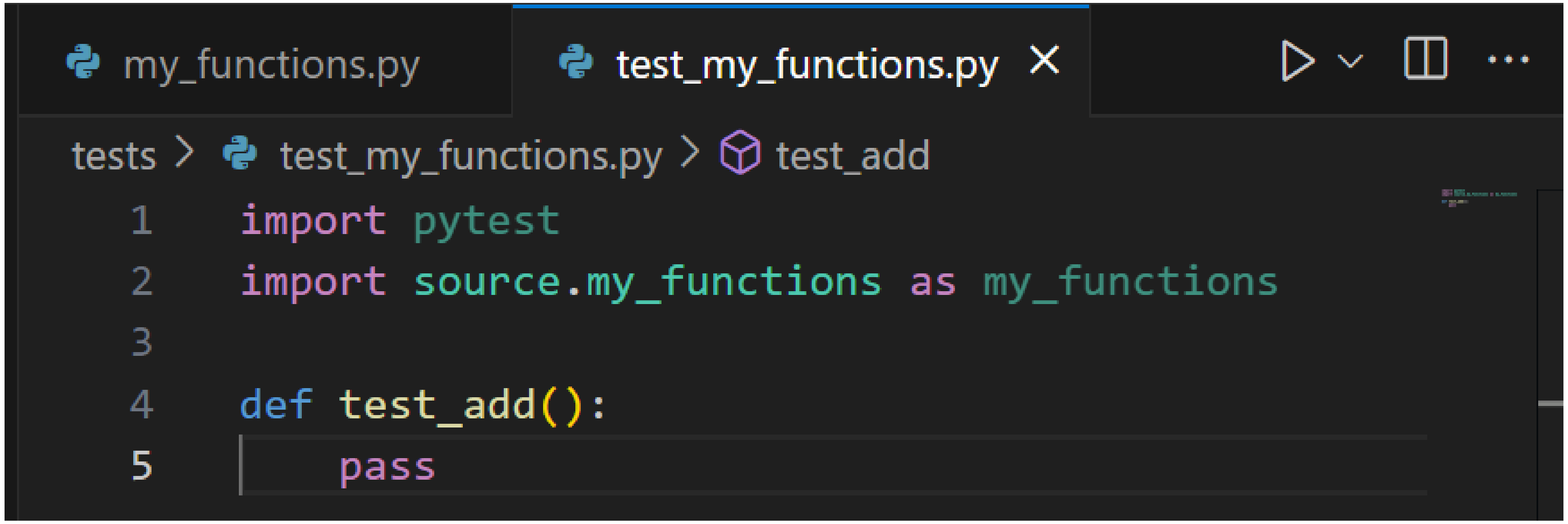
```python
def add(number_one, number_two):
    return number_one + number_two

def divide(number_one, number_two):
    return number_one / number_two

```

新增測試資料夾及程式碼



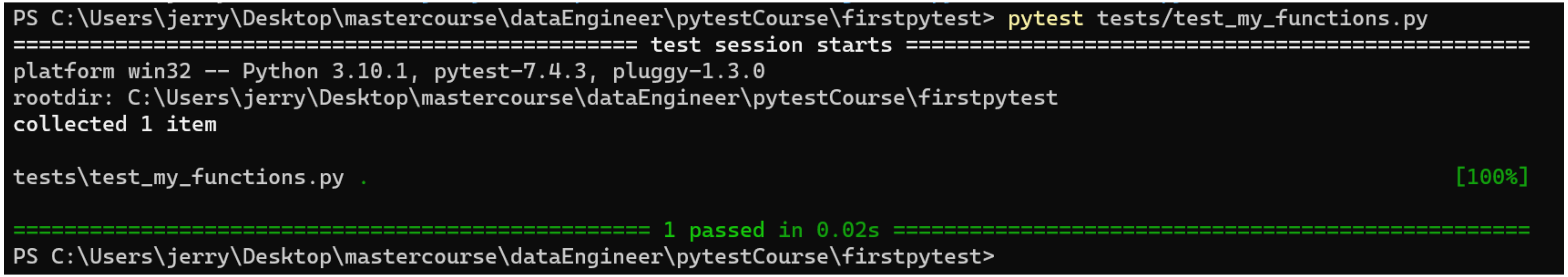
- 測試資料夾一定要命名為tests
- 測試程式碼開頭也必須命名test
- pytest會根據資料夾及程式碼名稱進行相對應的測試

測試 add()



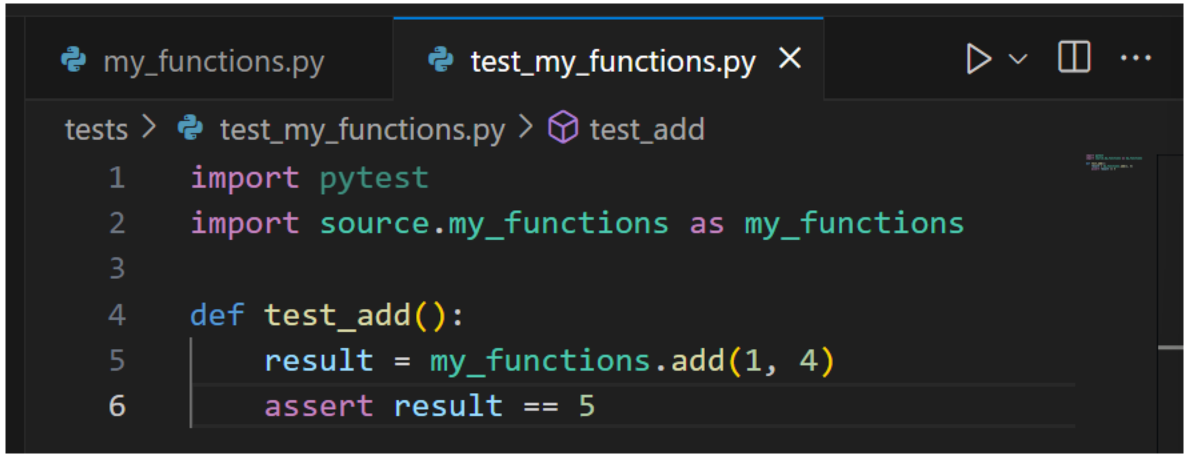
```python
import pytest
import source.my_functions as my_functions

def test_add():
    pass
```

成功



```
PS C:\Users\jerry\Desktop\mastercourse\dataEngineer\pytestCourse\firstpytest> pytest tests/test_my_functions.py
================================================= test session starts ==================================================
platform win32 -- Python 3.10.1, pytest-7.4.3, pluggy-1.3.0
rootdir: C:\Users\jerry\Desktop\mastercourse\dataEngineer\pytestCourse\firstpytest
collected 1 item

tests\test_my_functions.py .                                                                                      [100%]

================================================== 1 passed in 0.02s ===================================================
PS C:\Users\jerry\Desktop\mastercourse\dataEngineer\pytestCourse\firstpytest>
```

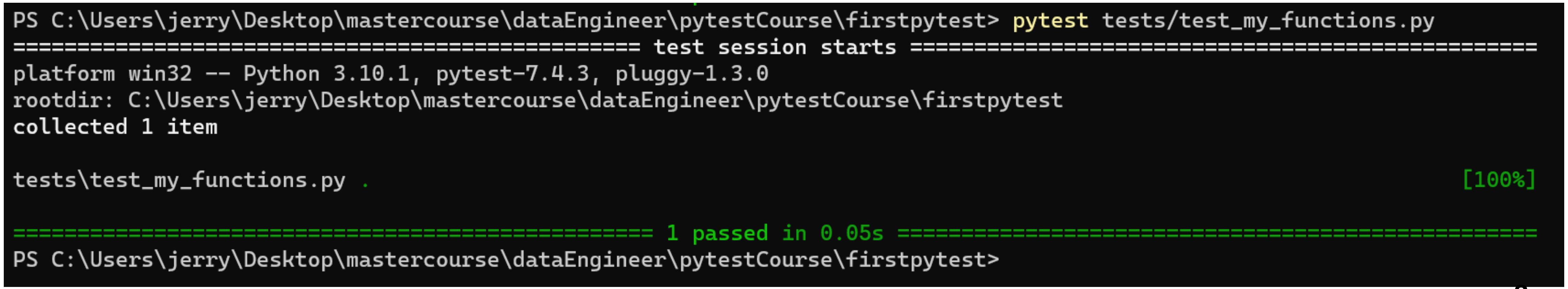
指定 assert


my_functions.py test_my_functions.py

tests > test_my_functions.py > test_add

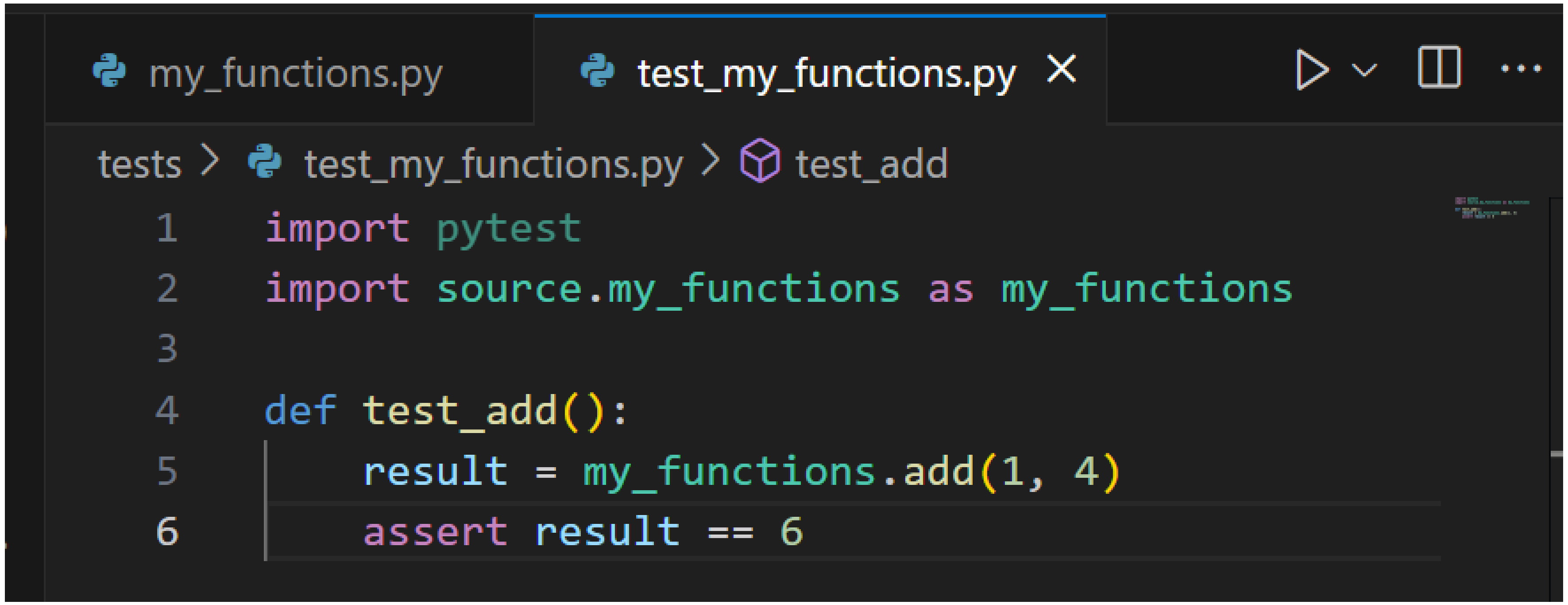
```python
import pytest
import source.my_functions as my_functions

def test_add():
    result = my_functions.add(1, 4)
    assert result == 5
```

指定斷言(assert)，
1 + 4 = 5



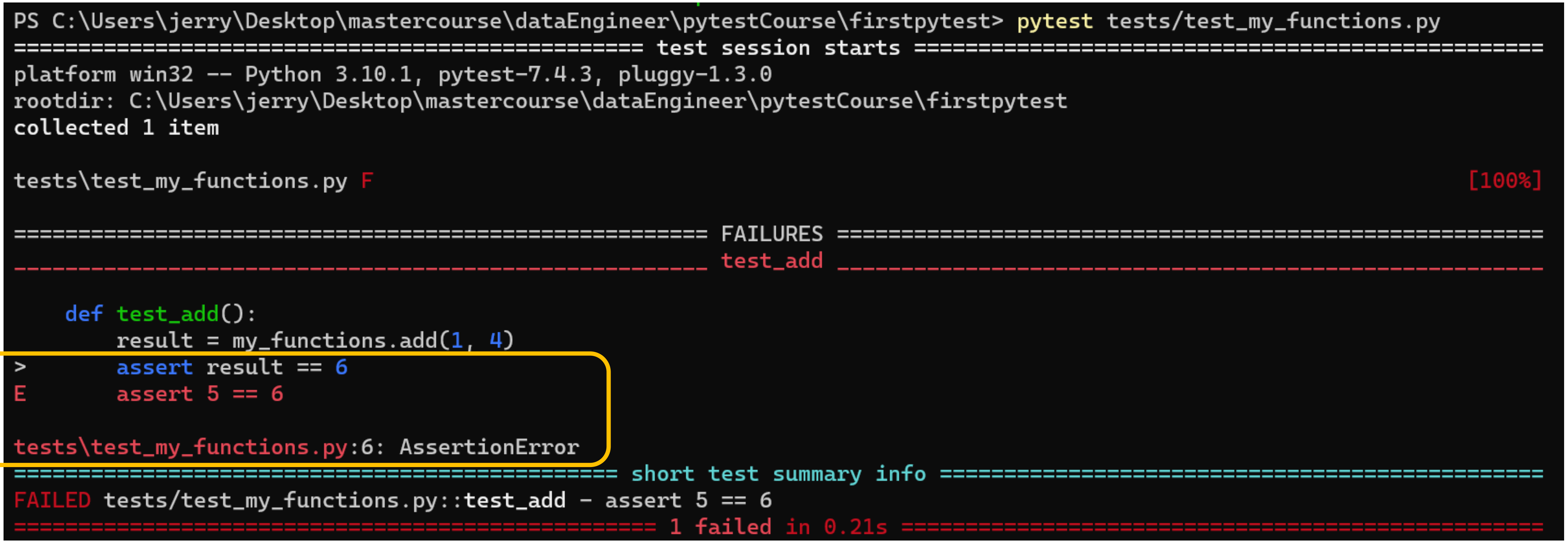
```
PS C:\Users\jerry\Desktop\mastercourse\dataEngineer\pytestCourse\firstpytest> pytest tests/test_my_functions.py
================================================= test session starts =================================================
platform win32 -- Python 3.10.1, pytest-7.4.3, pluggy-1.3.0
rootdir: C:\Users\jerry\Desktop\mastercourse\dataEngineer\pytestCourse\firstpytest
collected 1 item

tests\test_my_functions.py .                                                                                     [100%]

================================================== 1 passed in 0.05s ==================================================
PS C:\Users\jerry\Desktop\mastercourse\dataEngineer\pytestCourse\firstpytest>
```

故意將 assert 設為 6



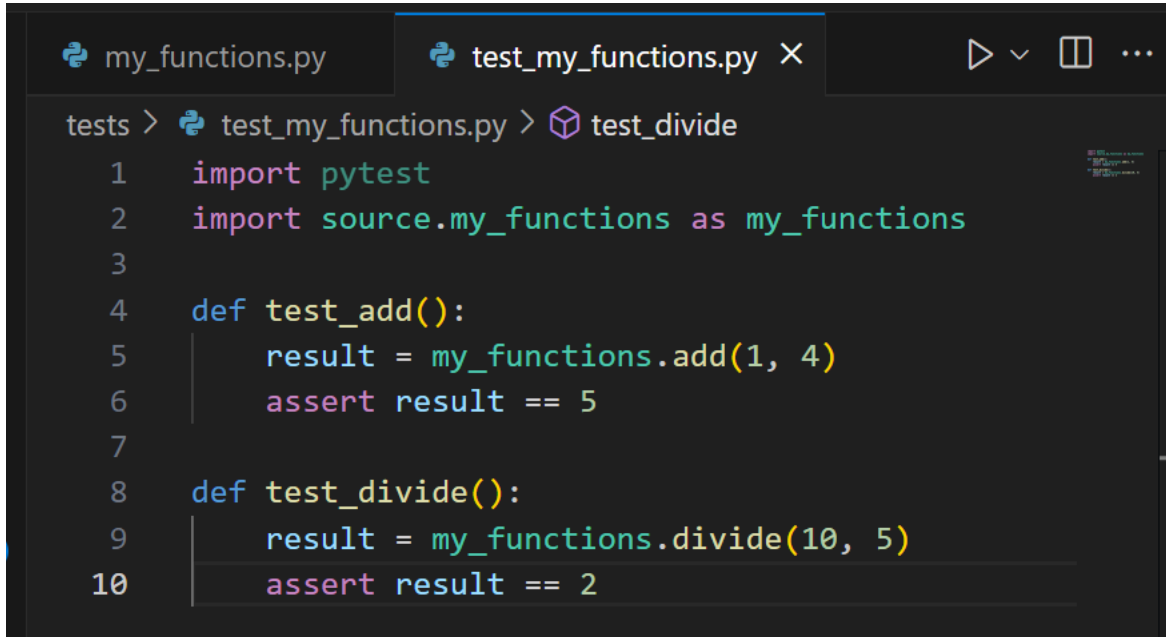
```python
import pytest
import source.my_functions as my_functions

def test_add():
    result = my_functions.add(1, 4)
    assert result == 6
```

會產生錯誤訊息，顯示 assert有錯誤

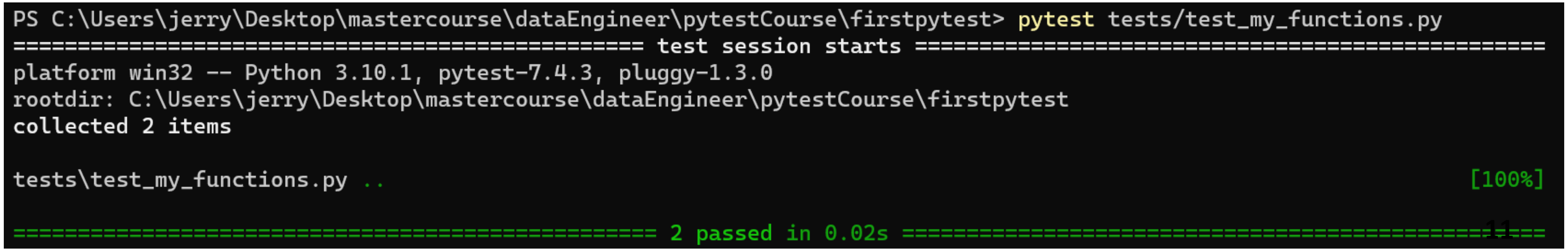


```
PS C:\Users\jerry\Desktop\mastercourse\dataEngineer\pytestCourse\firstpytest> pytest tests/test_my_functions.py
================================================= test session starts =================================================
platform win32 -- Python 3.10.1, pytest-7.4.3, pluggy-1.3.0
rootdir: C:\Users\jerry\Desktop\mastercourse\dataEngineer\pytestCourse\firstpytest
collected 1 item

tests\test_my_functions.py F                                                                                    [100%]

====================================================== FAILURES =======================================================
______________________________________________________ test_add _______________________________________________________

    def test_add():
        result = my_functions.add(1, 4)
>       assert result == 6
E       assert 5 == 6

tests\test_my_functions.py:6: AssertionError
=============================================== short test summary info ===============================================
FAILED tests/test_my_functions.py::test_add - assert 5 == 6
================================================== 1 failed in 0.21s ==================================================
```

新增除法測試 10除以5

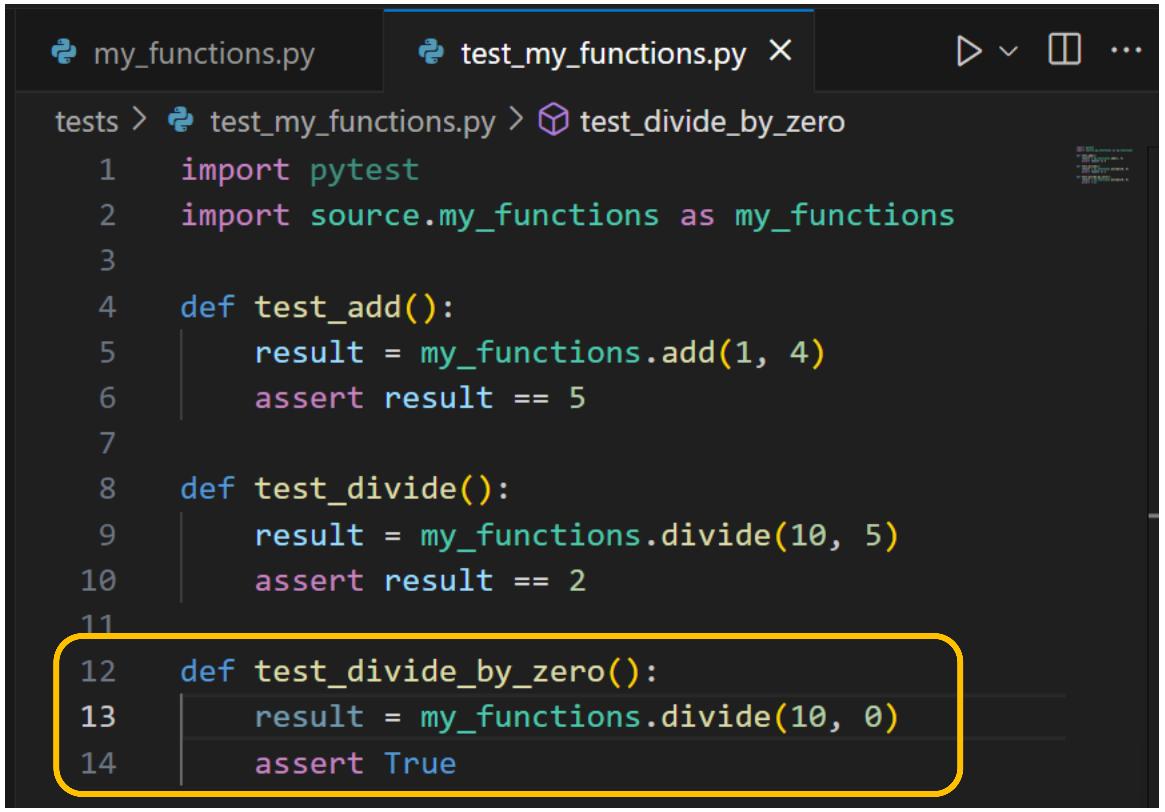


```python
import pytest
import source.my_functions as my_functions

def test_add():
    result = my_functions.add(1, 4)
    assert result == 5

def test_divide():
    result = my_functions.divide(10, 5)
    assert result == 2
```



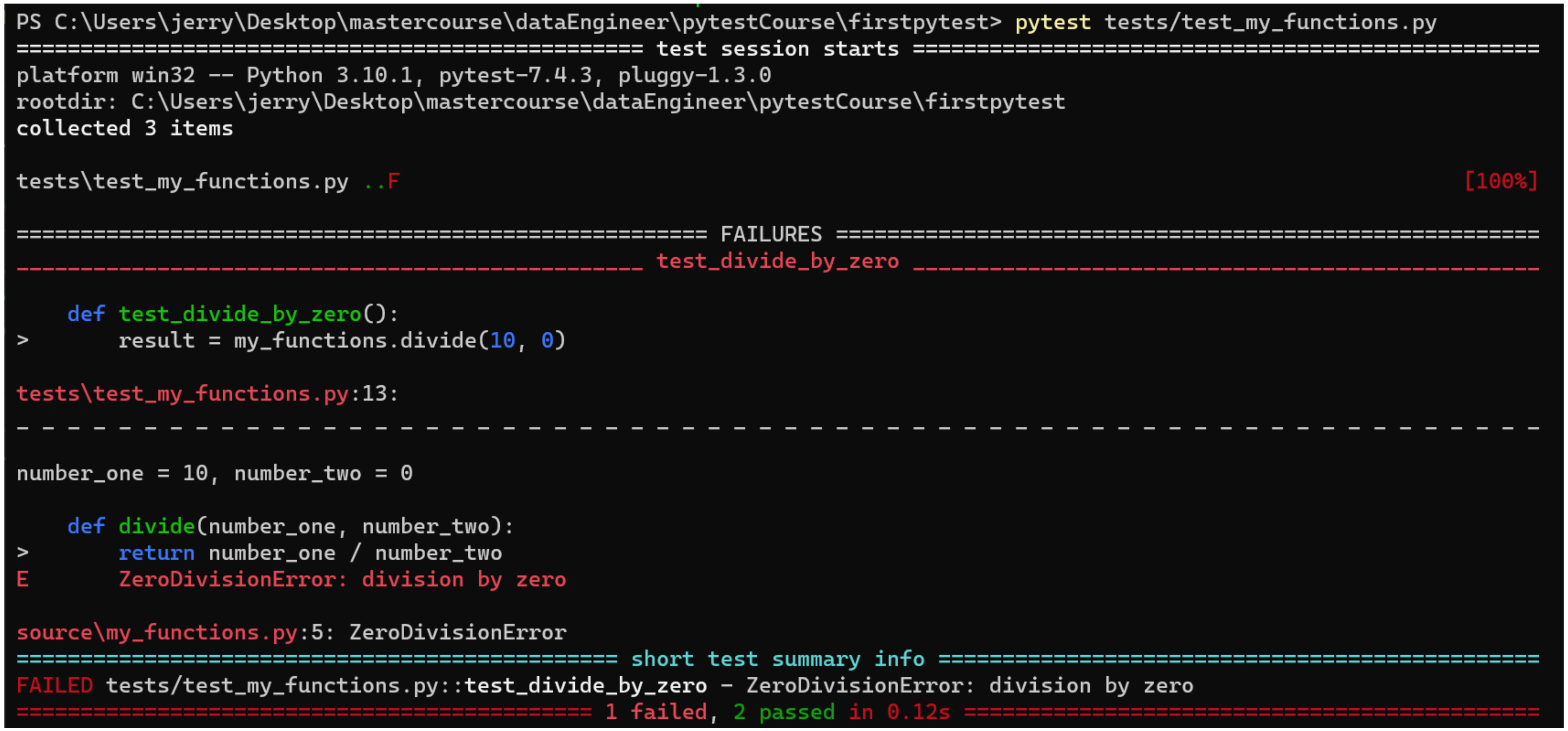
```
PS C:\Users\jerry\Desktop\mastercourse\dataEngineer\pytestCourse\firstpytest> pytest tests/test_my_functions.py
================================================= test session starts ==================================================
platform win32 -- Python 3.10.1, pytest-7.4.3, pluggy-1.3.0
rootdir: C:\Users\jerry\Desktop\mastercourse\dataEngineer\pytestCourse\firstpytest
collected 2 items

tests\test_my_functions.py ..                                                                                     [100%]

================================================== 2 passed in 0.02s ===================================================
```

測試除以0



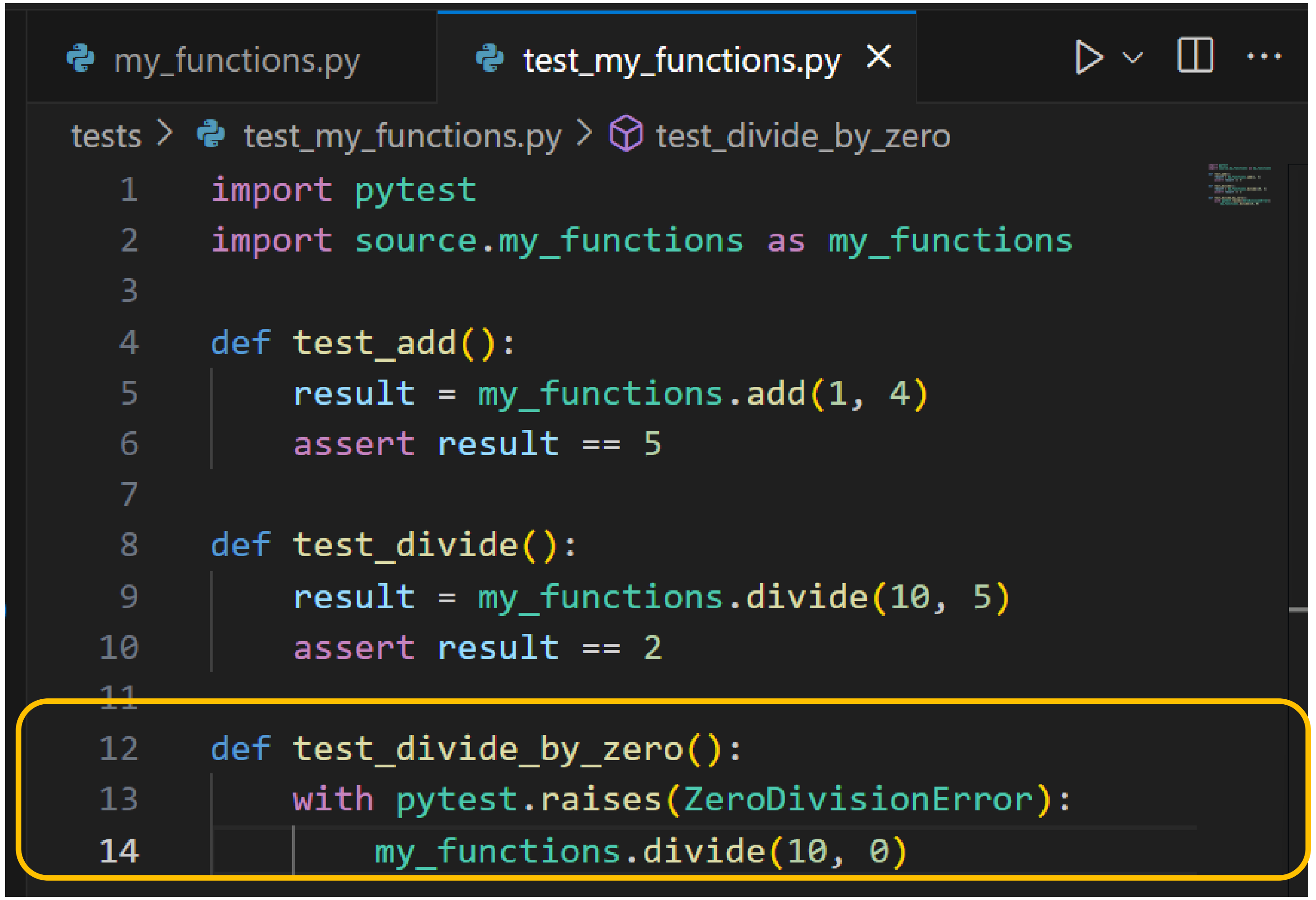
```python
import pytest
import source.my_functions as my_functions

def test_add():
    result = my_functions.add(1, 4)
    assert result == 5

def test_divide():
    result = my_functions.divide(10, 5)
    assert result == 2

def test_divide_by_zero():
    result = my_functions.divide(10, 0)
    assert True
```

會產生 ZeroDivisionError



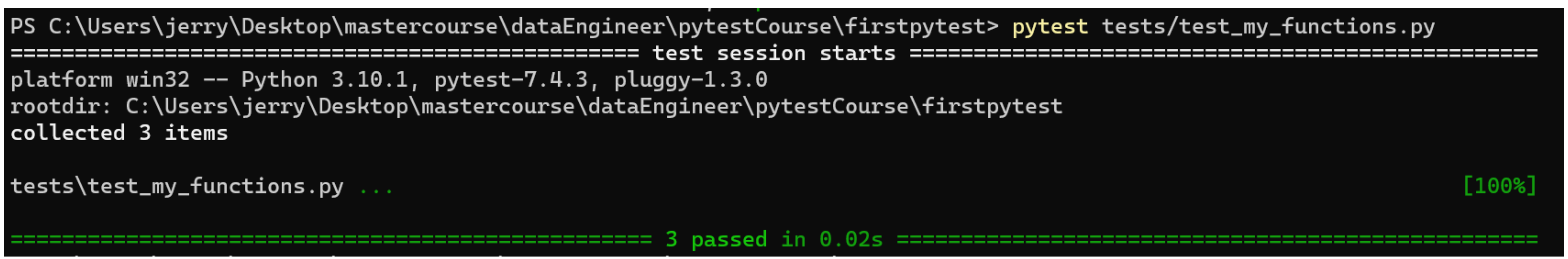
```
PS C:\Users\jerry\Desktop\mastercourse\dataEngineer\pytestCourse\firstpytest> pytest tests/test_my_functions.py
================================================= test session starts =================================================
platform win32 -- Python 3.10.1, pytest-7.4.3, pluggy-1.3.0
rootdir: C:\Users\jerry\Desktop\mastercourse\dataEngineer\pytestCourse\firstpytest
collected 3 items

tests\test_my_functions.py ..F                                                                                  [100%]

====================================================== FAILURES =======================================================
_________________________________________________ test_divide_by_zero _________________________________________________

    def test_divide_by_zero():
>       result = my_functions.divide(10, 0)

tests\test_my_functions.py:13:
_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _

number_one = 10, number_two = 0

    def divide(number_one, number_two):
>       return number_one / number_two
E       ZeroDivisionError: division by zero

source\my_functions.py:5: ZeroDivisionError
=============================================== short test summary info ===============================================
FAILED tests/test_my_functions.py::test_divide_by_zero - ZeroDivisionError: division by zero
============================================= 1 failed, 2 passed in 0.12s =============================================
```

將測試碼改為with ZeroDivisionError



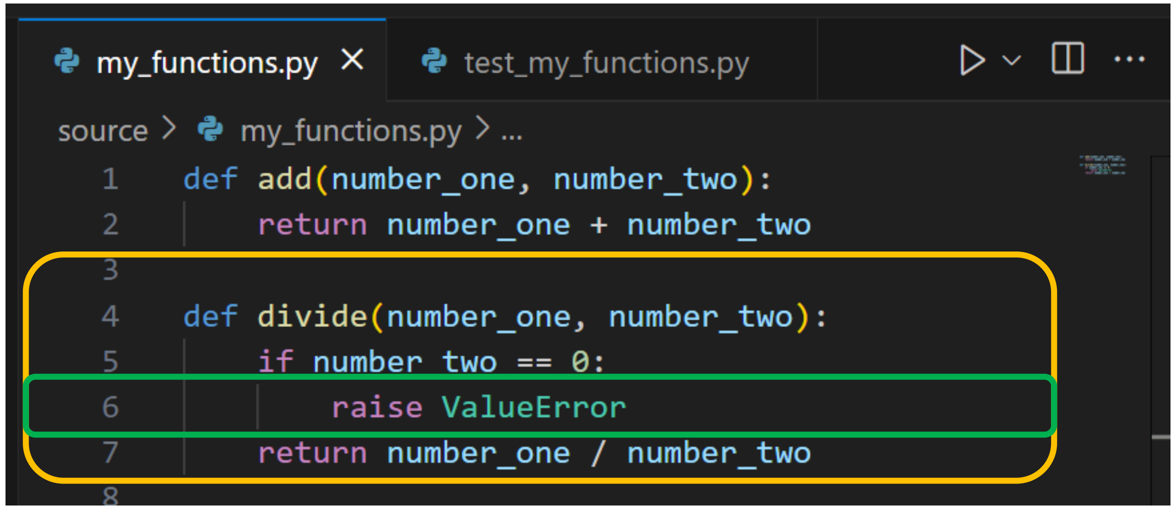
```python
import pytest
import source.my_functions as my_functions

def test_add():
    result = my_functions.add(1, 4)
    assert result == 5

def test_divide():
    result = my_functions.divide(10, 5)
    assert result == 2

def test_divide_by_zero():
    with pytest.raises(ZeroDivisionError):
        my_functions.divide(10, 0)
```

測試成功



```
PS C:\Users\jerry\Desktop\mastercourse\dataEngineer\pytestCourse\firstpytest> pytest tests/test_my_functions.py
================================================= test session starts ==================================================
platform win32 -- Python 3.10.1, pytest-7.4.3, pluggy-1.3.0
rootdir: C:\Users\jerry\Desktop\mastercourse\dataEngineer\pytestCourse\firstpytest
collected 3 items

tests\test_my_functions.py ...                                                                                   [100%]

================================================== 3 passed in 0.02s ===================================================
```

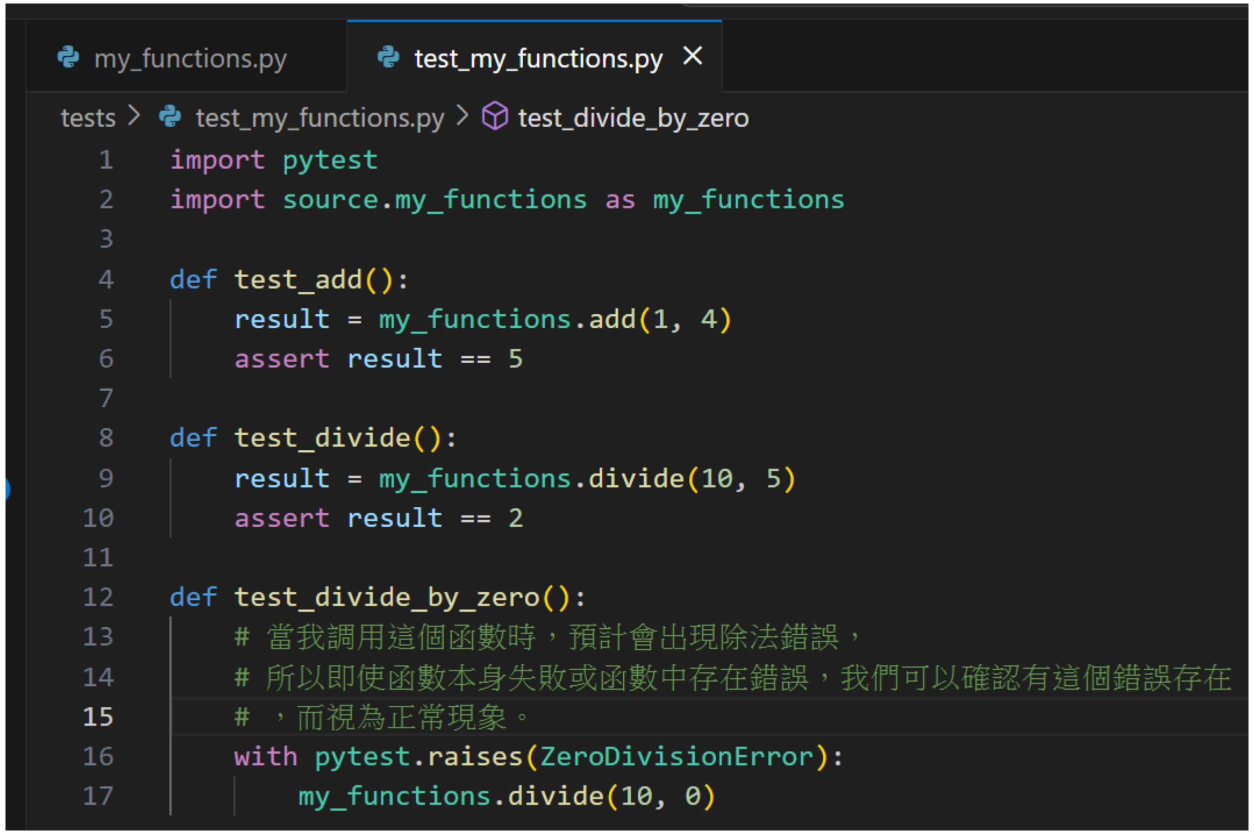
修改主程式碼 divide()



my_functions.py × test_my_functions.py

source > my_functions.py > ...

```python
def add(number_one, number_two):
    return number_one + number_two

def divide(number_one, number_two):
    if number_two == 0:
        raise ValueError
    return number_one / number_two
```

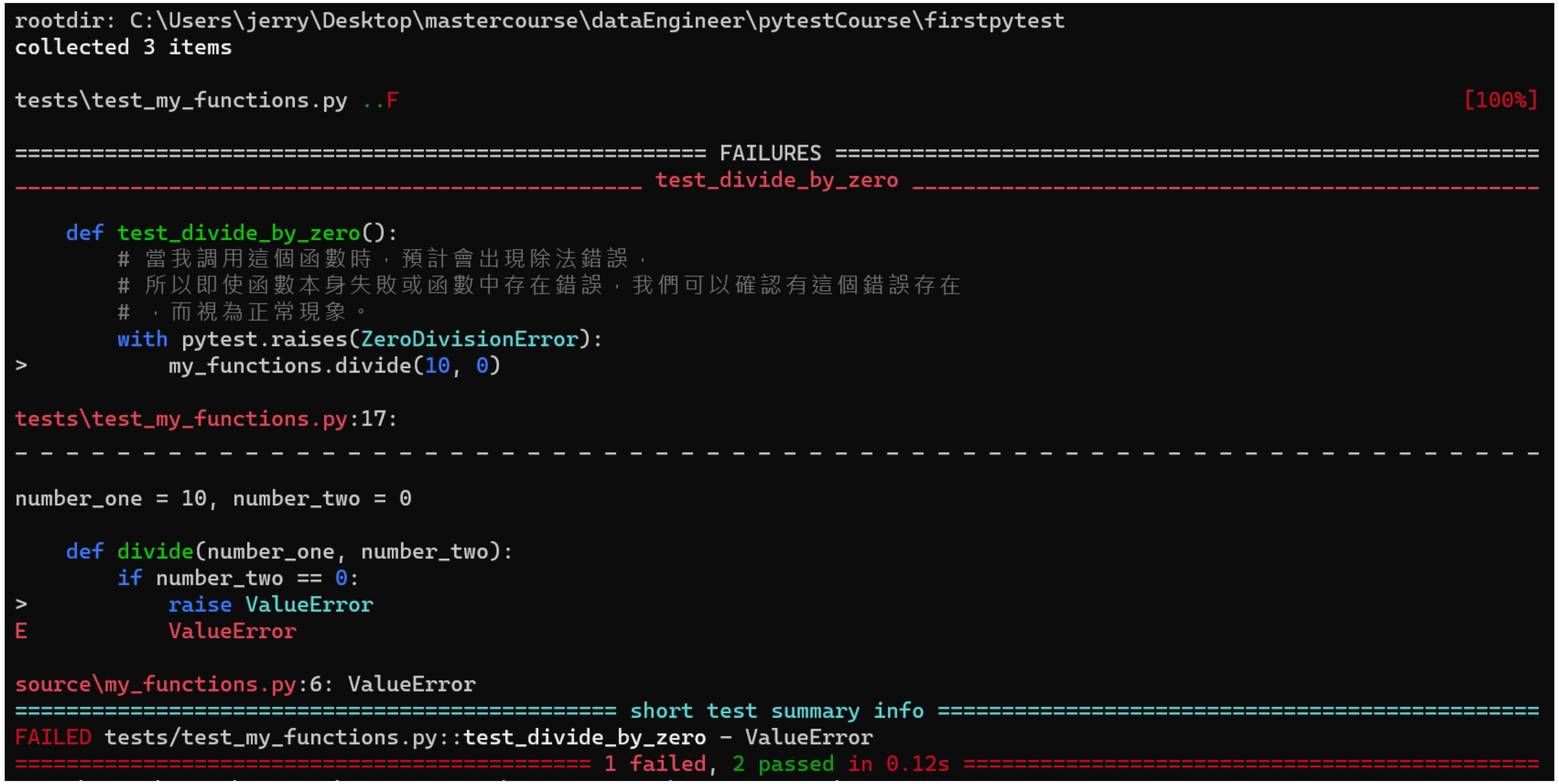


my_functions.py test_my_functions.py ×

tests > test_my_functions.py > test_divide_by_zero

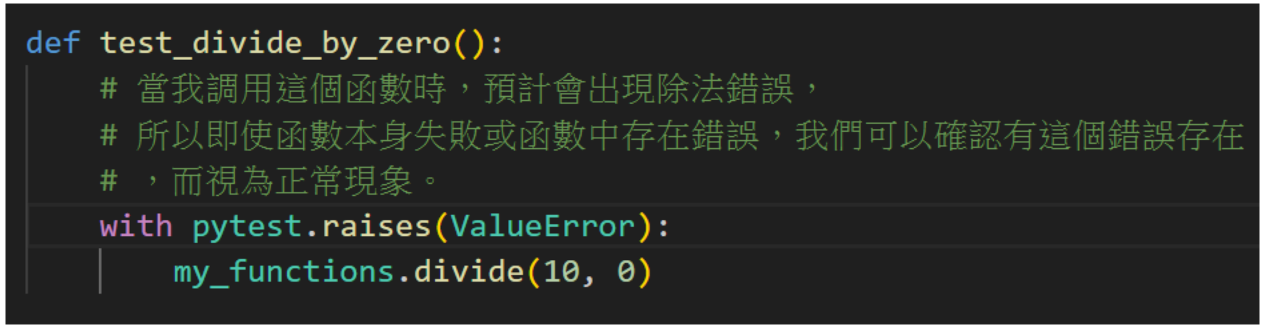
```python
import pytest
import source.my_functions as my_functions

def test_add():
    result = my_functions.add(1, 4)
    assert result == 5

def test_divide():
    result = my_functions.divide(10, 5)
    assert result == 2

def test_divide_by_zero():
    # 當我調用這個函數時，預計會出現除法錯誤，
    # 所以即使函數本身失敗或函數中存在錯誤，我們可以確認有這個錯誤存在
    # ，而視為正常現象。
    with pytest.raises(ZeroDivisionError):
        my_functions.divide(10, 0)
```

產生錯誤 ValueError 的問題

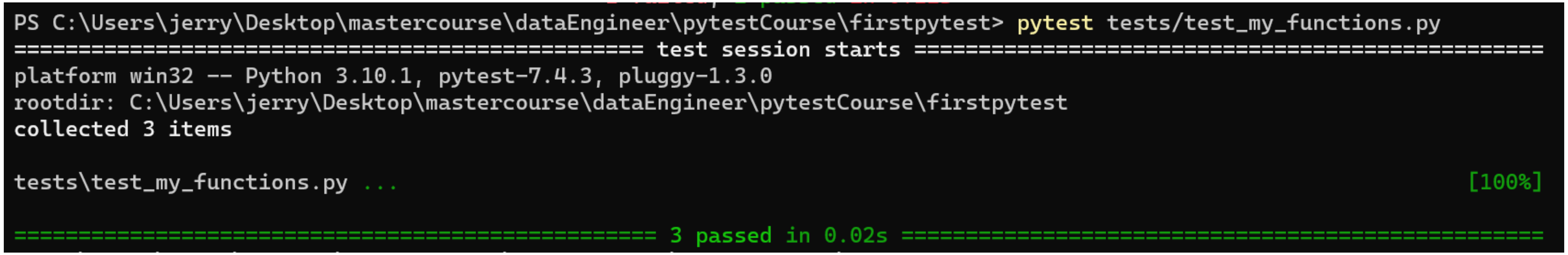


```
rootdir: C:\Users\jerry\Desktop\mastercourse\dataEngineer\pytestCourse\firstpytest
collected 3 items

tests\test_my_functions.py ..F                                                       [100%]

====================================================== FAILURES =======================================================
_______________________________________________ test_divide_by_zero ________________________________________________

    def test_divide_by_zero():
        # 當我調用這個函數時，預計會出現除法錯誤，
        # 所以即使函數本身失敗或函數中存在錯誤，我們可以確認有這個錯誤存在
        # ，而視為正常現象。
        with pytest.raises(ZeroDivisionError):
>           my_functions.divide(10, 0)

tests\test_my_functions.py:17:
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

number_one = 10, number_two = 0

    def divide(number_one, number_two):
        if number_two == 0:
>           raise ValueError
E           ValueError

source\my_functions.py:6: ValueError
=============================================== short test summary info ===============================================
FAILED tests/test_my_functions.py::test_divide_by_zero - ValueError
============================================ 1 failed, 2 passed in 0.12s ============================================
```

將測試碼的 with 改為 ValueError

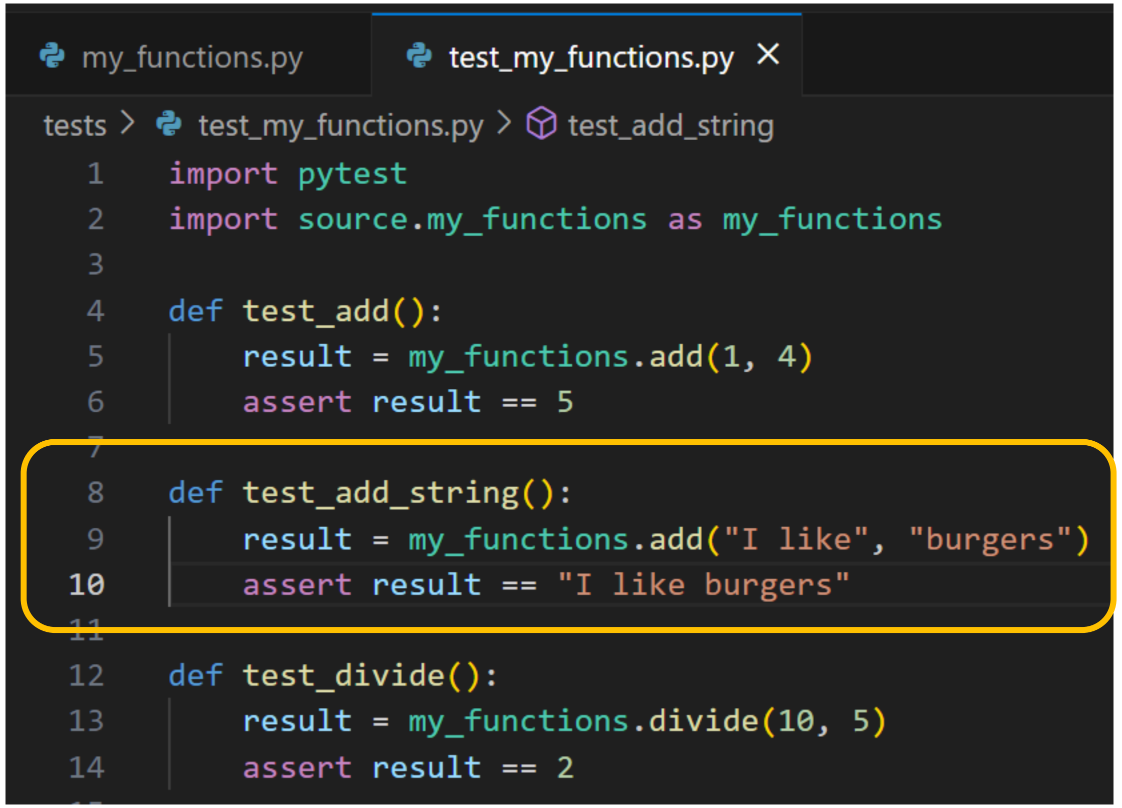


```python
def test_divide_by_zero():
    # 當我調用這個函數時，預計會出現除法錯誤，
    # 所以即使函數本身失敗或函數中存在錯誤，我們可以確認有這個錯誤存在
    # ，而視為正常現象。
    with pytest.raises(ValueError):
        my_functions.divide(10, 0)
```



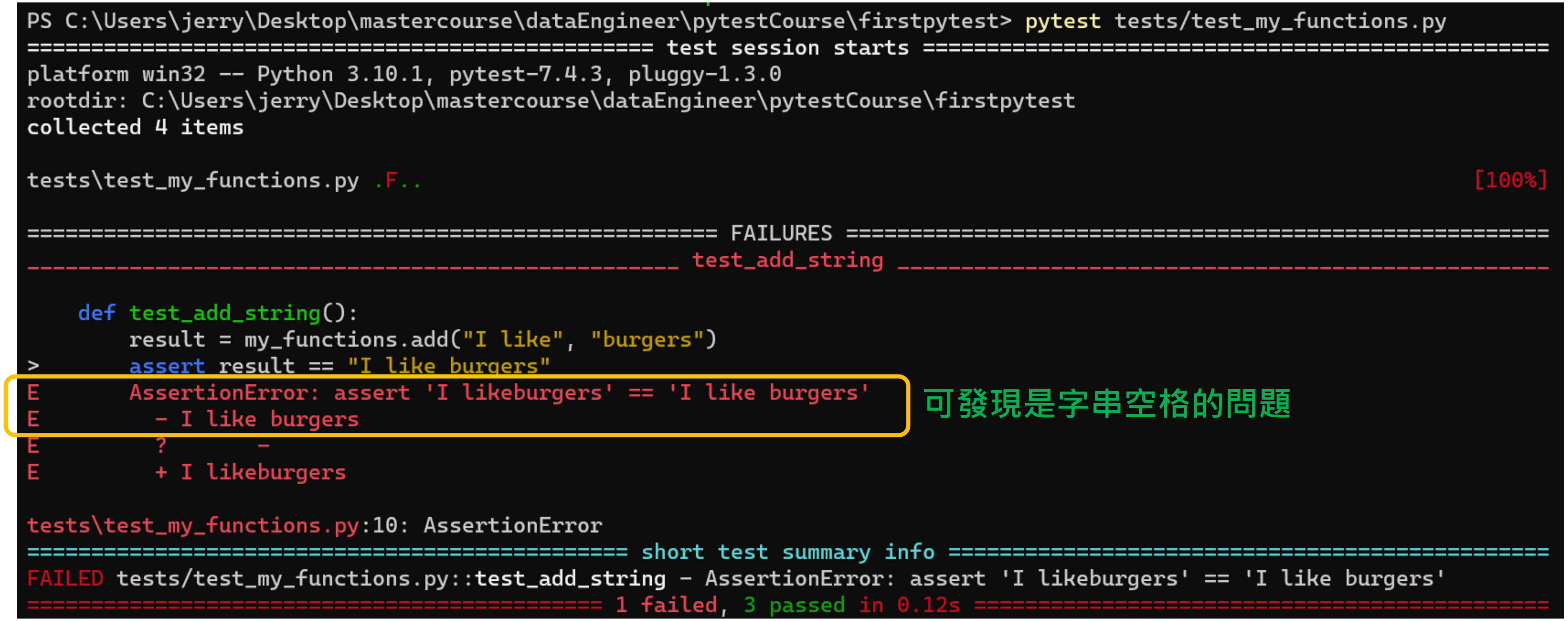
```
PS C:\Users\jerry\Desktop\mastercourse\dataEngineer\pytestCourse\firstpytest> pytest tests/test_my_functions.py
================================================= test session starts ==================================================
platform win32 -- Python 3.10.1, pytest-7.4.3, pluggy-1.3.0
rootdir: C:\Users\jerry\Desktop\mastercourse\dataEngineer\pytestCourse\firstpytest
collected 3 items

tests\test_my_functions.py ...                                                                                   [100%]

================================================== 3 passed in 0.02s ===================================================
```

測試字串相加


my_functions.py test_my_functions.py ×
tests > test_my_functions.py > test_add_string

```python
import pytest
import source.my_functions as my_functions

def test_add():
    result = my_functions.add(1, 4)
    assert result == 5

def test_add_string():
    result = my_functions.add("I like", "burgers")
    assert result == "I like burgers"

def test_divide():
    result = my_functions.divide(10, 5)
    assert result == 2
```

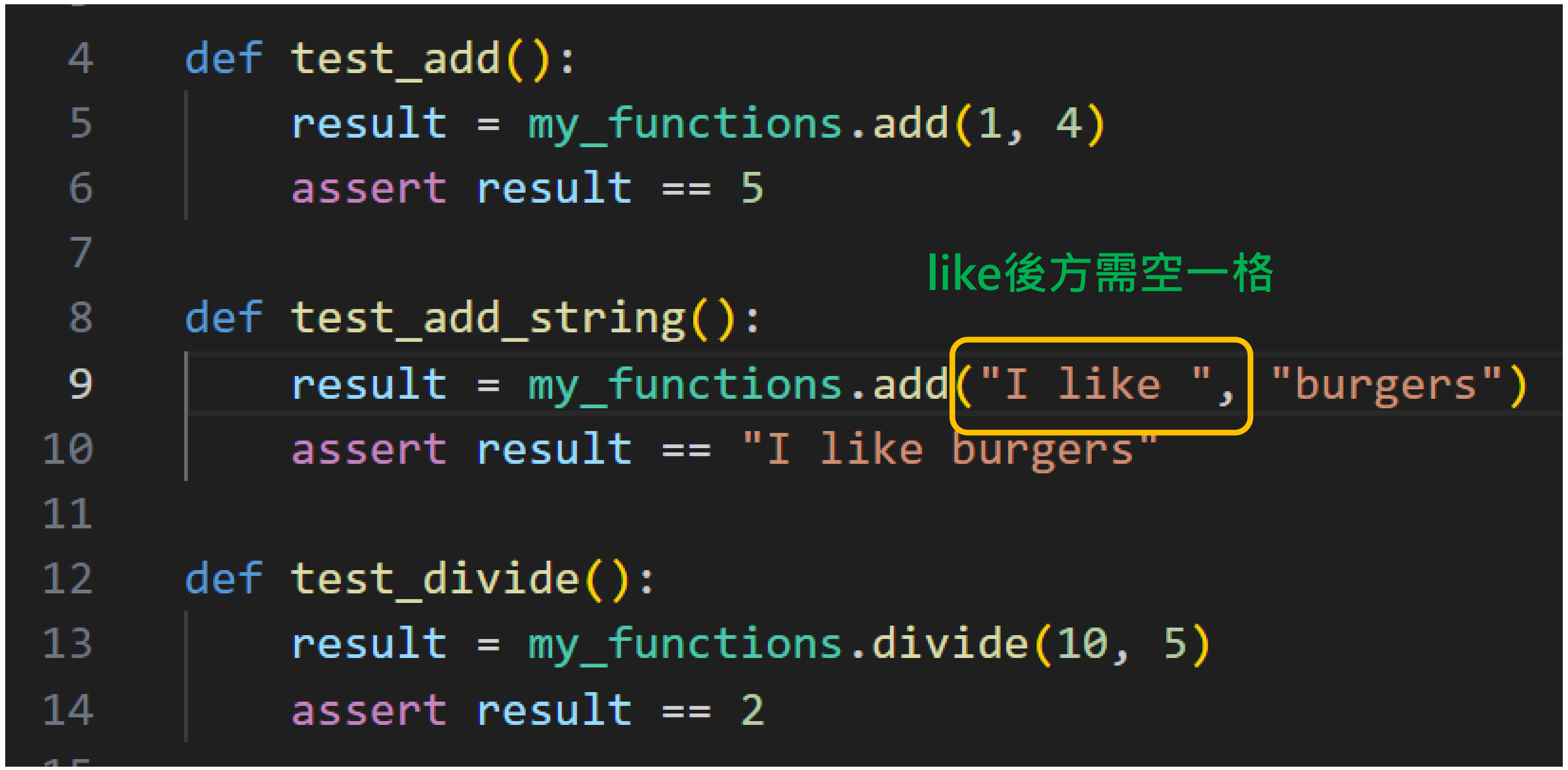
產生錯誤



```
PS C:\Users\jerry\Desktop\mastercourse\dataEngineer\pytestCourse\firstpytest> pytest tests/test_my_functions.py
=============================================== test session starts ================================================
platform win32 -- Python 3.10.1, pytest-7.4.3, pluggy-1.3.0
rootdir: C:\Users\jerry\Desktop\mastercourse\dataEngineer\pytestCourse\firstpytest
collected 4 items

tests\test_my_functions.py .F..                                                                               [100%]

===================================================== FAILURES =====================================================
_________________________________________________ test_add_string __________________________________________________

    def test_add_string():
        result = my_functions.add("I like", "burgers")
>       assert result == "I like burgers"
E       AssertionError: assert 'I likeburgers' == 'I like burgers'
E         - I like burgers
E         ?       -
E         + I likeburgers

tests\test_my_functions.py:10: AssertionError
============================================= short test summary info ==============================================
FAILED tests/test_my_functions.py::test_add_string - AssertionError: assert 'I likeburgers' == 'I like burgers'
=========================================== 1 failed, 3 passed in 0.12s ============================================
```

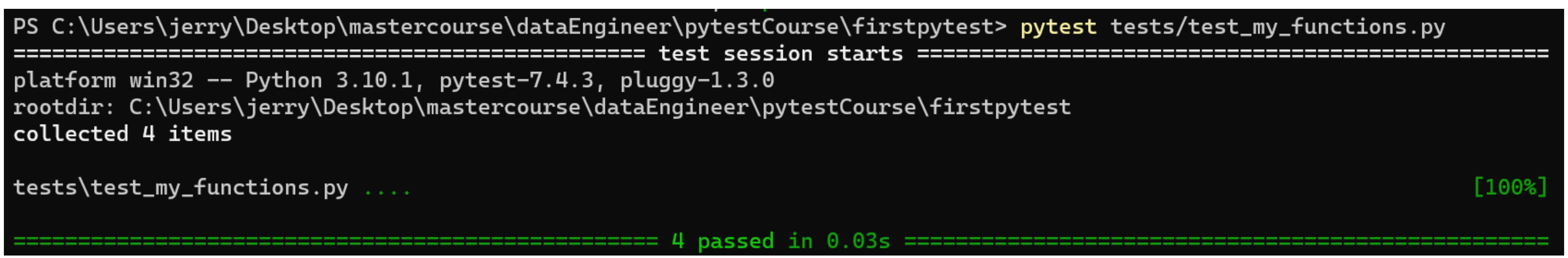
可發現是字串空格的問題

修改 like



```
def test_add():
    result = my_functions.add(1, 4)
    assert result == 5

def test_add_string():
    result = my_functions.add("I like ", "burgers")
    assert result == "I like burgers"

def test_divide():
    result = my_functions.divide(10, 5)
    assert result == 2
```

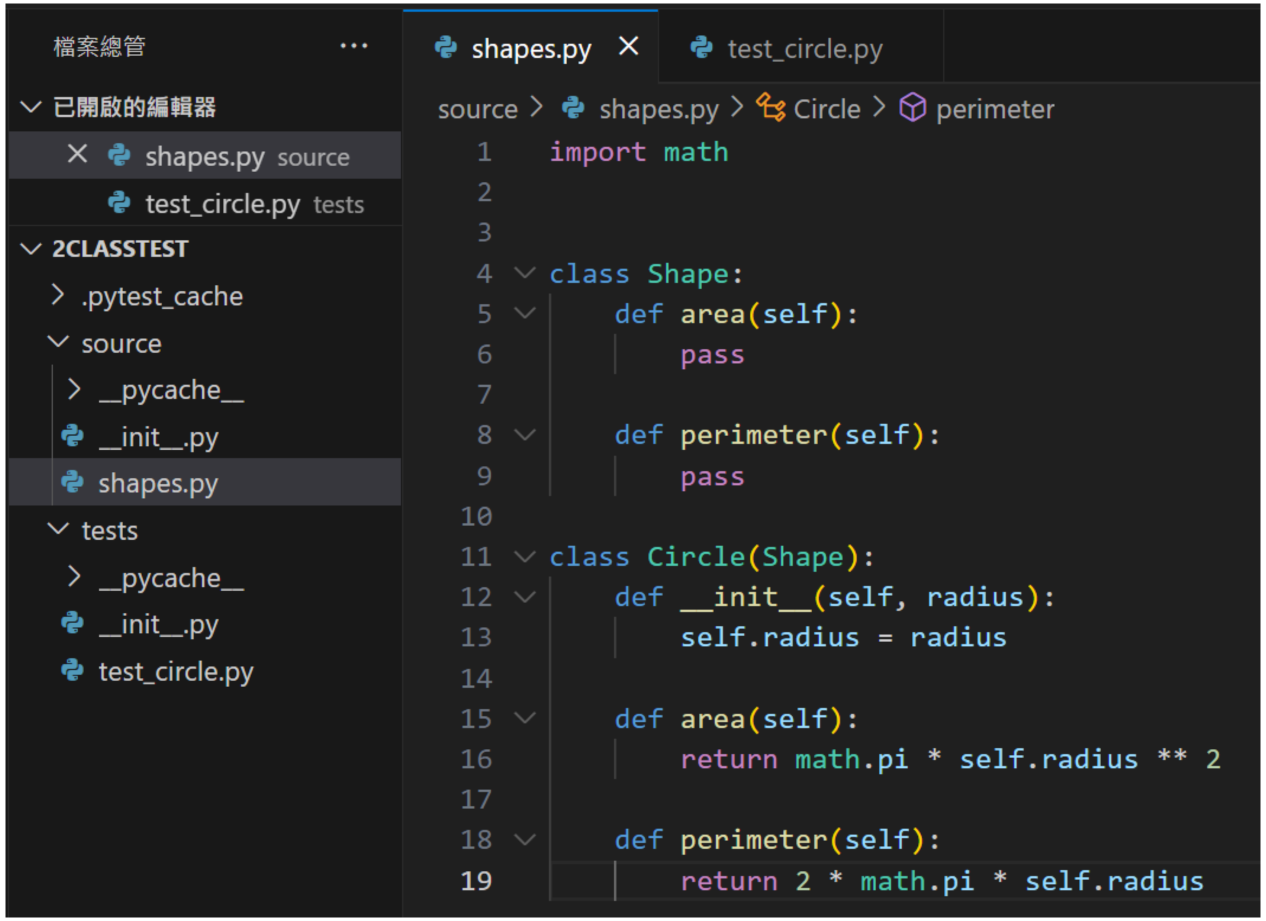
測試成功



```
PS C:\Users\jerry\Desktop\mastercourse\dataEngineer\pytestCourse\firstpytest> pytest tests/test_my_functions.py
================================================= test session starts ==================================================
platform win32 -- Python 3.10.1, pytest-7.4.3, pluggy-1.3.0
rootdir: C:\Users\jerry\Desktop\mastercourse\dataEngineer\pytestCourse\firstpytest
collected 4 items

tests\test_my_functions.py ....                                                                                  [100%]

================================================== 4 passed in 0.03s ===================================================
```

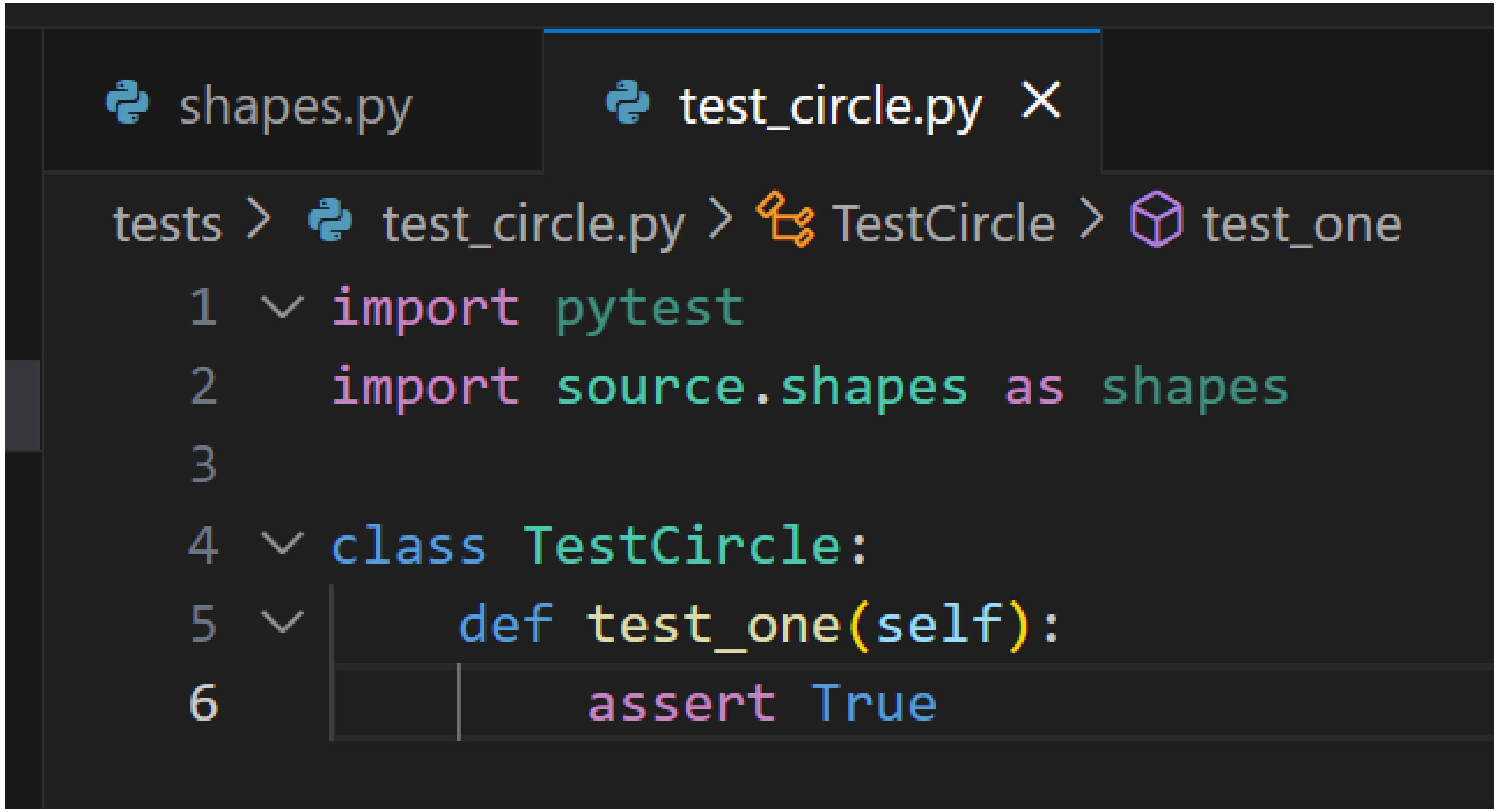
2. Class-based Tests

撰寫一個計算圓形面積周長的class



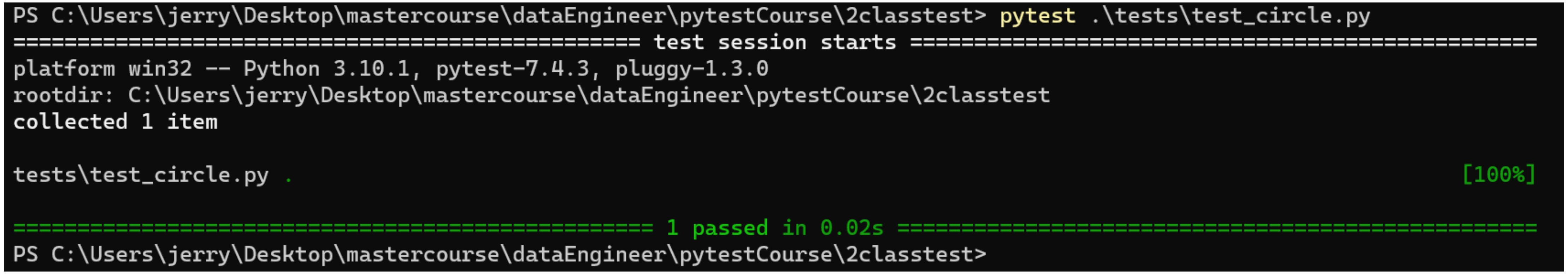
```python
import math


class Shape:
    def area(self):
        pass

    def perimeter(self):
        pass

class Circle(Shape):
    def __init__(self, radius):
        self.radius = radius

    def area(self):
        return math.pi * self.radius ** 2

    def perimeter(self):
        return 2 * math.pi * self.radius
```

撰寫測試.py

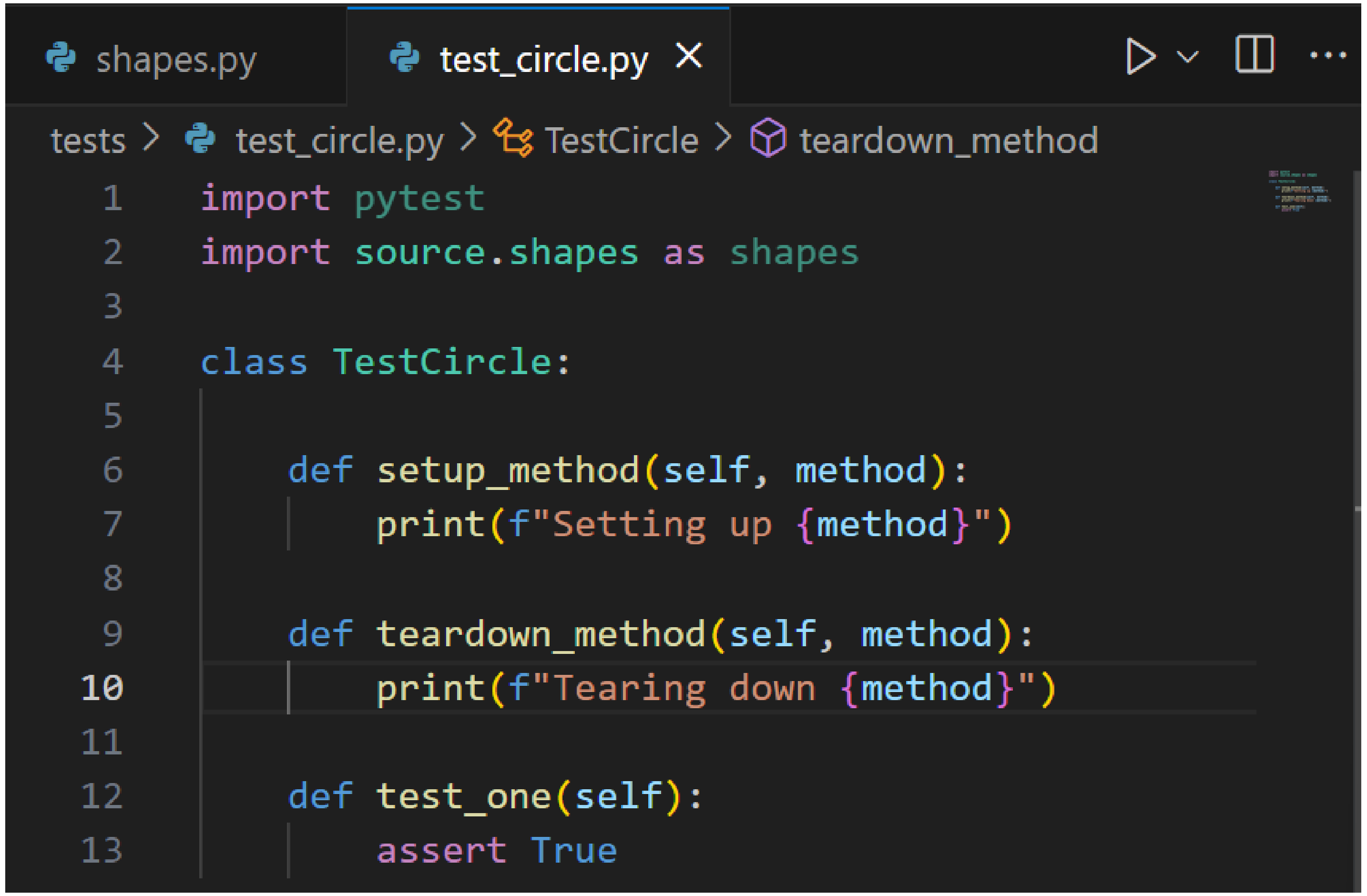


```python
import pytest
import source.shapes as shapes

class TestCircle:
    def test_one(self):
        assert True
```

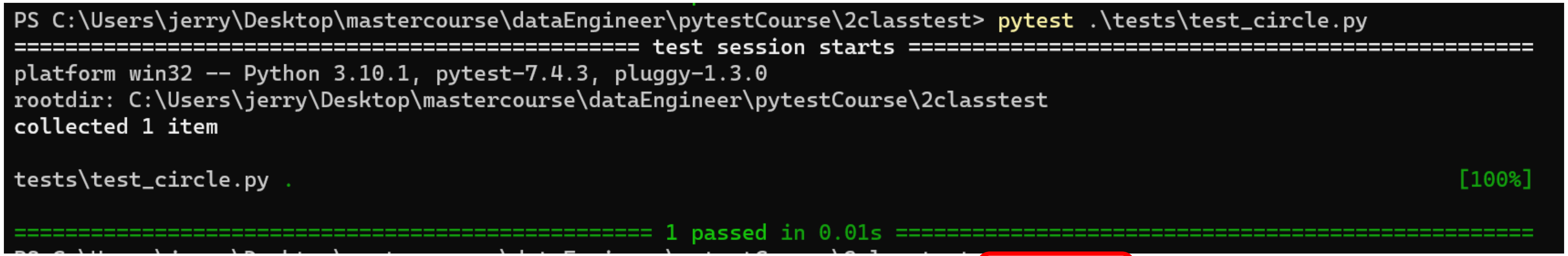
成功



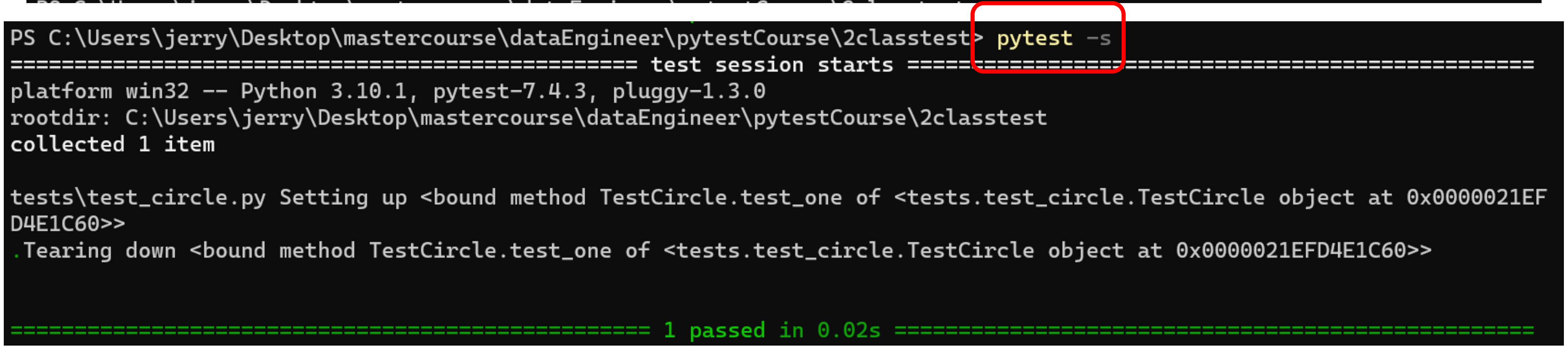
```
PS C:\Users\jerry\Desktop\mastercourse\dataEngineer\pytestCourse\2classtest> pytest .\tests\test_circle.py
================================================= test session starts =================================================
platform win32 -- Python 3.10.1, pytest-7.4.3, pluggy-1.3.0
rootdir: C:\Users\jerry\Desktop\mastercourse\dataEngineer\pytestCourse\2classtest
collected 1 item

tests\test_circle.py .                                                                                            [100%]

================================================== 1 passed in 0.02s ==================================================
PS C:\Users\jerry\Desktop\mastercourse\dataEngineer\pytestCourse\2classtest>
```



```python
import pytest
import source.shapes as shapes

class TestCircle:

    def setup_method(self, method):
        print(f"Setting up {method}")

    def teardown_method(self, method):
        print(f"Tearing down {method}")

    def test_one(self):
        assert True
```



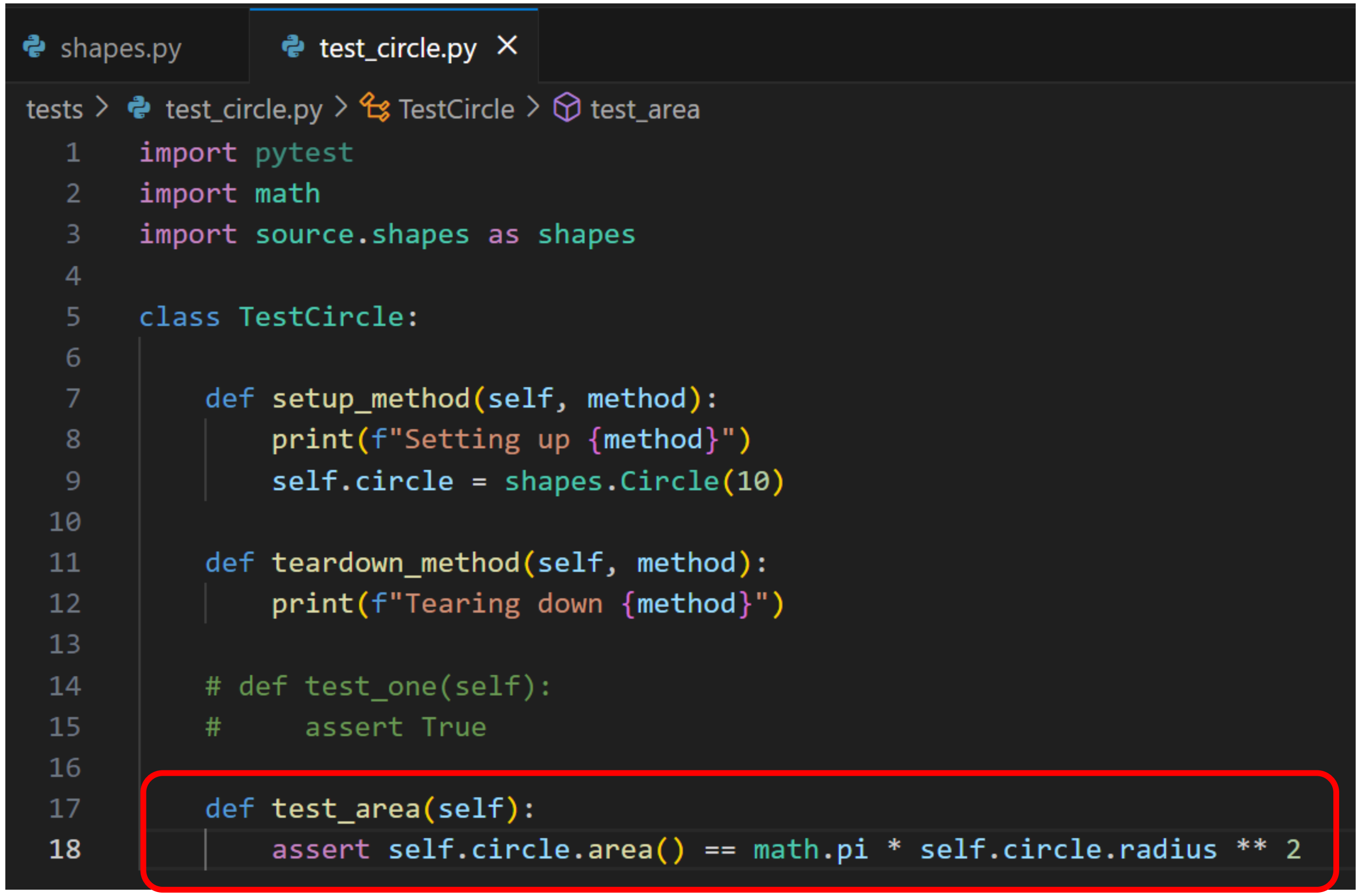
```
PS C:\Users\jerry\Desktop\mastercourse\dataEngineer\pytestCourse\2classtest> pytest .\tests\test_circle.py
================================================= test session starts =================================================
platform win32 -- Python 3.10.1, pytest-7.4.3, pluggy-1.3.0
rootdir: C:\Users\jerry\Desktop\mastercourse\dataEngineer\pytestCourse\2classtest
collected 1 item

tests\test_circle.py .                                                                                          [100%]

================================================== 1 passed in 0.01s ==================================================
```



```
PS C:\Users\jerry\Desktop\mastercourse\dataEngineer\pytestCourse\2classtest> pytest -s
================================================= test session starts =================================================
platform win32 -- Python 3.10.1, pytest-7.4.3, pluggy-1.3.0
rootdir: C:\Users\jerry\Desktop\mastercourse\dataEngineer\pytestCourse\2classtest
collected 1 item

tests\test_circle.py Setting up <bound method TestCircle.test_one of <tests.test_circle.TestCircle object at 0x0000021EF
D4E1C60>>
.Tearing down <bound method TestCircle.test_one of <tests.test_circle.TestCircle object at 0x0000021EFD4E1C60>>


================================================== 1 passed in 0.02s ==================================================
```

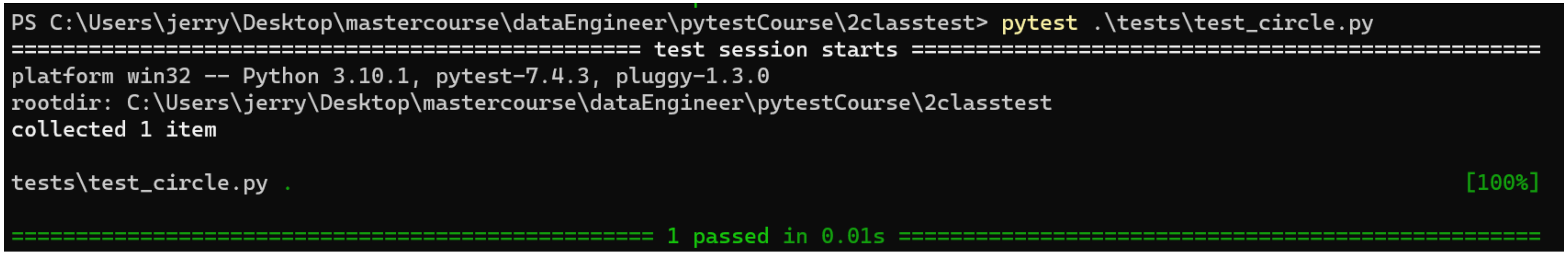
-s 是pytest的一個選項，它可以將測試過程中的標準輸出（stdout）和標準錯誤輸出（stderr）顯示在終端上

測試面積計算



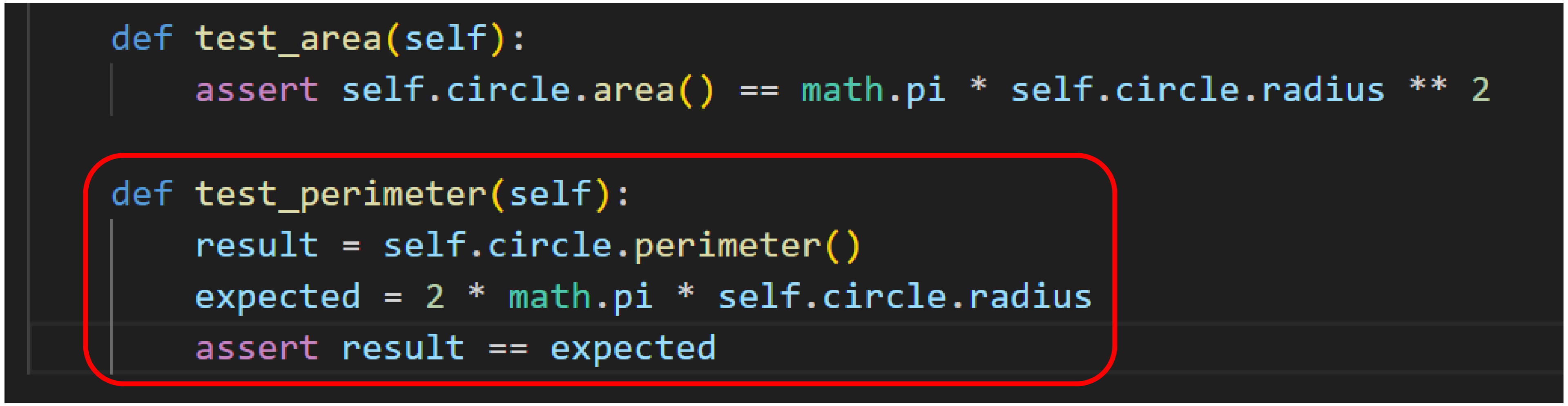
```python
import pytest
import math
import source.shapes as shapes

class TestCircle:

    def setup_method(self, method):
        print(f"Setting up {method}")
        self.circle = shapes.Circle(10)

    def teardown_method(self, method):
        print(f"Tearing down {method}")

    # def test_one(self):
    #     assert True

    def test_area(self):
        assert self.circle.area() == math.pi * self.circle.radius ** 2
```

成功

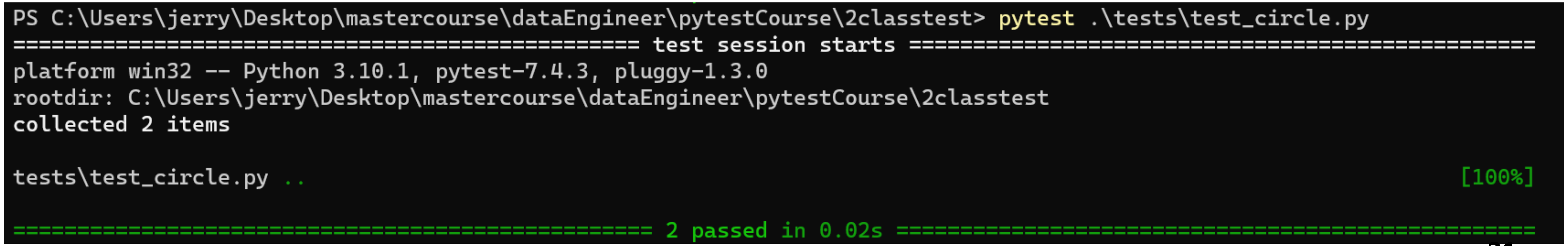


```
PS C:\Users\jerry\Desktop\mastercourse\dataEngineer\pytestCourse\2classtest> pytest .\tests\test_circle.py
================================================= test session starts =================================================
platform win32 -- Python 3.10.1, pytest-7.4.3, pluggy-1.3.0
rootdir: C:\Users\jerry\Desktop\mastercourse\dataEngineer\pytestCourse\2classtest
collected 1 item

tests\test_circle.py .                                                                                           [100%]

================================================== 1 passed in 0.01s ==================================================
```

測試周長



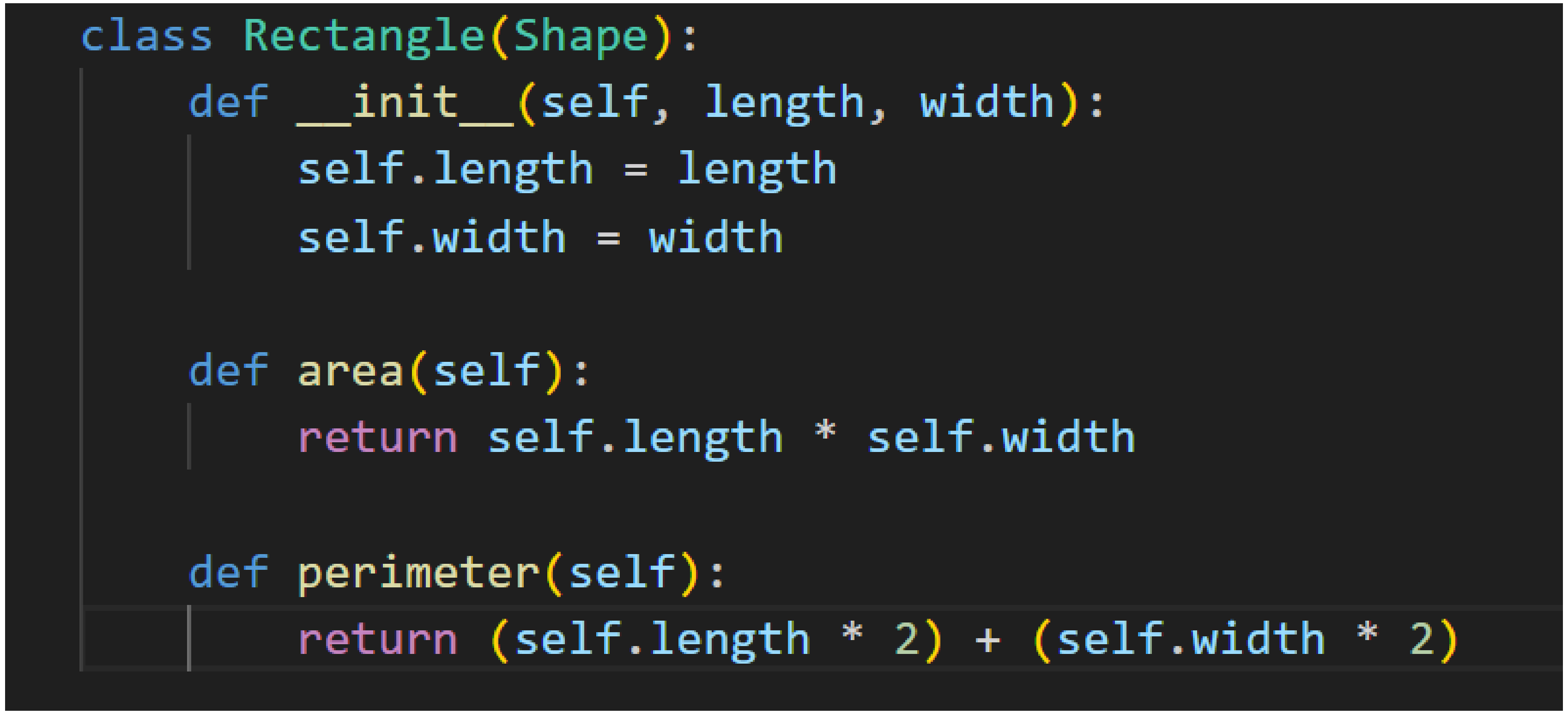
```
def test_area(self):
    assert self.circle.area() == math.pi * self.circle.radius ** 2

def test_perimeter(self):
    result = self.circle.perimeter()
    expected = 2 * math.pi * self.circle.radius
    assert result == expected
```



```
PS C:\Users\jerry\Desktop\mastercourse\dataEngineer\pytestCourse\2classtest> pytest .\tests\test_circle.py
================================================= test session starts =================================================
platform win32 -- Python 3.10.1, pytest-7.4.3, pluggy-1.3.0
rootdir: C:\Users\jerry\Desktop\mastercourse\dataEngineer\pytestCourse\2classtest
collected 2 items

tests\test_circle.py ..                                                                                          [100%]

================================================== 2 passed in 0.02s ==================================================
```

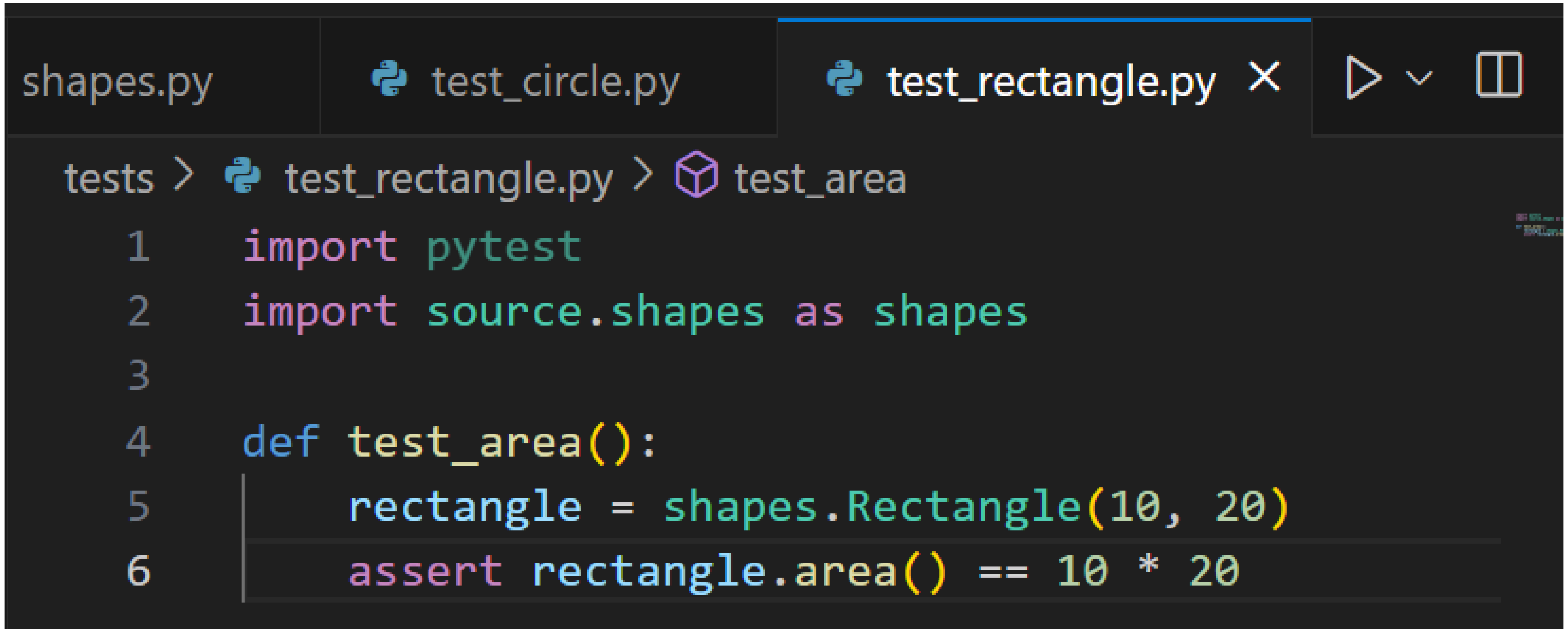
3. Fixtures

新撰寫一個計算長方形面積周長的class



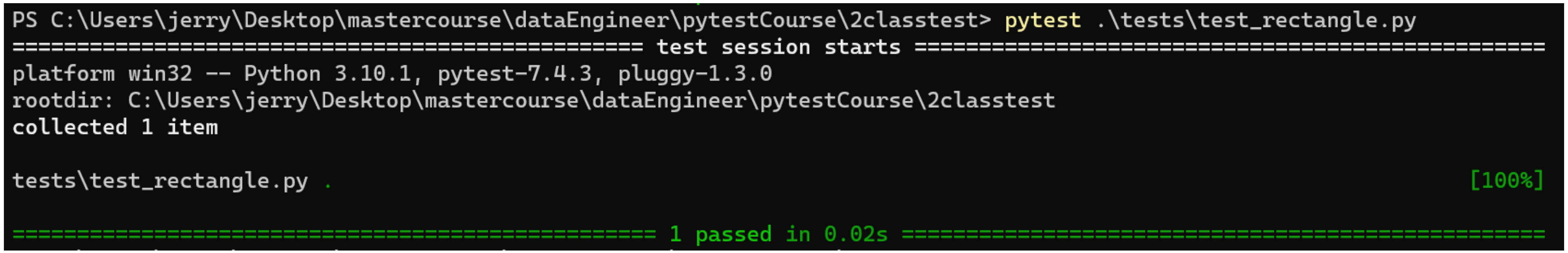
```
class Rectangle(Shape):
    def __init__(self, length, width):
        self.length = length
        self.width = width

    def area(self):
        return self.length * self.width

    def perimeter(self):
        return (self.length * 2) + (self.width * 2)
```

撰寫測試碼



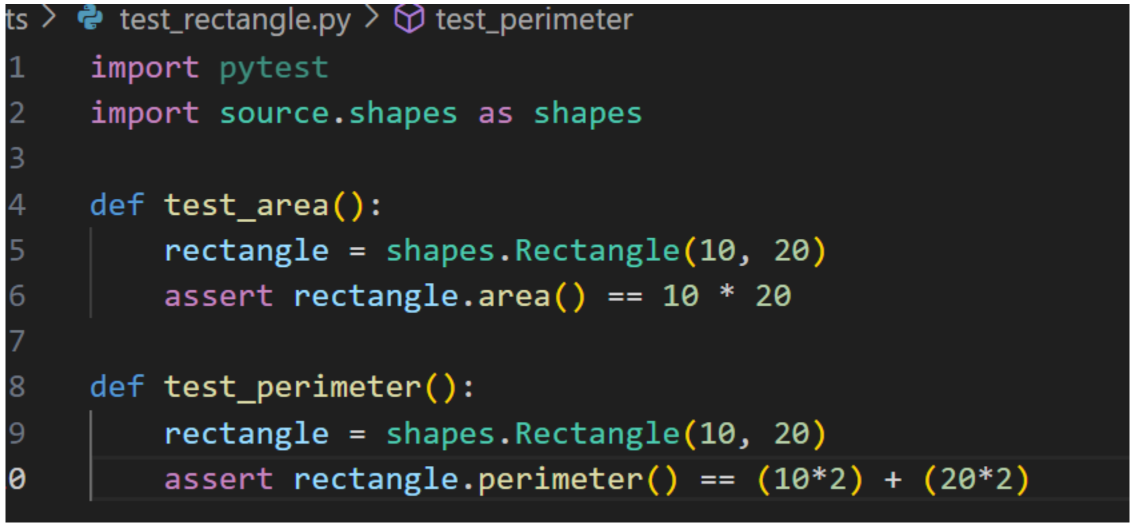
```python
import pytest
import source.shapes as shapes

def test_area():
    rectangle = shapes.Rectangle(10, 20)
    assert rectangle.area() == 10 * 20
```

成功

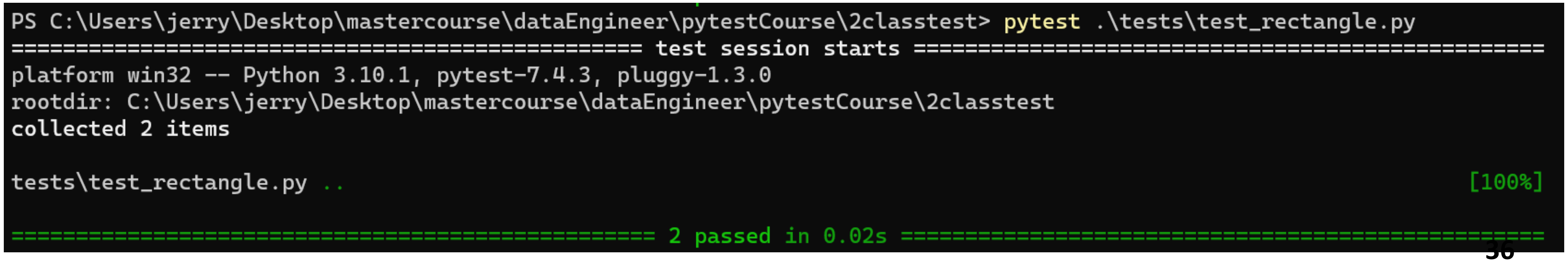


```
PS C:\Users\jerry\Desktop\mastercourse\dataEngineer\pytestCourse\2classtest> pytest .\tests\test_rectangle.py
================================================= test session starts =================================================
platform win32 -- Python 3.10.1, pytest-7.4.3, pluggy-1.3.0
rootdir: C:\Users\jerry\Desktop\mastercourse\dataEngineer\pytestCourse\2classtest
collected 1 item

tests\test_rectangle.py .                                                                                        [100%]

================================================== 1 passed in 0.02s ==================================================
```

測試周長與面積



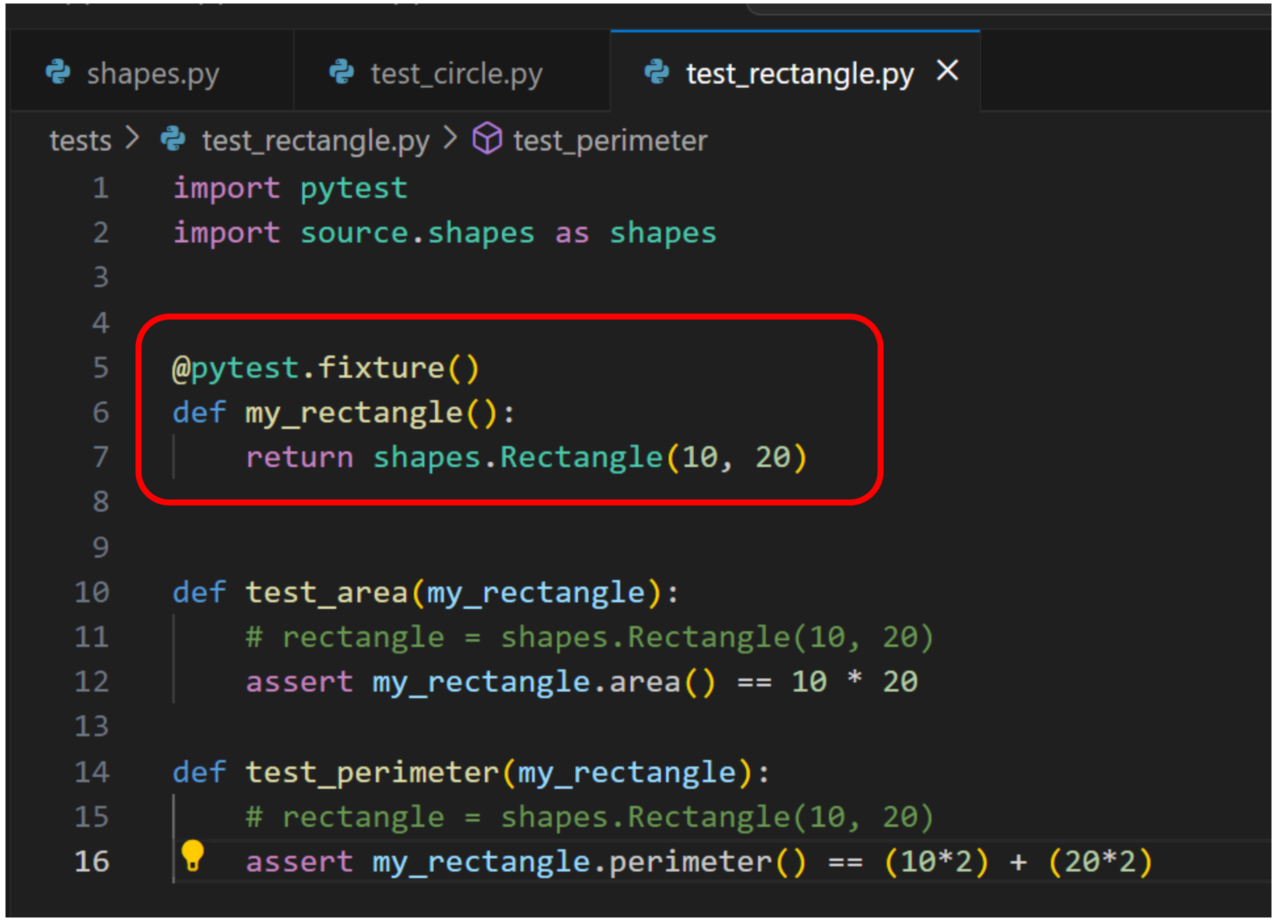
```
ts > test_rectangle.py > test_perimeter
import pytest
import source.shapes as shapes

def test_area():
    rectangle = shapes.Rectangle(10, 20)
    assert rectangle.area() == 10 * 20

def test_perimeter():
    rectangle = shapes.Rectangle(10, 20)
    assert rectangle.perimeter() == (10*2) + (20*2)
```



```
PS C:\Users\jerry\Desktop\mastercourse\dataEngineer\pytestCourse\2classtest> pytest .\tests\test_rectangle.py
================================================= test session starts =================================================
platform win32 -- Python 3.10.1, pytest-7.4.3, pluggy-1.3.0
rootdir: C:\Users\jerry\Desktop\mastercourse\dataEngineer\pytestCourse\2classtest
collected 2 items

tests\test_rectangle.py ..                                                                                      [100%]

================================================== 2 passed in 0.02s ==================================================
```

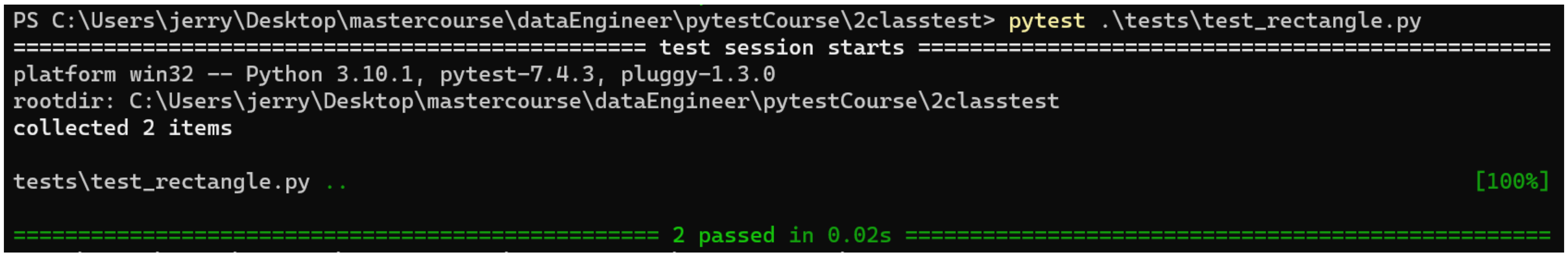
修改程式碼-Fixture

使用 fixture 先定義好測試參數，就可以重複呼叫，不需要每次都手動填寫參數



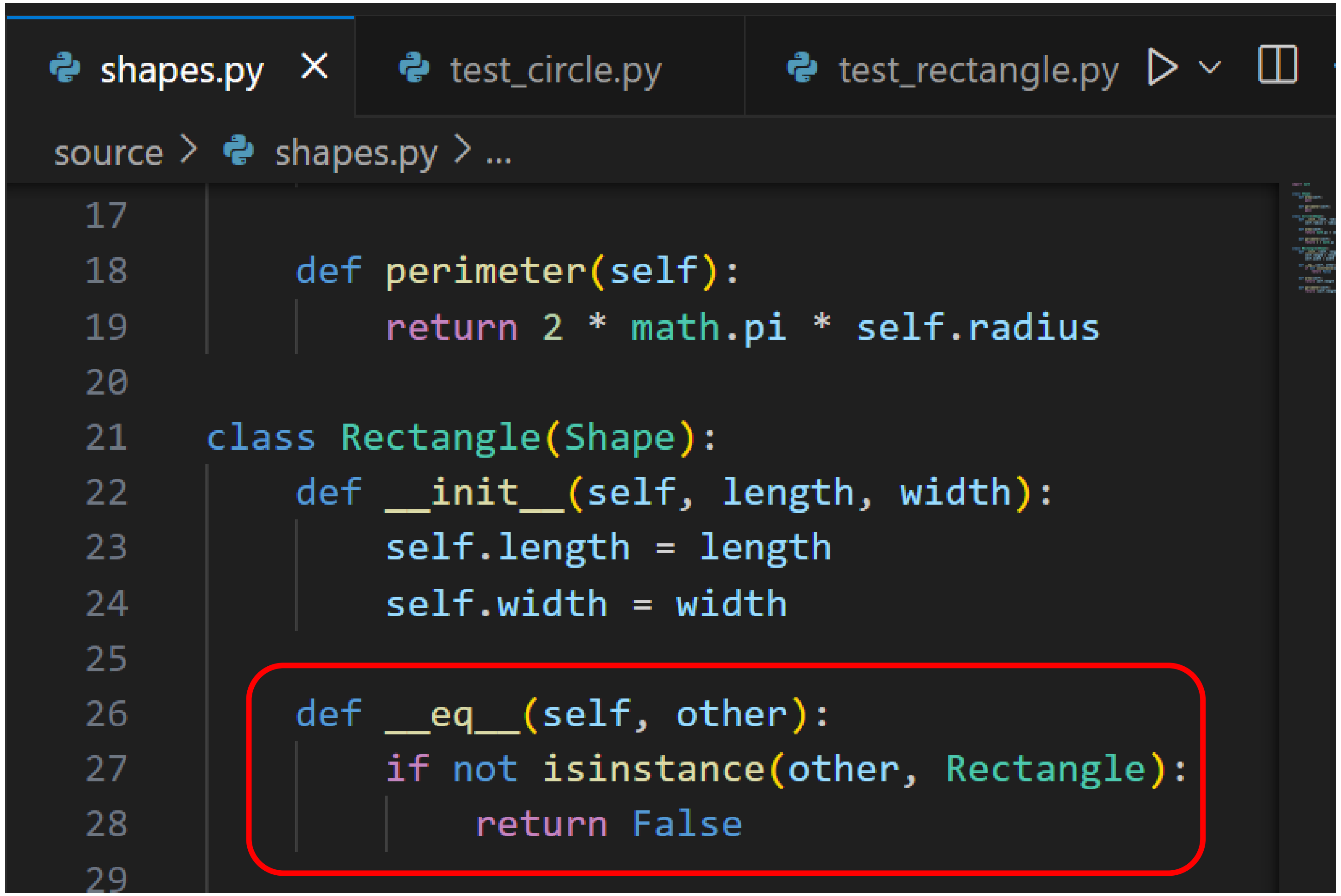
```python
import pytest
import source.shapes as shapes


@pytest.fixture()
def my_rectangle():
    return shapes.Rectangle(10, 20)


def test_area(my_rectangle):
    # rectangle = shapes.Rectangle(10, 20)
    assert my_rectangle.area() == 10 * 20

def test_perimeter(my_rectangle):
    # rectangle = shapes.Rectangle(10, 20)
    assert my_rectangle.perimeter() == (10*2) + (20*2)
```

成功



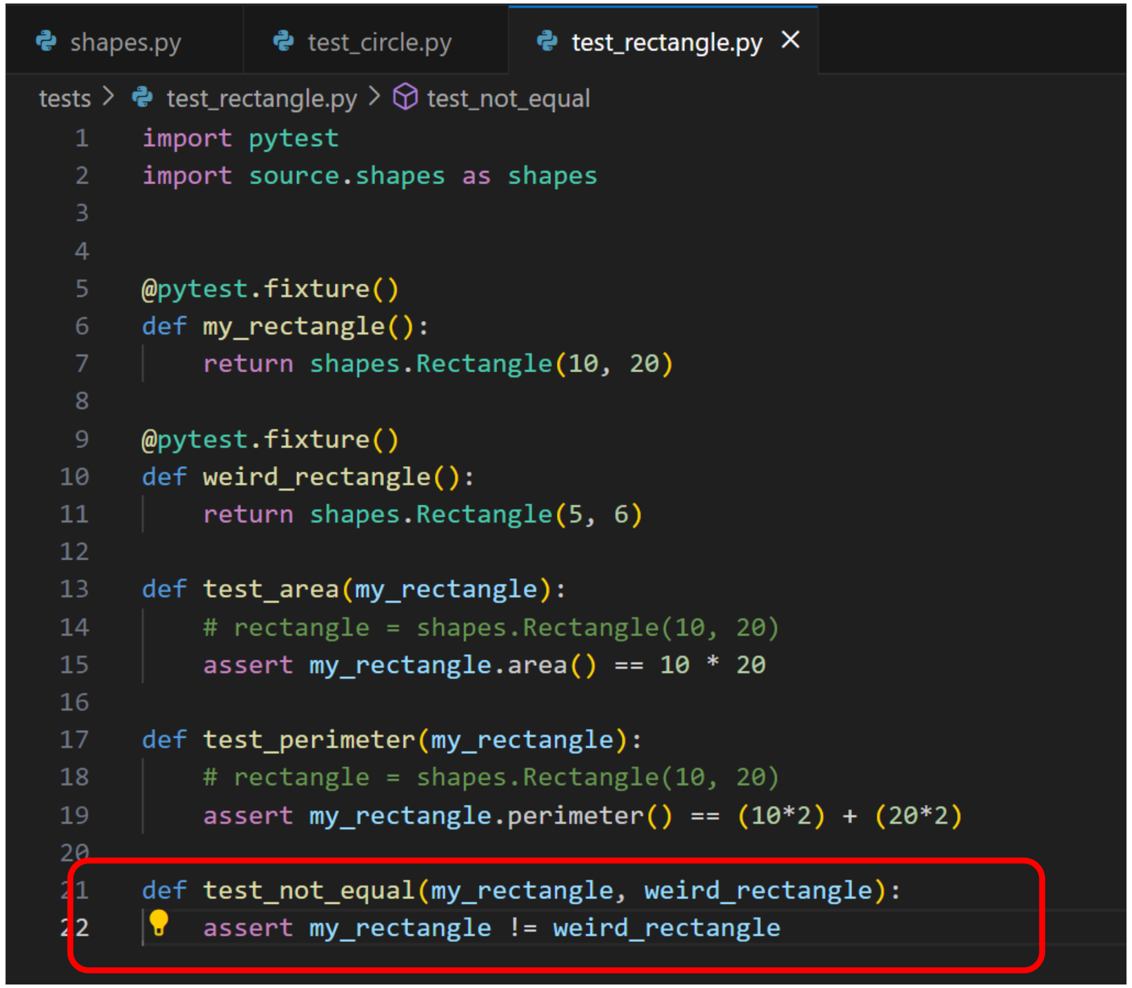
```
PS C:\Users\jerry\Desktop\mastercourse\dataEngineer\pytestCourse\2classtest> pytest .\tests\test_rectangle.py
================================================ test session starts ================================================
platform win32 -- Python 3.10.1, pytest-7.4.3, pluggy-1.3.0
rootdir: C:\Users\jerry\Desktop\mastercourse\dataEngineer\pytestCourse\2classtest
collected 2 items

tests\test_rectangle.py ..                                                                                     [100%]

================================================= 2 passed in 0.02s =================================================
```

在主程式碼新增一個 辨識是否為長方形的函式



```python

    def perimeter(self):
        return 2 * math.pi * self.radius

class Rectangle(Shape):
    def __init__(self, length, width):
        self.length = length
        self.width = width

    def __eq__(self, other):
        if not isinstance(other, Rectangle):
            return False

```

撰寫測試碼

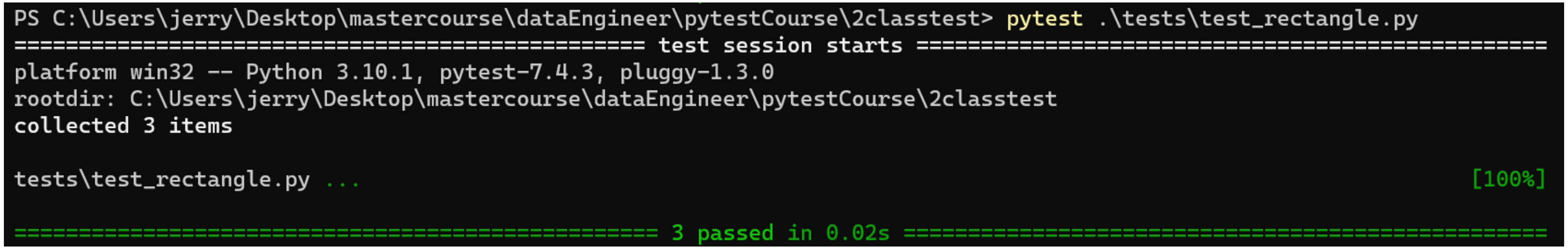


shapes.py test_circle.py test_rectangle.py ×

tests > test_rectangle.py > test_not_equal

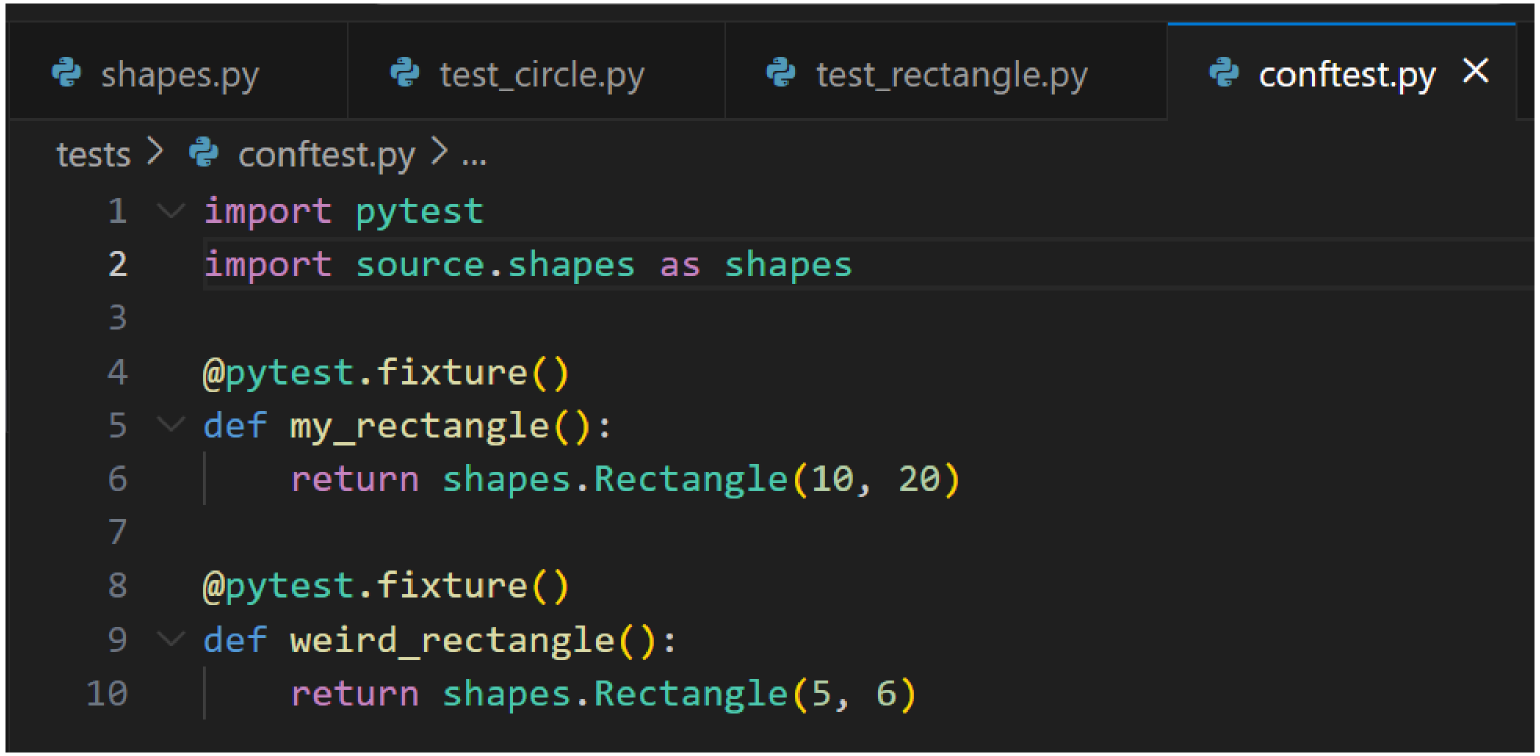
```python
import pytest
import source.shapes as shapes


@pytest.fixture()
def my_rectangle():
    return shapes.Rectangle(10, 20)

@pytest.fixture()
def weird_rectangle():
    return shapes.Rectangle(5, 6)

def test_area(my_rectangle):
    # rectangle = shapes.Rectangle(10, 20)
    assert my_rectangle.area() == 10 * 20

def test_perimeter(my_rectangle):
    # rectangle = shapes.Rectangle(10, 20)
    assert my_rectangle.perimeter() == (10*2) + (20*2)

def test_not_equal(my_rectangle, weird_rectangle):
    assert my_rectangle != weird_rectangle
```

成功



```
PS C:\Users\jerry\Desktop\mastercourse\dataEngineer\pytestCourse\2classtest> pytest .\tests\test_rectangle.py
================================================= test session starts =================================================
platform win32 -- Python 3.10.1, pytest-7.4.3, pluggy-1.3.0
rootdir: C:\Users\jerry\Desktop\mastercourse\dataEngineer\pytestCourse\2classtest
collected 3 items

tests\test_rectangle.py ...                                                                                      [100%]

================================================== 3 passed in 0.02s ==================================================
```

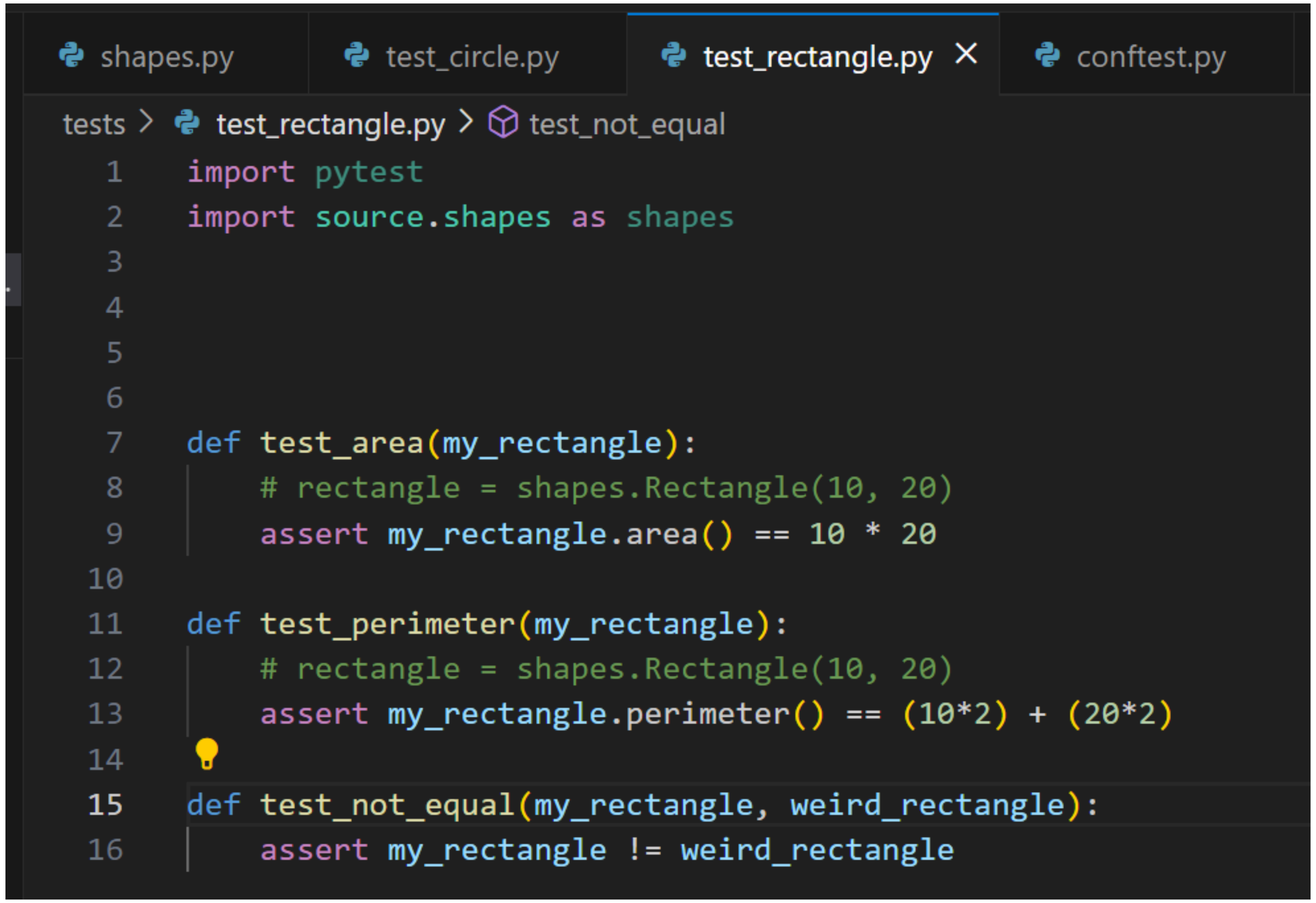
將Fixture獨立於一個檔案中



```python
import pytest
import source.shapes as shapes

@pytest.fixture()
def my_rectangle():
    return shapes.Rectangle(10, 20)

@pytest.fixture()
def weird_rectangle():
    return shapes.Rectangle(5, 6)
```

- 在前面Fixture與測試碼都混和在一起，太多可能導致程式碼混亂。
- 新增一個 名為 conftest.py
- 切記檔名必須為 conftest
- pytest會自動偵測檔名

測試碼保留原本的測試函式

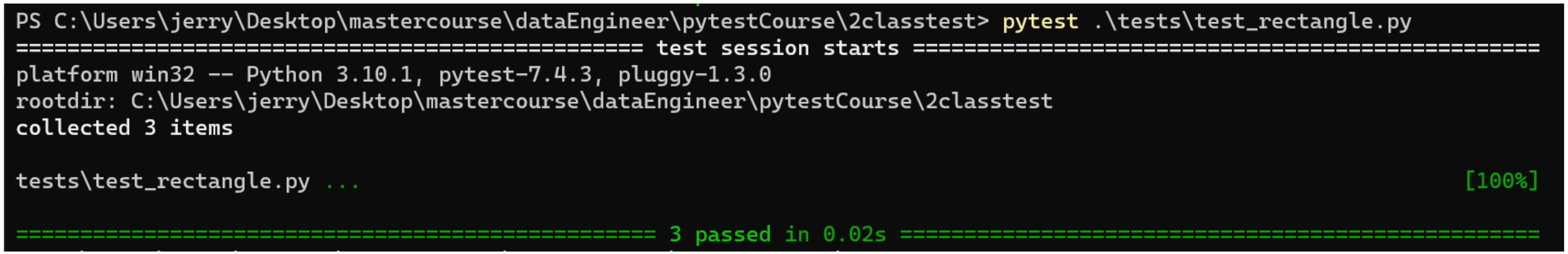


shapes.py | test_circle.py | test_rectangle.py | conftest.py

tests > test_rectangle.py > test_not_equal

```python
import pytest
import source.shapes as shapes




def test_area(my_rectangle):
    # rectangle = shapes.Rectangle(10, 20)
    assert my_rectangle.area() == 10 * 20

def test_perimeter(my_rectangle):
    # rectangle = shapes.Rectangle(10, 20)
    assert my_rectangle.perimeter() == (10*2) + (20*2)

def test_not_equal(my_rectangle, weird_rectangle):
    assert my_rectangle != weird_rectangle
```

成功



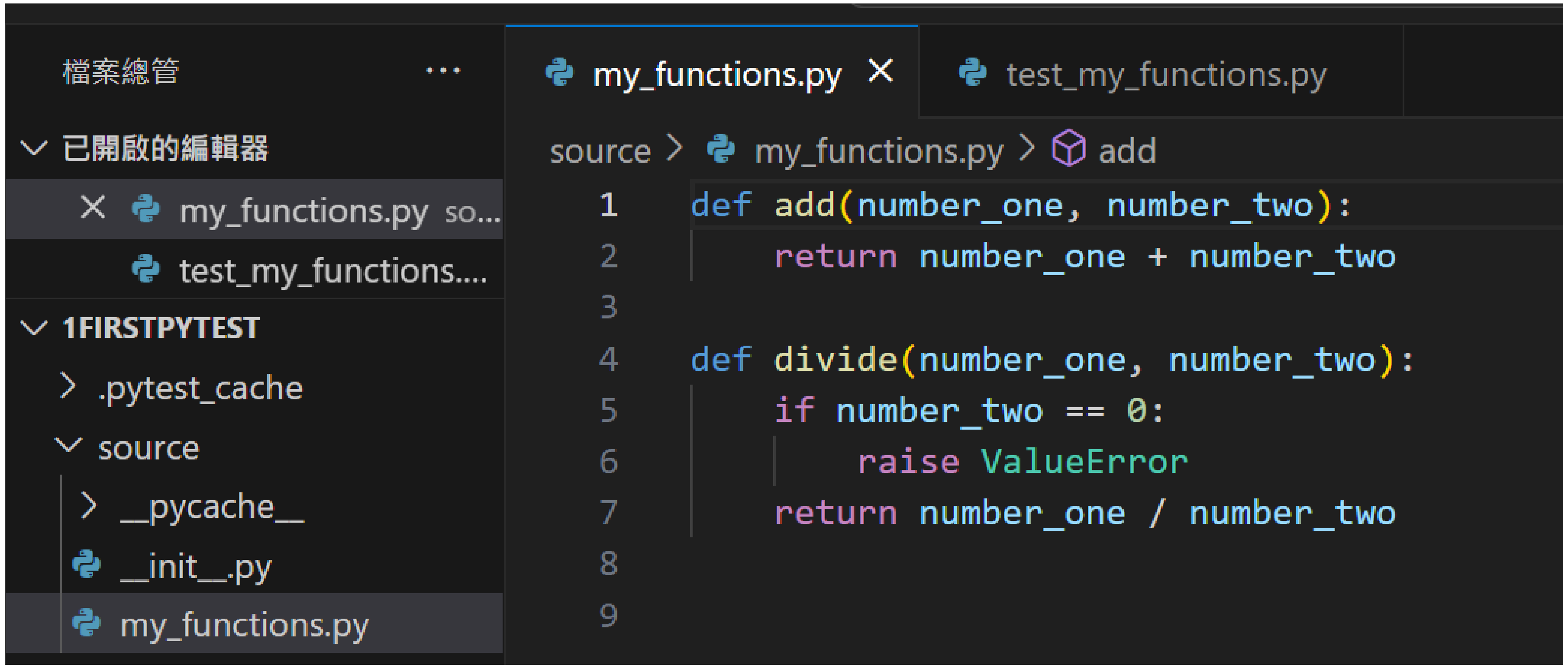
```
PS C:\Users\jerry\Desktop\mastercourse\dataEngineer\pytestCourse\2classtest> pytest .\tests\test_rectangle.py
================================================= test session starts =================================================
platform win32 -- Python 3.10.1, pytest-7.4.3, pluggy-1.3.0
rootdir: C:\Users\jerry\Desktop\mastercourse\dataEngineer\pytestCourse\2classtest
collected 3 items

tests\test_rectangle.py ...                                                                                      [100%]

================================================== 3 passed in 0.02s ==================================================
```

4. Mark & Parametrize

說明

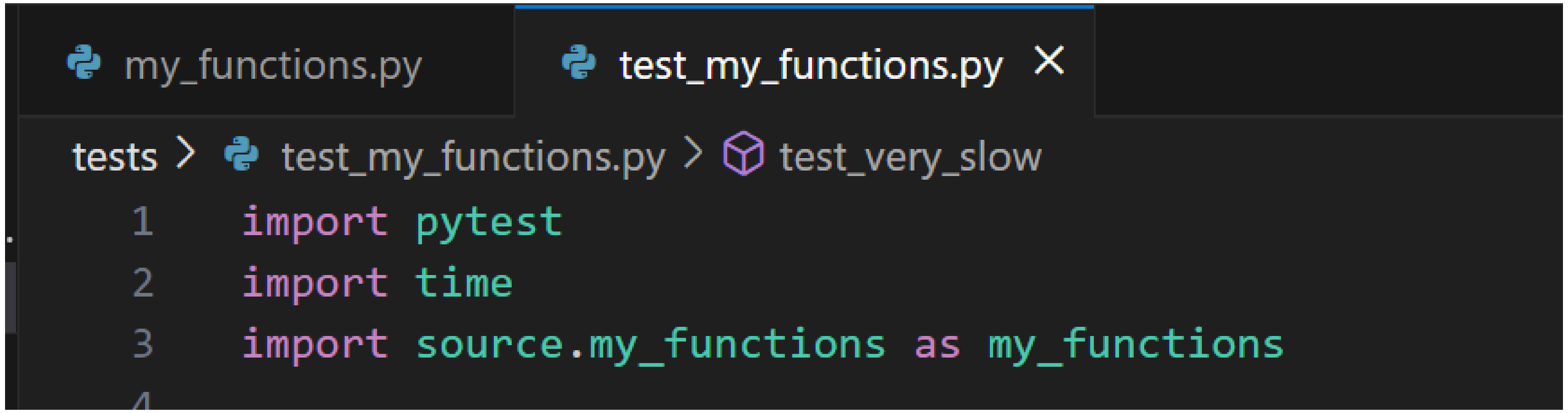
- usefixtures：在測試函數或類上使用 fixtures。
- filterwarnings：過濾某些測試函數的警告。
- skip：總是跳過測試函數。
- skipif：如果滿足某些條件，則跳過測試函數。
- xfail：如果滿足某些條件，則產生“預期失敗”的結果。
- parametrize：對同一個測試函數執行多次調用。

回到 my_functions.py

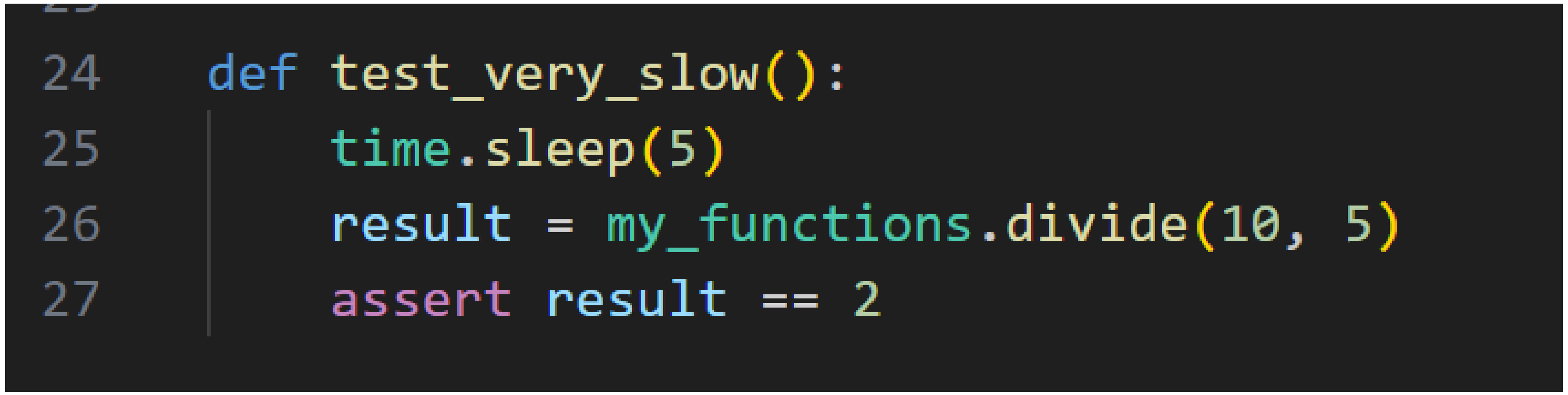


```python
def add(number_one, number_two):
    return number_one + number_two

def divide(number_one, number_two):
    if number_two == 0:
        raise ValueError
    return number_one / number_two
```

新增 time 作為後續範例使用

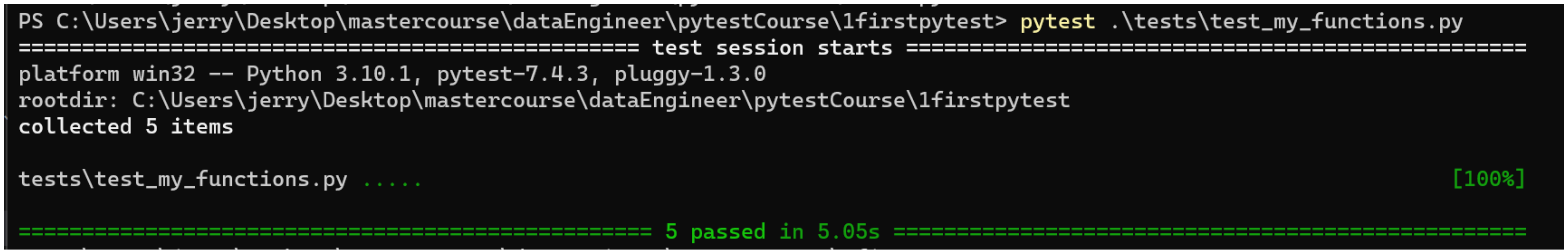


```python
import pytest
import time
import source.my_functions as my_functions
```



```python
def test_very_slow():
    time.sleep(5)
    result = my_functions.divide(10, 5)
    assert result == 2
```

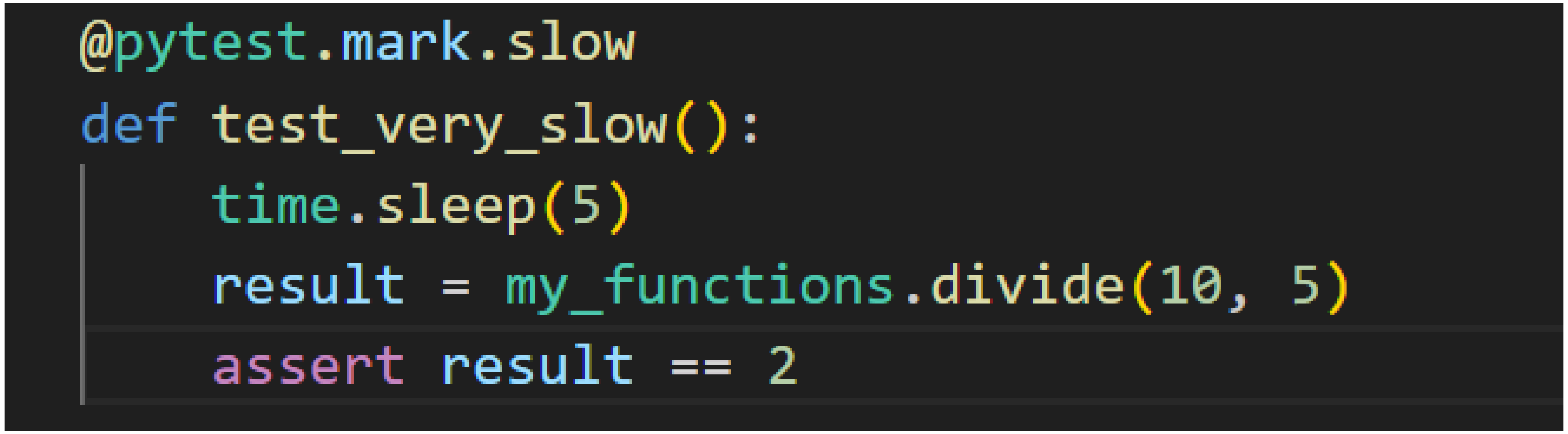
成功



```
PS C:\Users\jerry\Desktop\mastercourse\dataEngineer\pytestCourse\1firstpytest> pytest .\tests\test_my_functions.py
================================================= test session starts =================================================
platform win32 -- Python 3.10.1, pytest-7.4.3, pluggy-1.3.0
rootdir: C:\Users\jerry\Desktop\mastercourse\dataEngineer\pytestCourse\1firstpytest
collected 5 items

tests\test_my_functions.py .....                                                                                 [100%]

================================================== 5 passed in 5.05s ==================================================
```

使用 mark

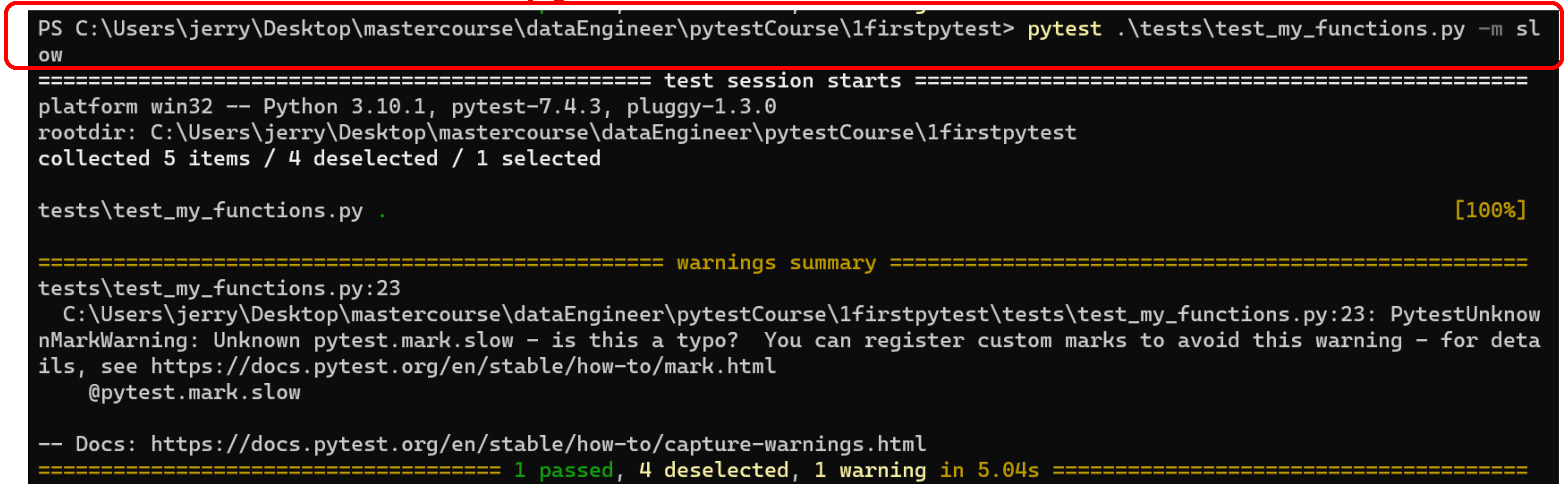
@pytest.mark.[name]



```python
@pytest.mark.slow
def test_very_slow():
    time.sleep(5)
    result = my_functions.divide(10, 5)
    assert result == 2
```

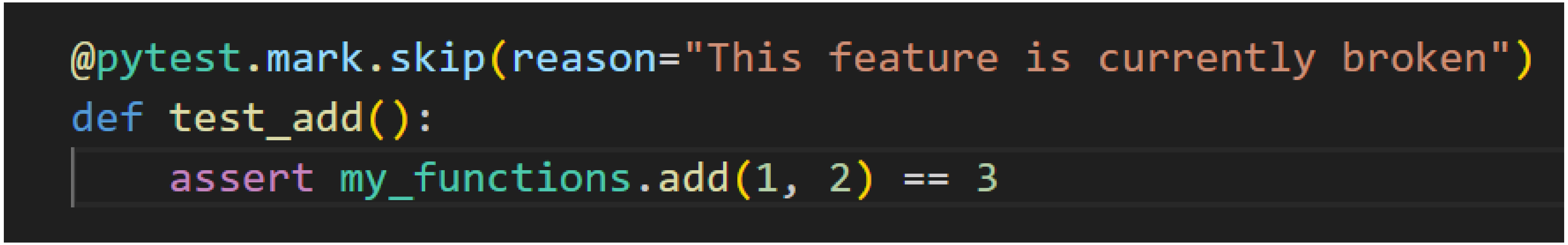
成功

測試的時候即可直接在終端輸入指令:
pytest –m slow



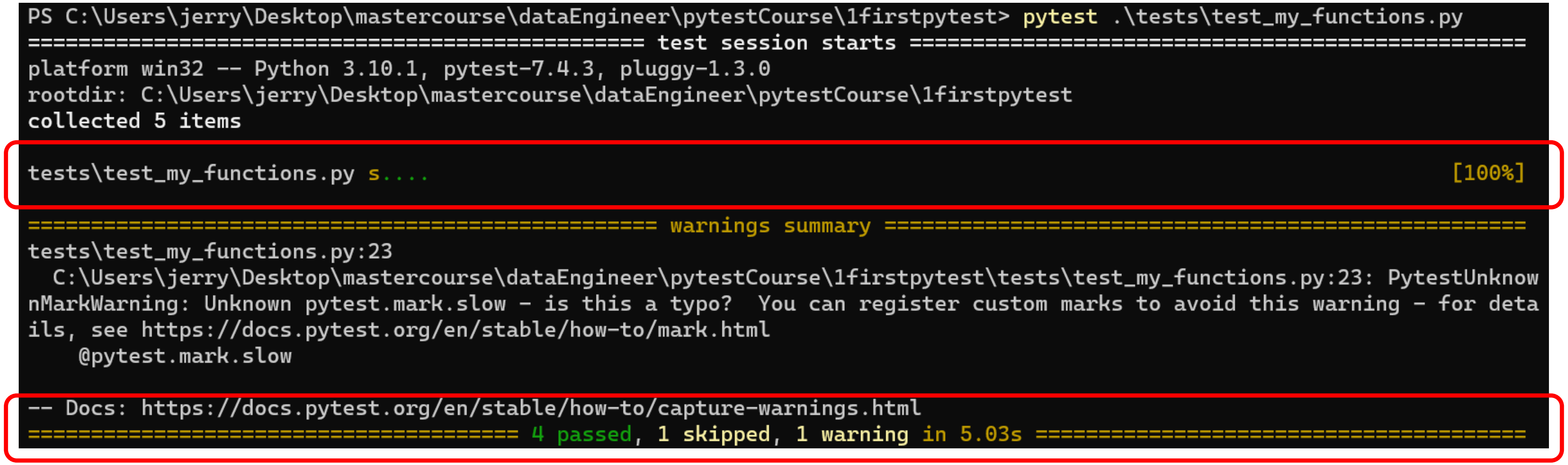
```
PS C:\Users\jerry\Desktop\mastercourse\dataEngineer\pytestCourse\1firstpytest> pytest .\tests\test_my_functions.py -m sl
ow
================================================== test session starts ===================================================
platform win32 -- Python 3.10.1, pytest-7.4.3, pluggy-1.3.0
rootdir: C:\Users\jerry\Desktop\mastercourse\dataEngineer\pytestCourse\1firstpytest
collected 5 items / 4 deselected / 1 selected

tests\test_my_functions.py .                                                                                        [100%]

=================================================== warnings summary ====================================================
tests\test_my_functions.py:23
  C:\Users\jerry\Desktop\mastercourse\dataEngineer\pytestCourse\1firstpytest\tests\test_my_functions.py:23: PytestUnknow
nMarkWarning: Unknown pytest.mark.slow - is this a typo?  You can register custom marks to avoid this warning - for deta
ils, see https://docs.pytest.org/en/stable/how-to/mark.html
    @pytest.mark.slow

-- Docs: https://docs.pytest.org/en/stable/how-to/capture-warnings.html
===================================== 1 passed, 4 deselected, 1 warning in 5.04s ======================================
```

使用skip



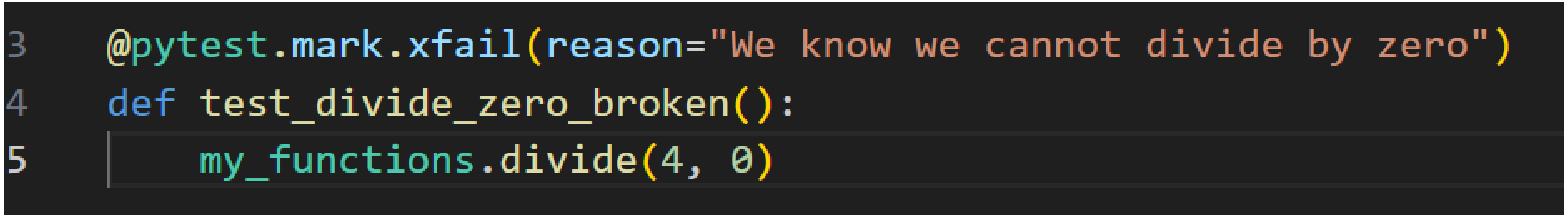
```
@pytest.mark.skip(reason="This feature is currently broken")
def test_add():
    assert my_functions.add(1, 2) == 3
```

成功



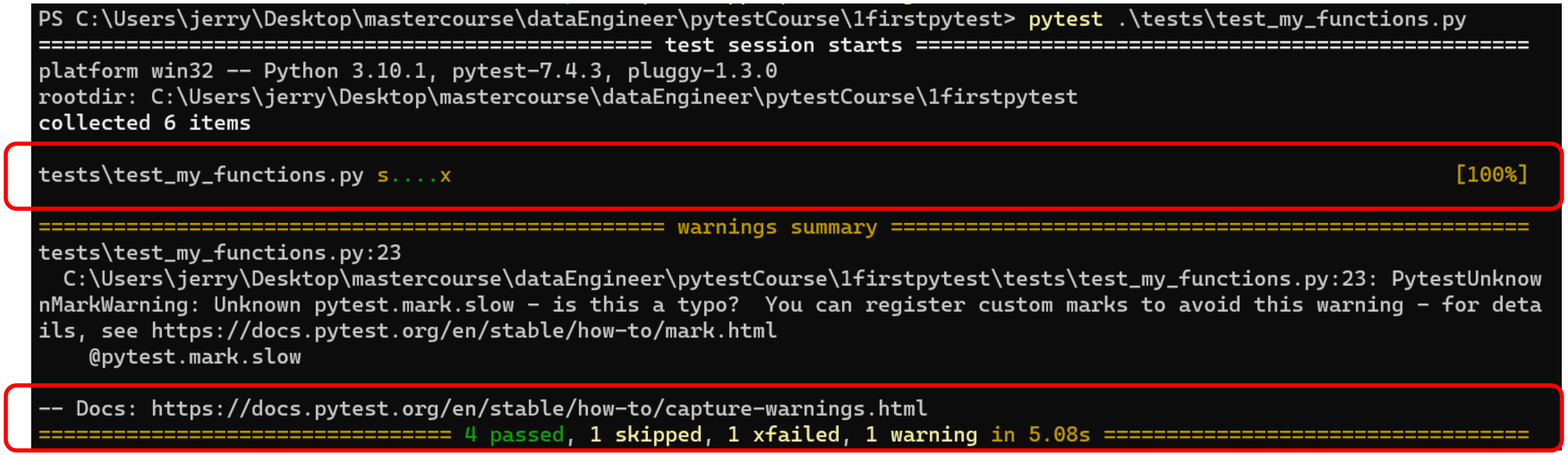
```
PS C:\Users\jerry\Desktop\mastercourse\dataEngineer\pytestCourse\1firstpytest> pytest .\tests\test_my_functions.py
================================================= test session starts ==================================================
platform win32 -- Python 3.10.1, pytest-7.4.3, pluggy-1.3.0
rootdir: C:\Users\jerry\Desktop\mastercourse\dataEngineer\pytestCourse\1firstpytest
collected 5 items

tests\test_my_functions.py s....                                                                                  [100%]

=================================================== warnings summary ===================================================
tests\test_my_functions.py:23
  C:\Users\jerry\Desktop\mastercourse\dataEngineer\pytestCourse\1firstpytest\tests\test_my_functions.py:23: PytestUnknow
nMarkWarning: Unknown pytest.mark.slow - is this a typo?  You can register custom marks to avoid this warning - for deta
ils, see https://docs.pytest.org/en/stable/how-to/mark.html
    @pytest.mark.slow

-- Docs: https://docs.pytest.org/en/stable/how-to/capture-warnings.html
======================================= 4 passed, 1 skipped, 1 warning in 5.03s ========================================
```

使用 xfail



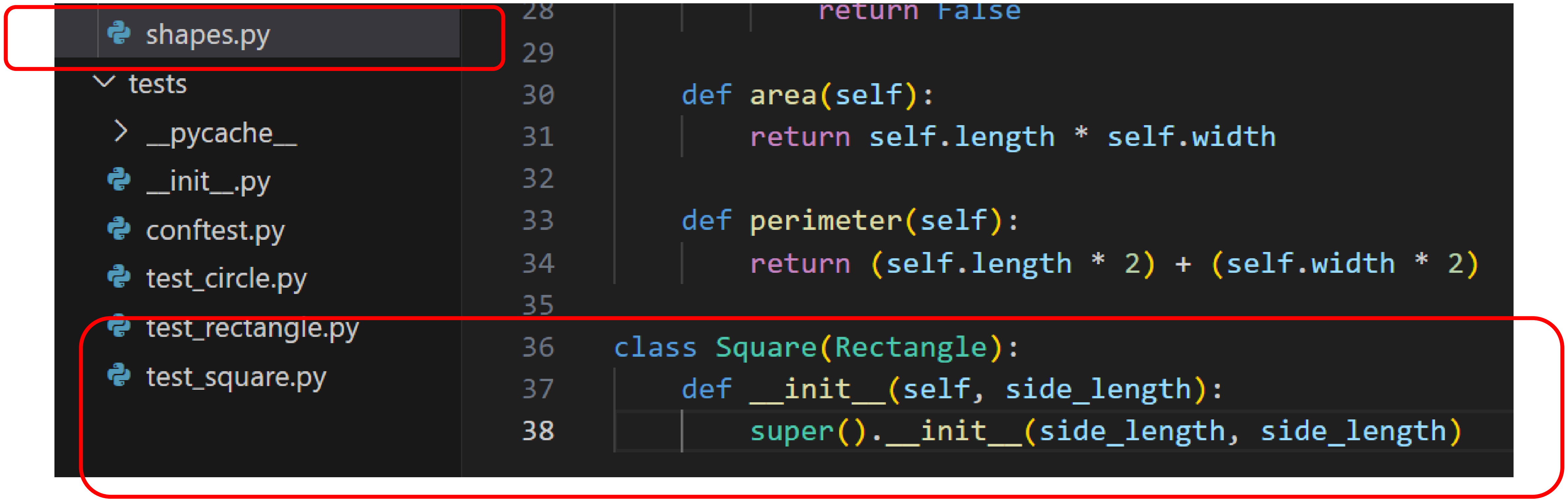
```python
@pytest.mark.xfail(reason="We know we cannot divide by zero")
def test_divide_zero_broken():
    my_functions.divide(4, 0)
```

成功



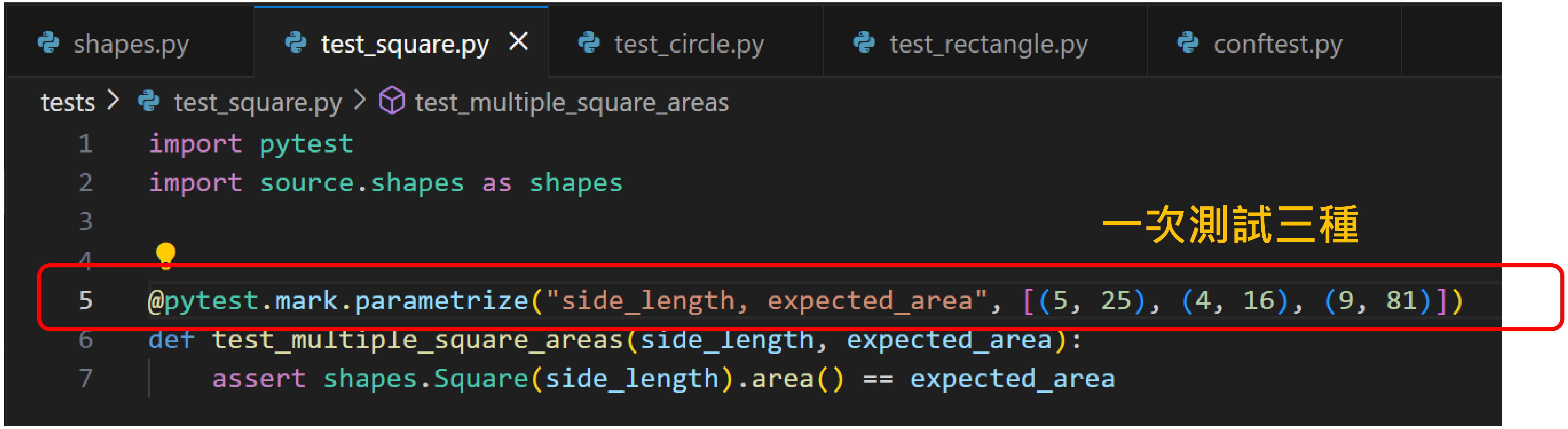
```
PS C:\Users\jerry\Desktop\mastercourse\dataEngineer\pytestCourse\1firstpytest> pytest .\tests\test_my_functions.py
=================================================== test session starts ===================================================
platform win32 -- Python 3.10.1, pytest-7.4.3, pluggy-1.3.0
rootdir: C:\Users\jerry\Desktop\mastercourse\dataEngineer\pytestCourse\1firstpytest
collected 6 items

tests\test_my_functions.py s....x                                                                                  [100%]

==================================================== warnings summary =====================================================
tests\test_my_functions.py:23
  C:\Users\jerry\Desktop\mastercourse\dataEngineer\pytestCourse\1firstpytest\tests\test_my_functions.py:23: PytestUnknow
nMarkWarning: Unknown pytest.mark.slow - is this a typo?  You can register custom marks to avoid this warning - for deta
ils, see https://docs.pytest.org/en/stable/how-to/mark.html
    @pytest.mark.slow

-- Docs: https://docs.pytest.org/en/stable/how-to/capture-warnings.html
================================= 4 passed, 1 skipped, 1 xfailed, 1 warning in 5.08s ==================================
```

回到 shapes.py 撰寫 class Square



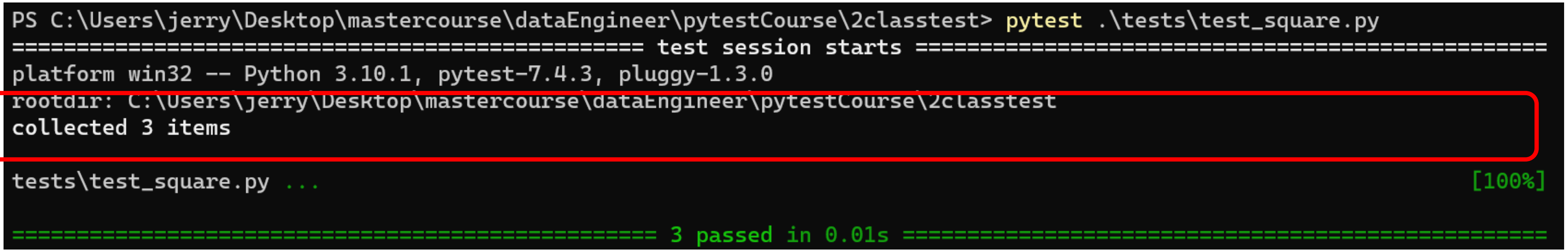
```python
                return False

    def area(self):
        return self.length * self.width

    def perimeter(self):
        return (self.length * 2) + (self.width * 2)

class Square(Rectangle):
    def __init__(self, side_length):
        super().__init__(side_length, side_length)
```

使用 parametrize 一次測試多種



```python
import pytest
import source.shapes as shapes


@pytest.mark.parametrize("side_length, expected_area", [(5, 25), (4, 16), (9, 81)])
def test_multiple_square_areas(side_length, expected_area):
    assert shapes.Square(side_length).area() == expected_area
```

成功



```
PS C:\Users\jerry\Desktop\mastercourse\dataEngineer\pytestCourse\2classtest> pytest .\tests\test_square.py
================================================= test session starts =================================================
platform win32 -- Python 3.10.1, pytest-7.4.3, pluggy-1.3.0
rootdir: C:\Users\jerry\Desktop\mastercourse\dataEngineer\pytestCourse\2classtest
collected 3 items

tests\test_square.py ...                                                                                        [100%]

================================================== 3 passed in 0.01s ==================================================
```

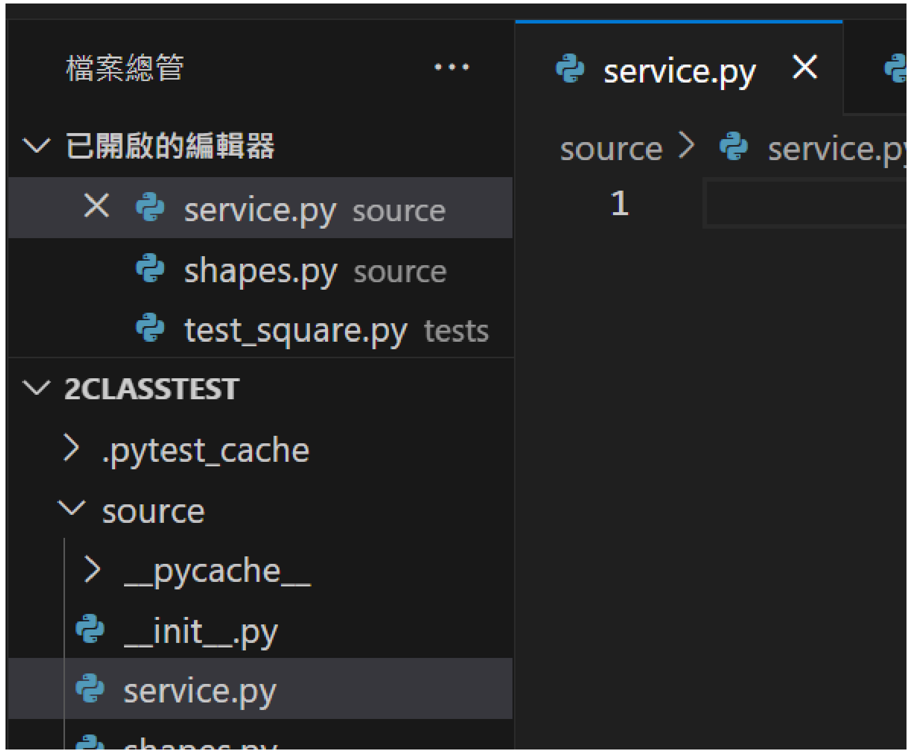
5. Mocking

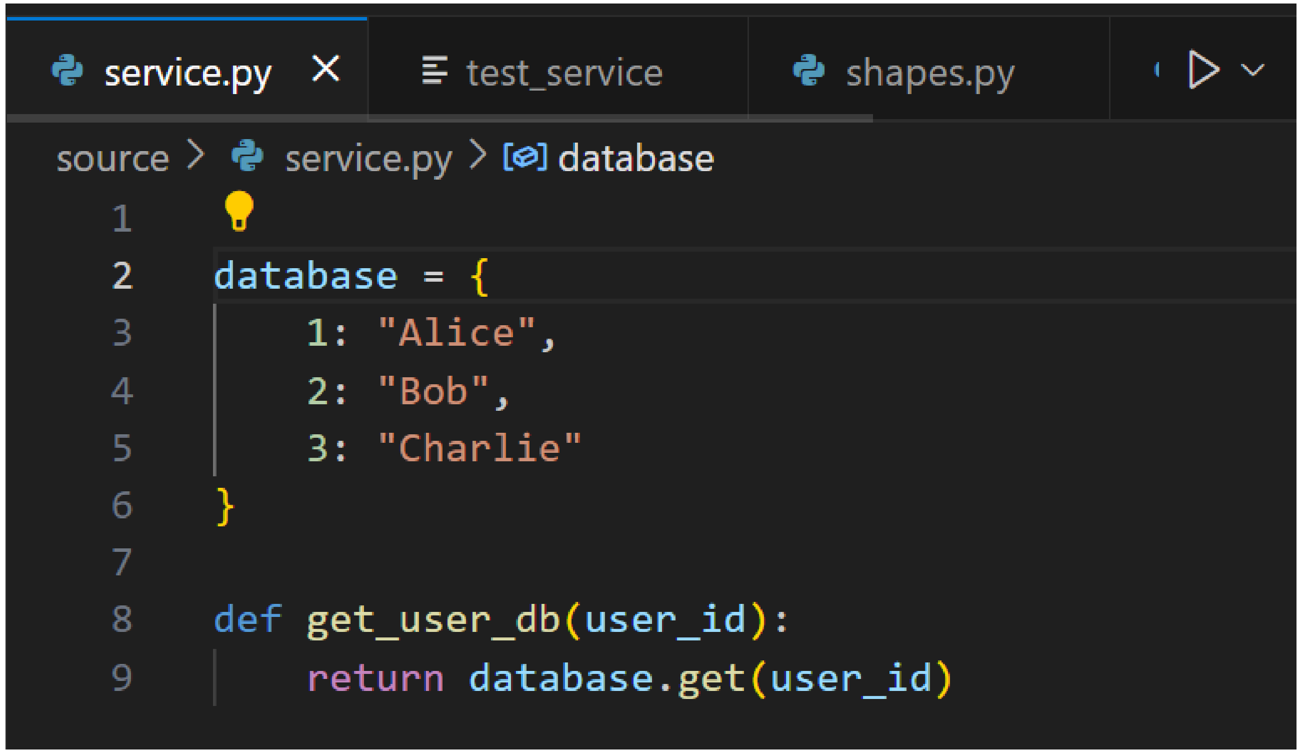
說明

- 模擬API測試，使用真正的API可能會需要付費或是遇到一些不可抗力的因素(ex: API網站故障、維護等等)導致API無法順利使用。

- 利用模擬API測試的方式則可以避免這樣的情形，增加測試效率。

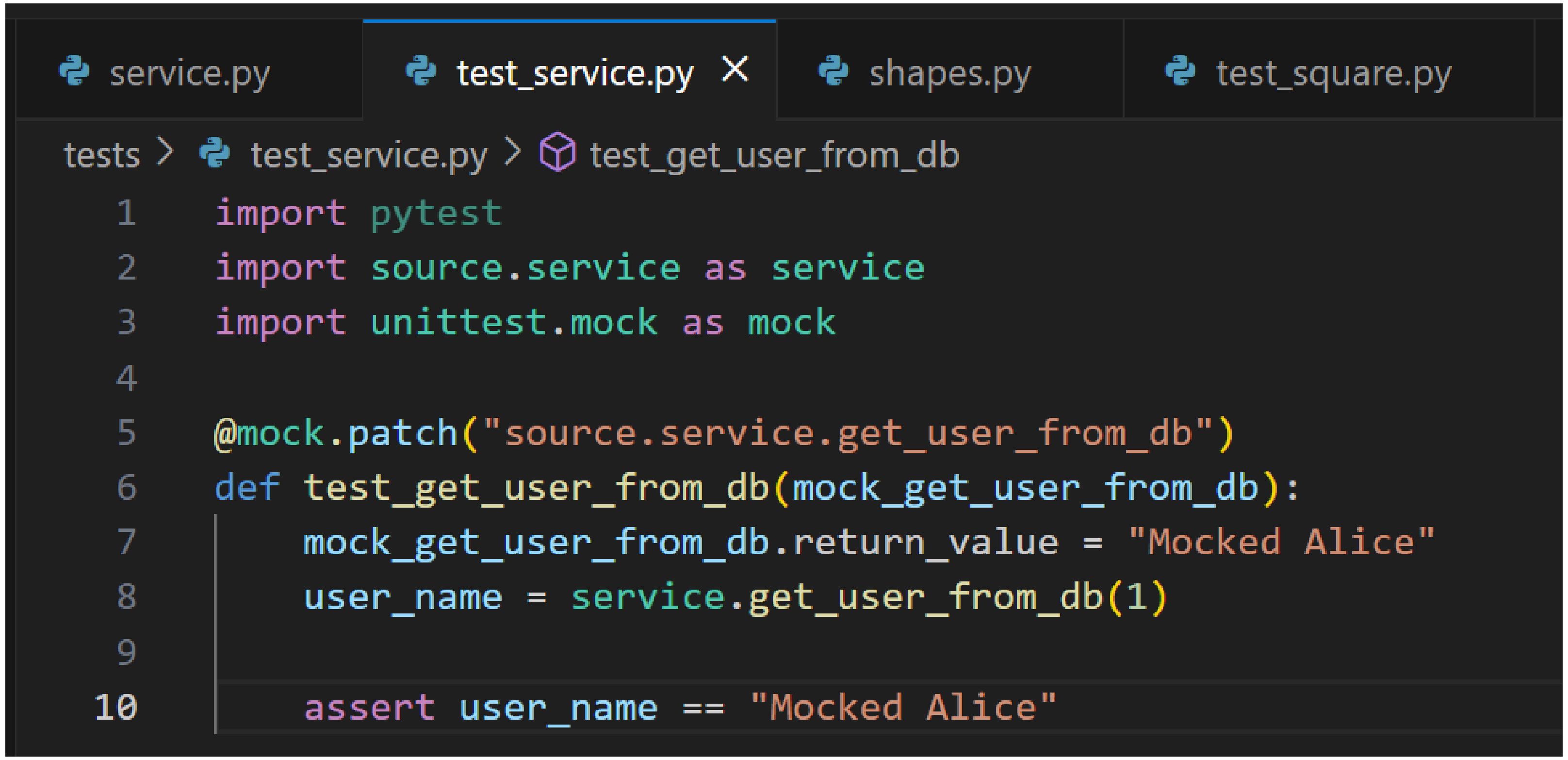
在source資料夾中先建立一個 service.py





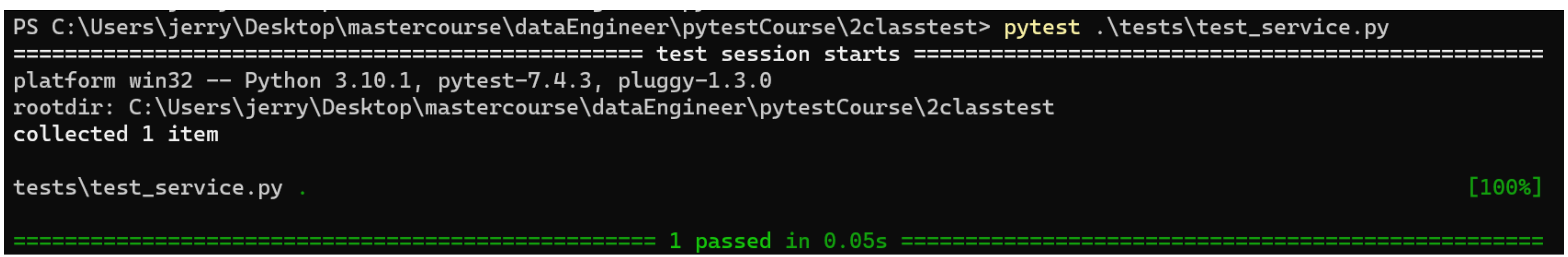
```python

database = {
    1: "Alice",
    2: "Bob",
    3: "Charlie"
}

def get_user_db(user_id):
    return database.get(user_id)
```

撰寫 test_service.py



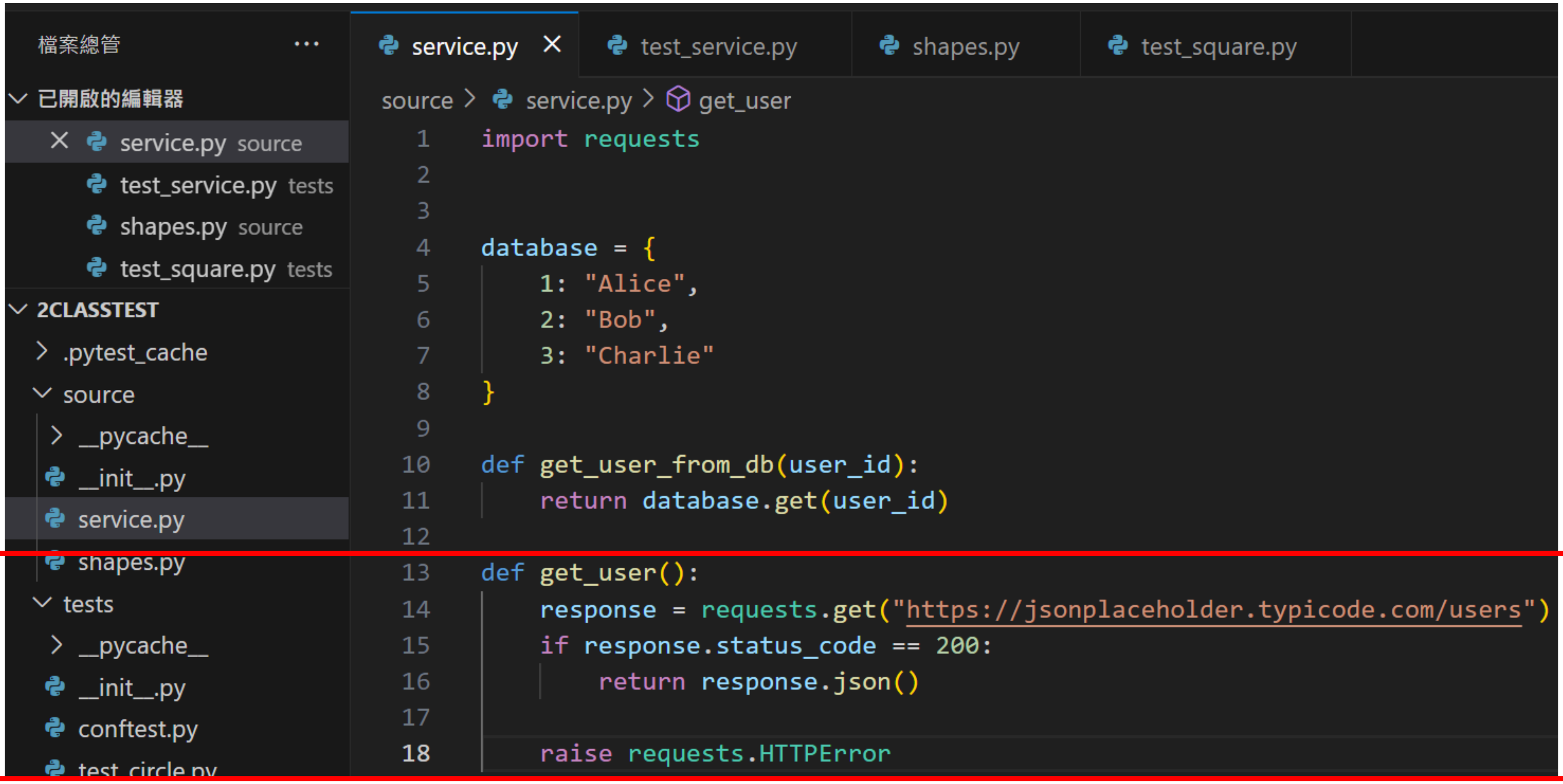
```python
import pytest
import source.service as service
import unittest.mock as mock

@mock.patch("source.service.get_user_from_db")
def test_get_user_from_db(mock_get_user_from_db):
    mock_get_user_from_db.return_value = "Mocked Alice"
    user_name = service.get_user_from_db(1)

    assert user_name == "Mocked Alice"
```

成功



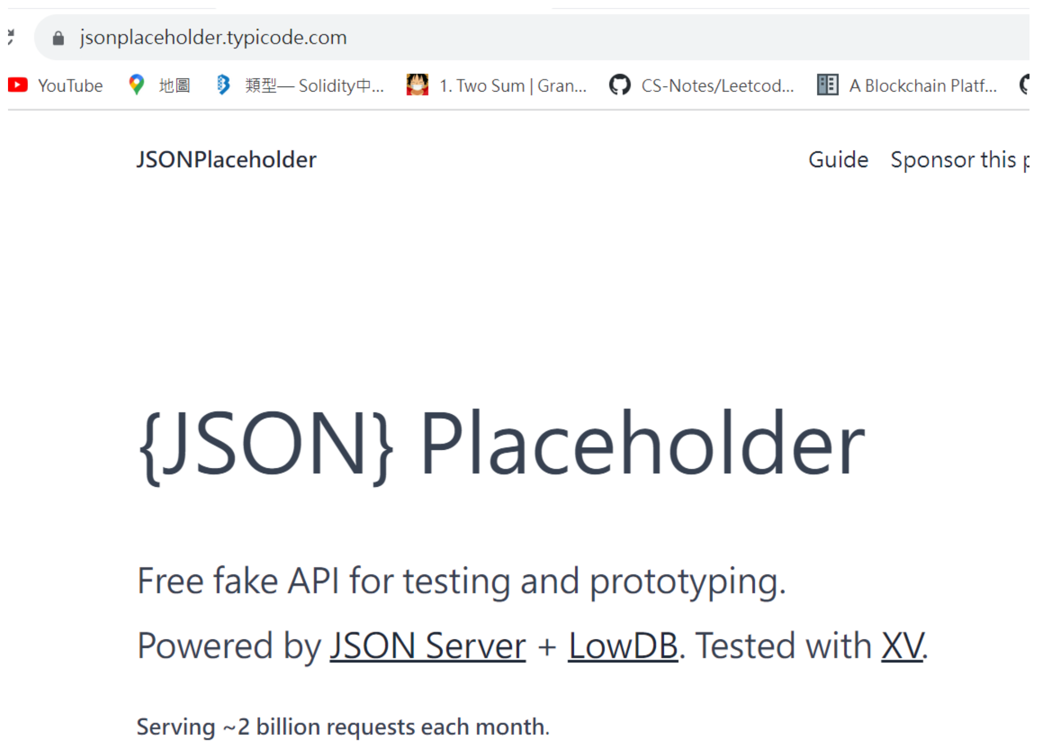
```
PS C:\Users\jerry\Desktop\mastercourse\dataEngineer\pytestCourse\2classtest> pytest .\tests\test_service.py
================================================= test session starts =================================================
platform win32 -- Python 3.10.1, pytest-7.4.3, pluggy-1.3.0
rootdir: C:\Users\jerry\Desktop\mastercourse\dataEngineer\pytestCourse\2classtest
collected 1 item

tests\test_service.py .                                                                                          [100%]

================================================== 1 passed in 0.05s ==================================================
```

使用 jsonplaceholider 網站API實際測試



```python
import requests


database = {
    1: "Alice",
    2: "Bob",
    3: "Charlie"
}

def get_user_from_db(user_id):
    return database.get(user_id)

def get_user():
    response = requests.get("https://jsonplaceholder.typicode.com/users")
    if response.status_code == 200:
        return response.json()

    raise requests.HTTPError
```

API產生出的範例使用者資料


jsonplaceholder.typicode.com

YouTube 地圖 類型— Solidity中... 1. Two Sum | Gran... CS-Notes/Leetcod... A Blockchain Platf...

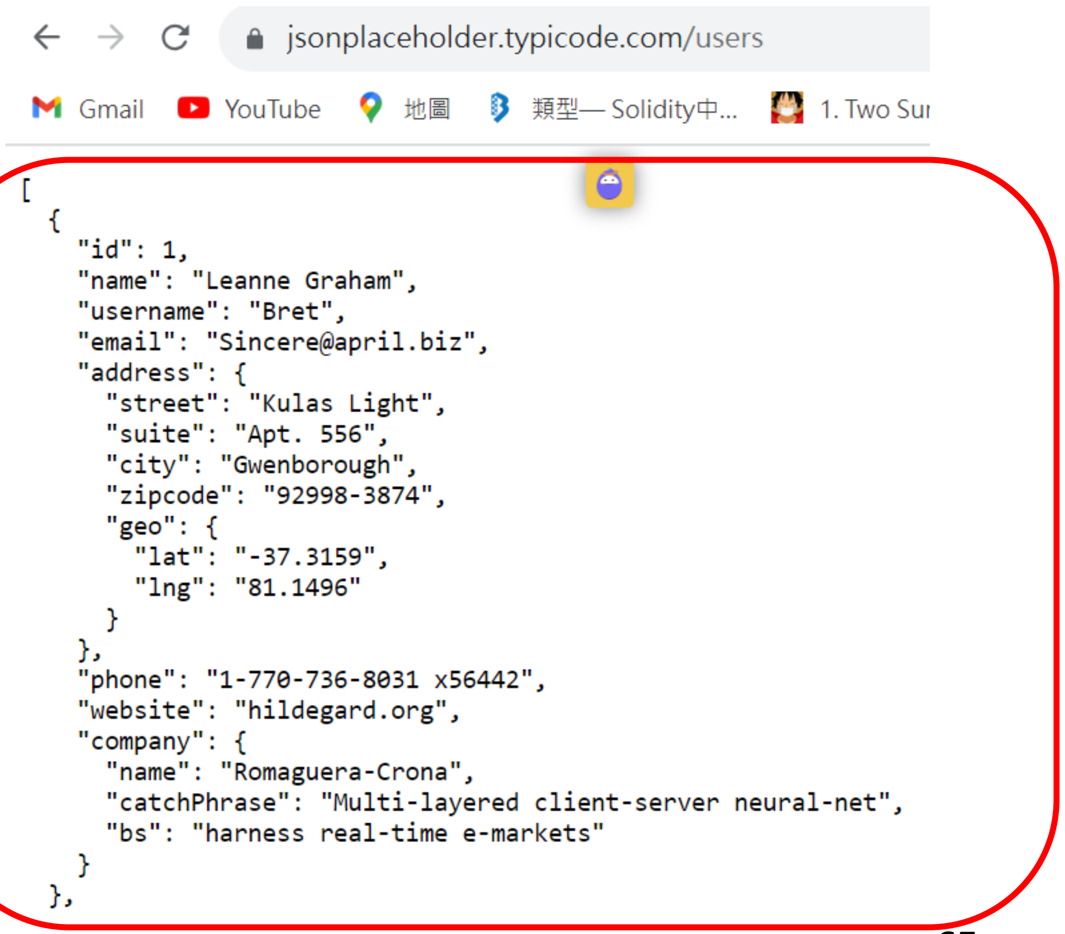
JSONPlaceholder Guide Sponsor this p

{JSON} Placeholder

Free fake API for testing and prototyping.

Powered by JSON Server + LowDB. Tested with XV.

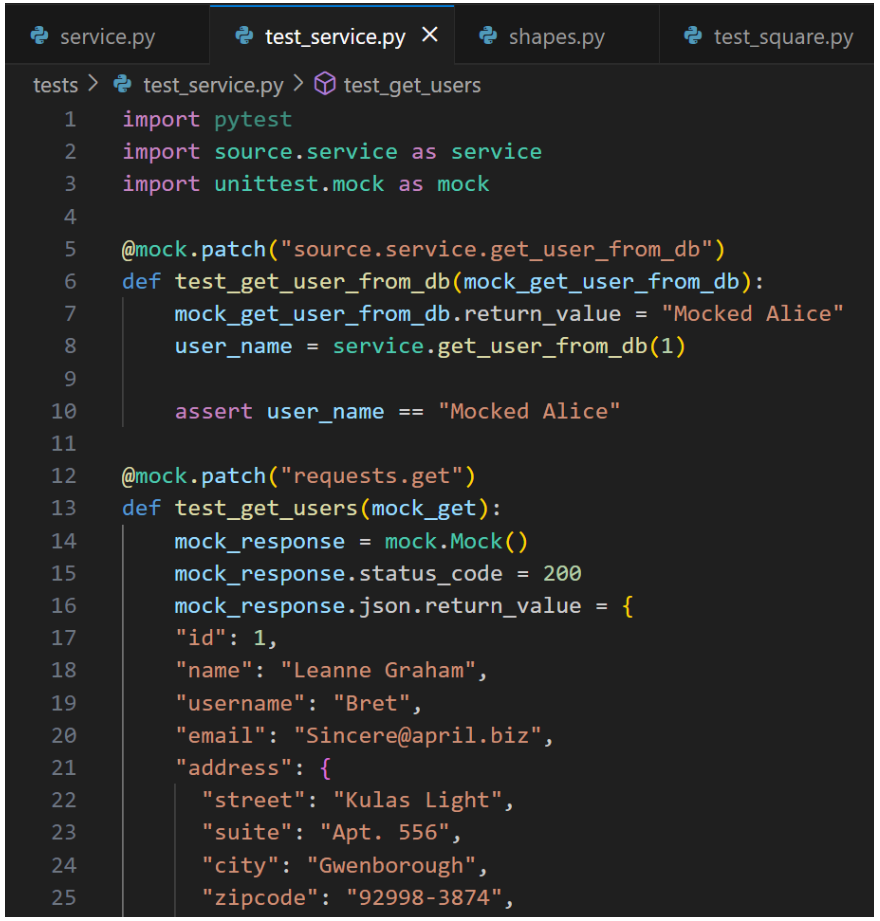
Serving ~2 billion requests each month.


jsonplaceholder.typicode.com/users

Gmail YouTube 地圖 類型— Solidity中... 1. Two Sur

```
[
  {
    "id": 1,
    "name": "Leanne Graham",
    "username": "Bret",
    "email": "Sincere@april.biz",
    "address": {
      "street": "Kulas Light",
      "suite": "Apt. 556",
      "city": "Gwenborough",
      "zipcode": "92998-3874",
      "geo": {
        "lat": "-37.3159",
        "lng": "81.1496"
      }
    },
    "phone": "1-770-736-8031 x56442",
    "website": "hildegard.org",
    "company": {
      "name": "Romaguera-Crona",
      "catchPhrase": "Multi-layered client-server neural-net",
      "bs": "harness real-time e-markets"
    }
  },
```

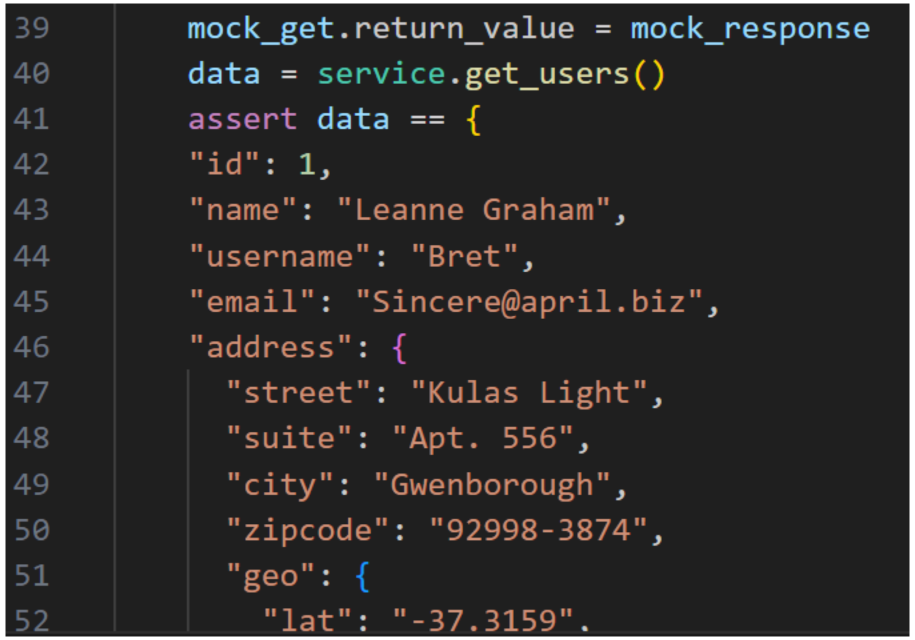
撰寫 模擬API回傳資料的測試



service.py | test_service.py | shapes.py | test_square.py

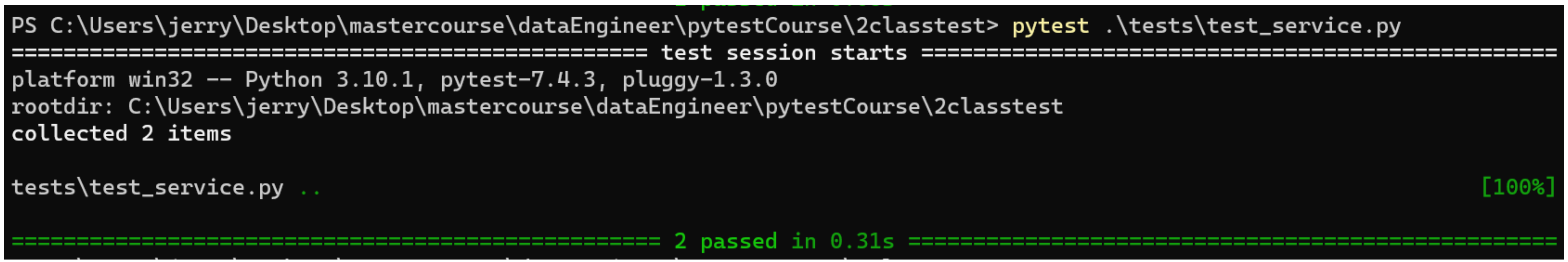
tests > test_service.py > test_get_users

```python
import pytest
import source.service as service
import unittest.mock as mock

@mock.patch("source.service.get_user_from_db")
def test_get_user_from_db(mock_get_user_from_db):
    mock_get_user_from_db.return_value = "Mocked Alice"
    user_name = service.get_user_from_db(1)

    assert user_name == "Mocked Alice"

@mock.patch("requests.get")
def test_get_users(mock_get):
    mock_response = mock.Mock()
    mock_response.status_code = 200
    mock_response.json.return_value = {
    "id": 1,
    "name": "Leanne Graham",
    "username": "Bret",
    "email": "Sincere@april.biz",
    "address": {
      "street": "Kulas Light",
      "suite": "Apt. 556",
      "city": "Gwenborough",
      "zipcode": "92998-3874",
```



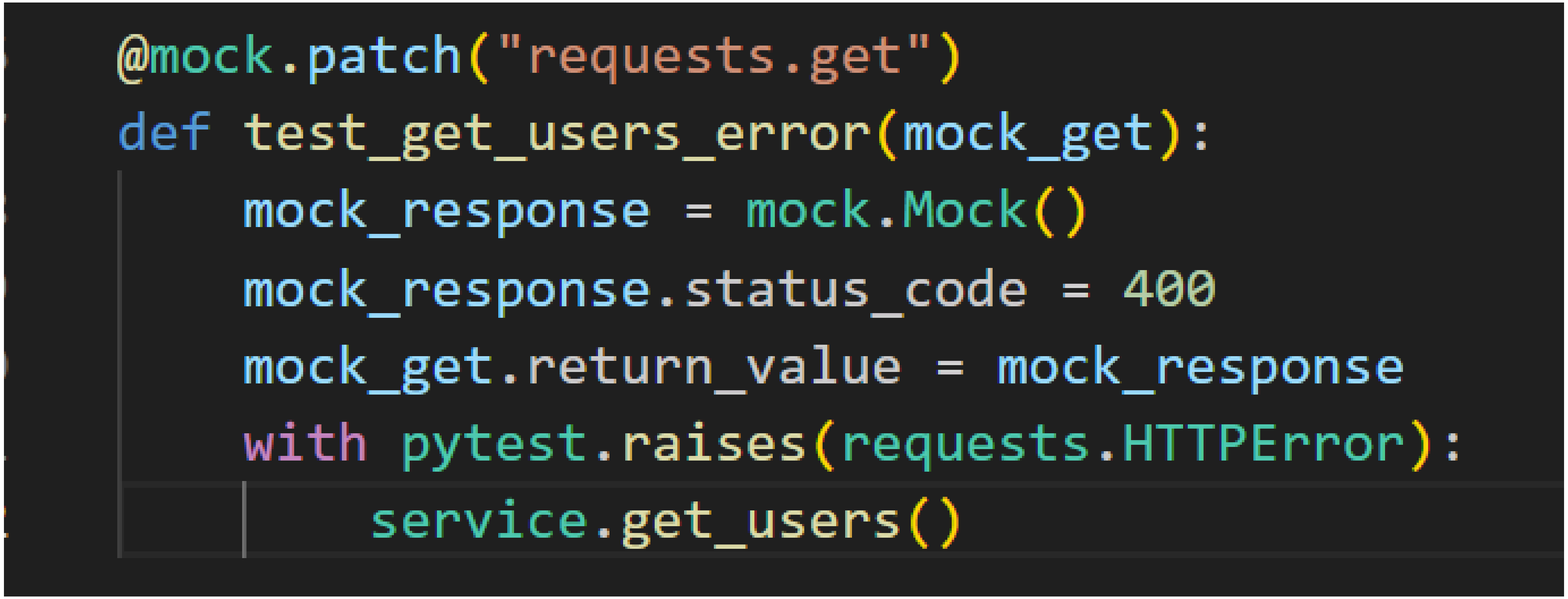
```python
    mock_get.return_value = mock_response
    data = service.get_users()
    assert data == {
    "id": 1,
    "name": "Leanne Graham",
    "username": "Bret",
    "email": "Sincere@april.biz",
    "address": {
      "street": "Kulas Light",
      "suite": "Apt. 556",
      "city": "Gwenborough",
      "zipcode": "92998-3874",
      "geo": {
        "lat": "-37.3159",
```

成功



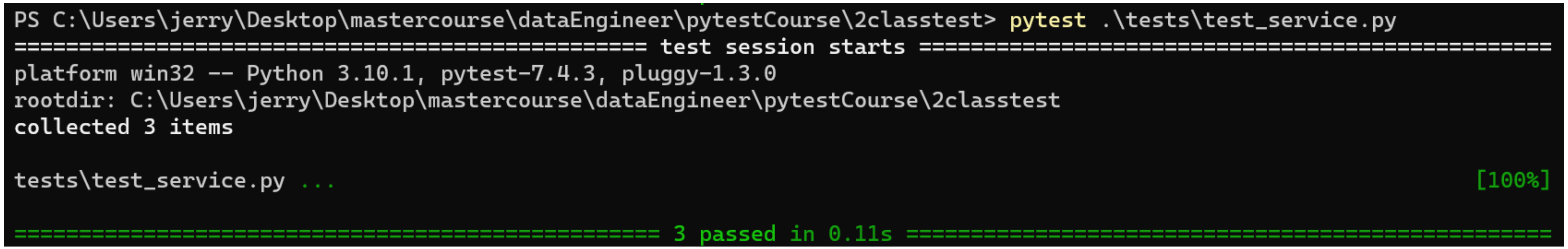
```
PS C:\Users\jerry\Desktop\mastercourse\dataEngineer\pytestCourse\2classtest> pytest .\tests\test_service.py
================================================= test session starts =================================================
platform win32 -- Python 3.10.1, pytest-7.4.3, pluggy-1.3.0
rootdir: C:\Users\jerry\Desktop\mastercourse\dataEngineer\pytestCourse\2classtest
collected 2 items

tests\test_service.py ..                                                                                        [100%]

================================================== 2 passed in 0.31s ==================================================
```

測試 Error



```python
@mock.patch("requests.get")
def test_get_users_error(mock_get):
    mock_response = mock.Mock()
    mock_response.status_code = 400
    mock_get.return_value = mock_response
    with pytest.raises(requests.HTTPError):
        service.get_users()
```

成功



```
PS C:\Users\jerry\Desktop\mastercourse\dataEngineer\pytestCourse\2classtest> pytest .\tests\test_service.py
================================================= test session starts =================================================
platform win32 -- Python 3.10.1, pytest-7.4.3, pluggy-1.3.0
rootdir: C:\Users\jerry\Desktop\mastercourse\dataEngineer\pytestCourse\2classtest
collected 3 items

tests\test_service.py ...                                                                                        [100%]

================================================== 3 passed in 0.11s ==================================================
```

End