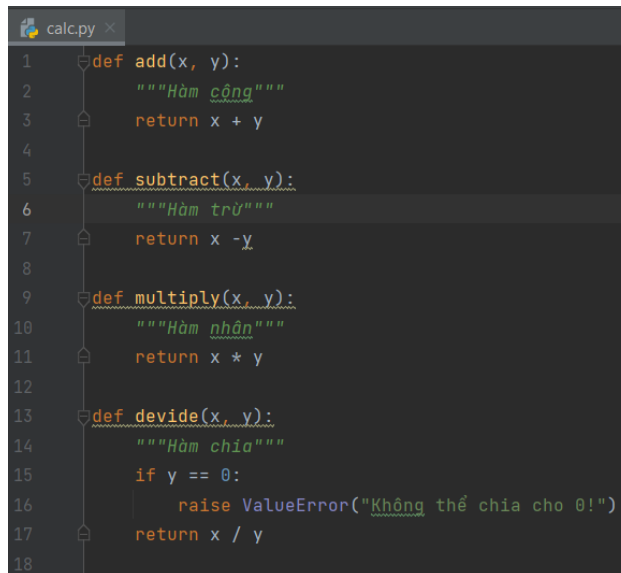


KỸ THUẬT TESTING

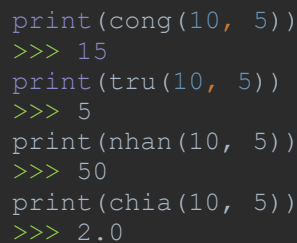
- Rất nhiều người thắc mắc làm sao để chương trình mình viết ra chạy đúng, không bị lỗi hoặc ít lỗi hơn?
- Thì hôm nay mình sẽ đến với kỹ thuật unit testing trong Python.
- Trong bài này chúng ta sẽ đi kiểm tra chương trình tính cộng, trừ, nhân, chia có đúng hay không?
- Trước tiên, ta tạo file cần test có tên là calc.py. Tiếp theo, chúng ta sẽ thiết lập từng hàm (Hình 1)



```
1 def add(x, y):  
2     """Hàm cộng"""  
3     return x + y  
4  
5 def subtract(x, y):  
6     """Hàm trừ"""  
7     return x - y  
8  
9 def multiply(x, y):  
10    """Hàm nhân"""  
11    return x * y  
12  
13 def divide(x, y):  
14    """Hàm chia"""  
15    if y == 0:  
16        raise ValueError("Không thể chia cho 0!")  
17    return x / y  
18
```

(Hình 1)

- Giống như hiện tại có rất nhiều người trong chúng ta muốn kiểm tra 1 đoạn chương trình thì họ sẽ in ra màn hình kết quả đó, ví dụ như (Hình 2).

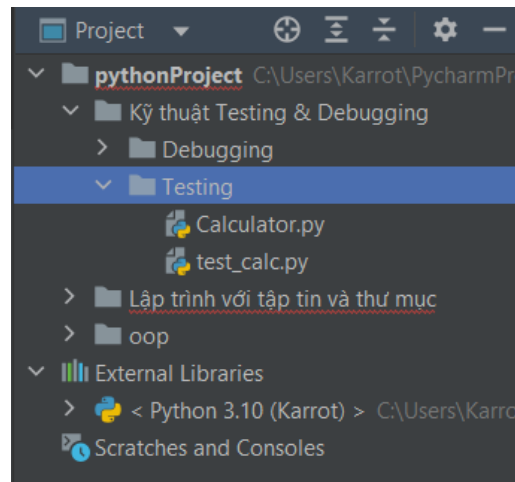


```
print(cong(10, 5))  
>>> 15  
print(tru(10, 5))  
>>> 5  
print(nhan(10, 5))  
>>> 50  
print(chia(10, 5))  
>>> 2.0
```

(Hình 2)

- Chúng ta nhìn bề ngoài thì có vẻ đúng nhưng test theo cách này thì sẽ không được hiệu quả cho lắm. Vậy bây giờ ta muốn test nhiều hàm bằng cách in chúng ra màn hình thì sẽ rất tốn thời gian, công sức,... để test hiệu quả hơn ta sẽ sử dụng kỹ thuật testing.

- Trước khi bắt đầu test, ta sẽ tạo thêm file mới để test (test_calc.py)



(Hình 3)

- Tiếp theo, ta thêm các module cần thiết:

1. import unittest

2. import calc (file cần test – calc ở đây là file cần test)

- Sau đó, tạo một class, ở đây ta sẽ đặt tên class này là TestCalc và chúng ta muốn kế thừa chúng thì thêm unittest.TestCase.

- Ở phía trong class là các hàm, ta sẽ đặt tên hàm test_add, test_subtract,... Để so sánh xem 2 giá trị có bằng nhau không thì dùng assertEquals (Tham khảo bảng dưới).

Method	Checks that	New in
assertEqual(a, b)	a == b	
assertNotEqual(a, b)	a != b	
assertTrue(x)	bool(x) is True	
assertFalse(x)	bool(x) is False	
assertIs(a, b)	a is b	3.1
assertIsNot(a, b)	a is not b	3.1
assertIsNone(x)	x is None	3.1
assertIsNotNone(x)	x is not None	3.1
assertIn(a, b)	a in b	3.1
assertNotIn(a, b)	a not in b	3.1
assertIsInstance(a, b)	isinstance(a, b)	3.2
assertNotIsInstance(a, b)	not isinstance(a, b)	3.2

(Hình 4)

- Được code như hình:

```
import unittest
import calc

class TestCalc(unittest.TestCase):

    def test_add(self):
        result = calc.add(10, 5)
        self.assertEqual(result, 15)

    def test_subtract(self):
        result = calc.subtract(10, 5)
        self.assertEqual(result, 5)

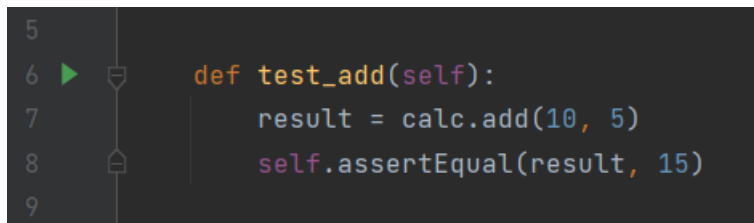
    def test_multiply(self):
        result = calc.multiply(10, 5)
        self.assertEqual(result, 50)

    def test_devide(self):
        result = calc.devide(10, 5)
        self.assertEqual(result, 2)

if __name__ == '__main__':
    unittest.main()
```

(Hình 5)

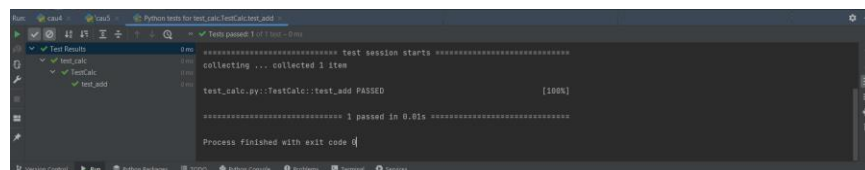
- Kiểm tra phép cộng (10 + 5), nhấn nút mũi tên màu xanh bên trái để chạy code của hàm test_add(). Ở đây, ta kiểm tra xem 10 + 5 có bằng 15 hay không?



```
5
6 ▶ def test_add(self):
7   result = calc.add(10, 5)
8   self.assertEqual(result, 15)
9
```

(Hình 6)

- Kết quả báo đúng.



(Hình 7)

- Xét đến trường hợp sai, ta sẽ đổi $15 \rightarrow 25$ thì kết quả hiển thị sai ($25 \neq 15$) nói cách dễ hiểu thì về bên trái là kết quả của tổng 10 và 5 (kết quả sai), còn về phải là kết quả kỳ vọng (kết quả đúng).

```
test_calc.py::TestCalc::test_add FAILED
test_calc.py:5 (TestCalc.test_add)
25 != 15

Expected :15
Actual   :25
<Click to see difference>
```

(Hình 8)

- Giờ chúng ta sẽ kiểm tra nhiều hàm cùng một lúc (đã thay đổi kết quả so sánh – Hình 9).
Tiến hành nhấn nút Run để test.

```
def test_add(self):
    self.assertEqual(calc.add(10, 5), 25)
    self.assertEqual(calc.add(10, 5), 15)
    self.assertEqual(calc.add(-1, 5), 4)
    self.assertEqual(calc.add(-10, 5), -5)

def test_subtract(self):
    self.assertEqual(calc.subtract(10, 5), 105)
    self.assertEqual(calc.subtract(10, 5), 5)
    self.assertEqual(calc.subtract(10, 0), 10)
    self.assertEqual(calc.subtract(-8, -2), 4)

def test_multiply(self):
    self.assertEqual(calc.multiply(1, 1), 1)
    self.assertEqual(calc.multiply(10, 9), 90)
    self.assertEqual(calc.multiply(7, 2), 16)
    self.assertEqual(calc.multiply(2, 5), 10)

def test_devide(self):
    self.assertEqual(calc.devide(8, 4), 2)
    self.assertEqual(calc.devide(1, 0), 0)
    self.assertEqual(calc.devide(100, 20), 5)
    self.assertEqual(calc.devide(24, 8), 5)
```

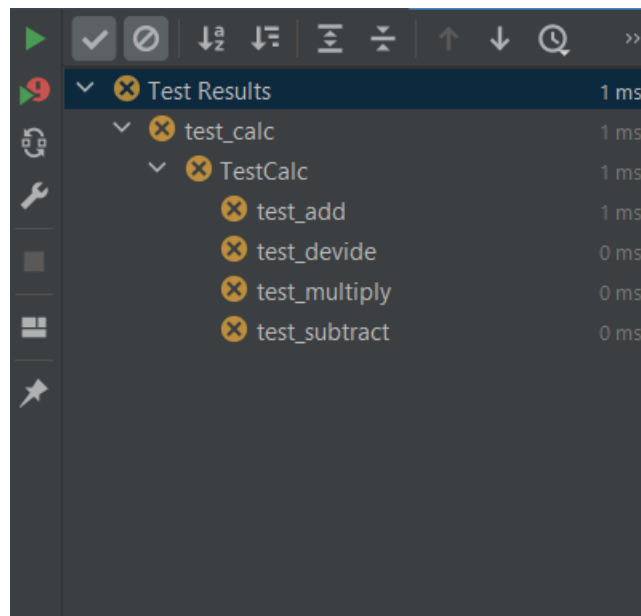
(Hình 9)

- Sau khi chạy xong, nó sẽ báo kết quả ra màn hình rằng có 4 lỗi.

```
===== 4 failed in 0.07s =====  
  
Process finished with exit code 1
```

(Hình 10)

- Ta đi kiểm tra lỗi đầu tiên, nhấn vào test_add (Hình 11) để xem kết quả.



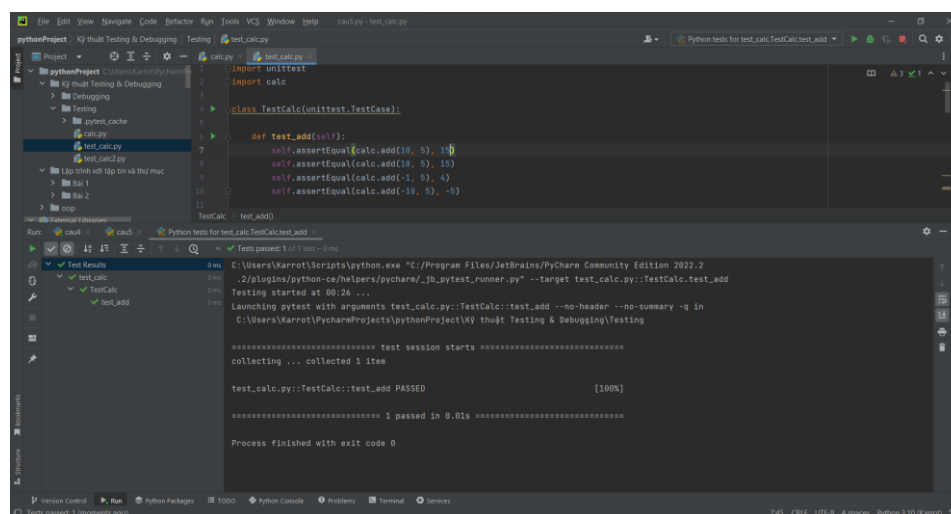
(Hình 11)

- Ở phần test_add, kết quả báo sai (Hình 12) dòng thứ 7, tổng của 10 và 5 không thể bằng 25 được → sửa 25 thành 15 thì kết quả đúng (Hình 13).

```
def test_add(self):  
    self.assertEqual(calc.add(10, 5), 25)  
    self.assertEqual(calc.add(10, 5), 15)  
    self.assertEqual(calc.add(-1, 5), 4)  
    self.assertEqual(calc.add(-10, 5), -5)
```

```
FAILED [ 25%]  
test_calc.py:5 (TestCalc.test_add)  
25 != 15  
  
Expected :15  
Actual   :25  
<Click to see difference>  
  
self = <test_calc.TestCalc testMethod=test_add>  
  
def test_add(self):  
> self.assertEqual(calc.add(10, 5), 25)  
test_calc.py:7: AssertionError
```

(Hình 12)



(Hình 13)

- Tiếp đến là test_subtract, xuất hiện lỗi ở dòng 13, sau khi sửa 105 → 5, ta test lại phần này thì thấy vẫn còn lỗi ở dòng 16 → tiến hành sửa → test lại thì in ra màn hình kết quả đúng (Hình 15).

```
test_calc.py::TestCalc::test_subtract FAILED
test_calc.py:11 (TestCalc.test_subtract)
4 != -6

Expected :-6
Actual   :4
<Click to see difference>

self = <test_calc.TestCalc testMethod=test_subtract>

def test_subtract(self):
    self.assertEqual(calc.subtract(10, 5), 5)
    self.assertEqual(calc.subtract(10, 5), 5)
    self.assertEqual(calc.subtract(10, 0), 10)
>    self.assertEqual(calc.subtract(-8, -2), 4)
```

(Hình 14)

```
C:\Users\Karrot\Scripts\python.exe "C:/Program Files/JetBrains/PyCharm Community Edition 2022.2
.2/plugins/python-ce/helpers/pycharm/_jb_pytest_runner.py" --target test_calc.py::TestCalc.test_subtract
Testing started at 00:36 ...
Launching pytest with arguments test_calc.py::TestCalc::test_subtract --no-header --no-summary -q in
C:\Users\Karrot\PycharmProjects\pythonProject\Kỹ thuật Testing & Debugging\Testing

===== test session starts =====
collecting ... collected 1 item

test_calc.py::TestCalc::test_subtract PASSED [100%]

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

Process finished with exit code 0
```

(Hình 15)

- Tiếp đến phần test_multiply, nó báo sai ở dòng 21 vì $7 * 2 = 14 \neq 16$ → sửa → test → kết quả đúng.

- Cuối cùng là phần test_divide, sau khi test thì báo sai ở dòng 26 do 1 không thể chia cho 0 được (Hình 16) . Để sửa lỗi này ta có 2 cách (Hình 17).

```

def test_devide(self):
    self.assertEqual(calc.devide(8, 4), 2)
> self.assertEqual(calc.devide(1, 0), 0)

test_calc.py:26:
-----
x = 1, y = 0

def dividex(x, y):
    """Hàm chia"""
    if y == 0:
>         raise ValueError("Không thể chia cho 0!")
E         ValueError: Không thể chia cho 0!

calc.py:16: ValueError

```

(Hình 16)

```

def test_devide(self):
    self.assertEqual(calc.devide(8, 4), 2)
    self.assertRaises(ValueError, calc.devide, 10, 0)    #Cach 1
    with self.assertRaises(ValueError):                  #Cach 2
        calc.devide(10, 0)
    self.assertEqual(calc.devide(100, 20), 5)
    self.assertEqual(calc.devide(24, 8), 5)

```

(Hình 17)

- Test lại thì nó báo chỉ còn 1 lỗi ở dòng 30, do $24 / 8 = 3$ không thể bằng 5 (Hình 18).

```

collecting ... collected 1 item

test_calc.py::TestCalc::test_devide FAILED
test_calc.py:23 (TestCalc.test_devide)
5 != 3.0

Expected :3.0
Actual   :5
<Click to see difference>

self = <test_calc.TestCalc testMethod=test_devide>

```

(Hình 18)

- Sau khi tiến hành sửa, thì ta chạy lại lần nữa để test toàn bộ chương trình.

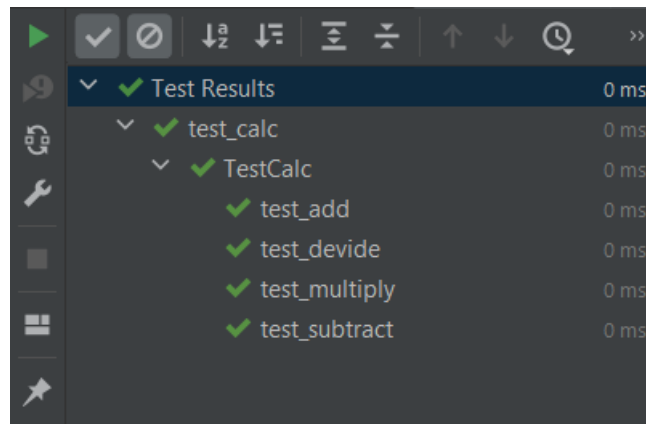

```
===== test session starts =====
collecting ... collected 4 items

test_calc.py::TestCalc::test_add PASSED [ 25%]
test_calc.py::TestCalc::test_devide PASSED [ 50%]
test_calc.py::TestCalc::test_multiply PASSED [ 75%]
test_calc.py::TestCalc::test_subtract PASSED [100%]

===== 4 passed in 0.02s =====

Process finished with exit code 0
```

(Hình 19)



(Hình 20)

- Hoàn tất kiểm tra.