LAB - Giới thiệu về Unit Test

1. Giới thiệu về Unit Test

Unit là thành phần nhỏ nhất mà chúng ta có thể kiểm tra được.VD: function, procedure, class hoặc method.

Mỗi Unit Test đều được thiết kế theo trình tự sau:

- Thiết lập các điều kiện cần thiết: khởi tạo các đối tượng, xác định tài nguyên cần thiết, xây dựng các dữ liệu giả…
- Gọi các phương thức cần kiểm tra.
- Kiểm tra sự hoạt động đúng đắn của các phương thức.
- Dọn dẹp tài nguyên sau khi kết thúc kiểm tra.

Để có thể hiểu đúng hơn về unit test, chúng ta nên biết về TDD (Test-Driven Development), là một quy trình phát triển phần mềm, bao gồm 5 bước:

- Write tests
- Run these tests – must fail
- Write source code
- Run these tests – must pass
- Refactor the code

Vào giai đoạn đầu tiên phát triển phần mềm, chúng ta phải nắm chắc được những chức năng cần phát triển, thông số đầu vào của chức năng và kết quả mong đợi của chức năng đó. Nếu chúng ta không viết test trước mà viết code luôn thì khi chạy code sẽ không chắc chắn được đầu vào và đầu ra đã bao phủ hết tất cả các trường hợp có thể xảy ra hay chưa. Khi viết một đoạn test, người phát triển phải hiểu rõ các yêu cầu và đặc điểm kỹ thuật của tính năng, làm cho người phát triển tập trung vào các yêu cầu trước khi viết code. Lợi ích của việc này là giúp ta thấy được những chi tiết còn mơ hồ và bỏ sót trước khi bắt đầu viết code, tránh được việc phải sửa hay thậm chí viết lại code.

Vậy nên Unit Test là một quá trình thiết kế chứ không phải là quá trình kiểm thử. Tiếp theo để nắm rõ hơn chúng ta sẽ bắt đầu vào phần ví dụ.

2. Kiểm tra các phép tính

- Đầu tiên chúng ta sẽ tạo file test_calc.py, sau đó import thư viện unittest và file calc.py chứa các hàm như add, subtract, multiply, devide.

```
import unittest

import calc
```

- Khai báo class TestCalc và định nghĩa các chức năng, assertEqual là để kiểm tra hai số có bằng nhau hay không:

```
class TestCalc(unittest.TestCase):

    def test_add(self):

        self.assertEqual(calc.add(5,5),10)

        self.assertEqual(calc.add(-1,1),0)

    def test_subtract(self):

        self.assertEqual(calc.subtract(6,3),3)

        self.assertEqual(calc.subtract(-1,1),-2)

    def test_multiply(self):

        self.assertEqual(calc.multiply(3,3),9)

        self.assertEqual(calc.multiply(-2,-3),6)


    def test_devide(self):
```

```
        self.assertEqual(calc.devide(6,2),3)

        self.assertEqual(calc.devide(-1,-1),1)
```

- Cuối cùng là khai báo hàm main để chạy chương trình:

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

- Tiếp theo chúng ta sẽ viết file calc.py:

```
def add(a,b):
    return a+b
def subtract(a,b):
    return a-b
def multiply(a,b):
    return a*b
def devide(a,b):
    return a/b
```

- Sau khi chạy chúng ta sẽ được kết quả như sau:

```
....
----------------------------------------------------------------------
Ran 4 tests in 0.010s

OK
>>>
```

3. Lấy tên nhà sản xuất, loại thiết bị và trả về tên đầy đủ của thiết bị

- Đầu tiên chúng ta sẽ tạo file test_device_name.py, import thư viện unittest và file device_name.py chứa hàm device _name:

```
from device_name import parse_device_name
import unittest
```

- Khai báo class NamesTestCase:

```
class NamesTestCase(unittest.TestCase):


   def test_parse_device_name(self):
      result = parse_device_name("Cisco","Router","2911")
      self.assertEqual(result,"Cisco Router 2911")
if __name__=="__main__":
   unittest.main()
```

- Sau đó chúng ta sẽ tạo file device_name.py:

```
def parse_device_name(vendor, device_type, platform):
   full_name = vendor + ' '+device_type+' '+platform
   return full_name.title()
```

- Chạy file sẽ cho ra kết quả:

```
.
----------------------------------
Ran 1 test in 0.017s

OK
>>>
```

- Nhưng nếu chúng ta gặp trường hợp tên thiết bị như “Cisco Router 2911 SEC” thì khi đó chạy test sẽ fail
- Chúng ta sẽ vào file test_device_name.py và sửa lại như sau:

```
def test_parse_device_name_(self):
    result = parse_device_name("Cisco","Router","2911","Sec")
    self.assertEqual(result,"Cisco Router 2911 Sec")
```

- Tiếp theo vào file device_name.py và sửa lại thêm trường hợp tham số đầu vào spec=’’. Nếu spec được khai báo thì sẽ thêm spec vào full_name.

```
def parse_device _name(vendor, device_type,platform, spec=''):
  if spec !='':
    full_name = vendor +' '+device_type+' '+platform+' '+spec
  else:
    full_name = vendor +' '+device_type+' '+platform
  return full_name.title()
```

- Cuối cùng chúng ta sẽ chạy lại file test_device_name.py và được kết quả cho thấy test pass.

```
.
----------------------------
Ran 1 test in 0.006s

OK
```