Unittest / PyUnit

Antonio Espín Herranz

unittest / PyUnit

- **unittest** y **PyUnit** son lo mismo.
- Este módulo para realizar pruebas es muy similar al framework de Java JUnit.

- Para cada **grupo de pruebas** tenemos que **crear una clase** que herede de **unittest.TestCase**, y añadir una serie de métodos que comiencen con test, que serán cada una de las pruebas que queremos ejecutar dentro de esa batería de pruebas

Ejecución de las pruebas

- Llamar a la función **main**() del módulo, con lo que se ejecutarán todos los métodos cuyo nombre comience con **test**, en orden alfanumérico.

- Al ejecutar cada una de las pruebas el **resultado** puede ser:
  - **OK**: La prueba ha pasado con éxito.
  - **FAIL**: La prueba no ha pasado con éxito. Se lanza una excepción AssertionError para indicarlo.
  - **ERROR**: Al ejecutar la prueba se lanzó una excepción distinta de AssertionError.

Ejemplo

```
import unittest
class EjemploPruebas(unittest.TestCase):
    def test(self):
        pass
if __name__ == “__main__”:
    unittest.main()
```

Dado que el método que modela nuestra prueba no lanza ninguna excepción, la prueba pasaría con éxito.

Ejemplo II

```
import unittest
class EjemploPruebas(unittest.TestCase):
    def test(self):
        raise AssertionError()

if __name__ == “__main__”:
    unittest.main()
```

Métodos de la clase: TestCase

- **assertAlmostEqual**(first, second, places=7, msg=None)
  - Comprueba que los objetos pasados como parámetros sean iguales hasta el séptimo decimal (o el número de decimales indicado por places).

- **assertEqual**(first, second, msg=None)
  - Comprueba que los objetos pasados como parámetros sean iguales.

- **assertFalse**(expr, msg=None)
  - Comprueba que la expresión sea falsa.

Métodos de la clase: TestCase

- **assertNotAlmostEqual**(first, second, places=7, msg=None)
  - Comprueba que los objetos pasados como parámetros no sean iguales hasta el séptimo decimal (o hasta el número de decimales indicado por places).

- **assertNotEqual**(first, second, msg=None)
  - Comprueba que los objetos pasados como parámetros no sean iguales.

- **assertRaises**(excClass, callableObj, *args, **kwargs)
  - Comprueba que al llamar al objeto callableObj con los parámetros definidos por *args y **kwargs se lanza una excepción de tipo excClass.

- **assertTrue**(expr, msg=None)
  - Comprueba que la expresión sea cierta.

Métodos de la clase: TestCase

- **assert_**(expr, msg=None)
  - Comprueba que la expresión sea cierta.
- **fail**(msg=None)
  - Falla inmediatamente.
- **failIf**(expr, msg=None)
  - Falla si la expresión es cierta.
- **failIfAlmostEqual**(first, second, places=7, msg=None)
  - Falla si los objetos pasados como parámetros son iguales hasta el séptimo decimal (o hasta el número de decimales indicado por places).

Métodos de la clase: TestCase

- **failIfEqual**(first, second, msg=None)
  - Falla si los objetos pasados como parámetros son iguales.

- **failUnless**(expr, msg=None)
  - Falla a menos que la expresión sea cierta.

- **failUnlessAlmostEqual**(first, second, places=7, msg=None)
  - Falla a menos que los objetos pasados como parámetros sean iguales hasta el séptimo decimal (o hasta el número de decimales indicado por places).

- **failUnlessEqual**(first, second, msg=None)
  - Falla a menos que los objetos pasados como parámetros sean iguales.

Métodos de la clase: TestCase

- **failUnlessRaises**(excClass, callableObj, *args, **kwargs)
  - Falla a menos que al llamar al objeto callableObj con los parámetros definidos por *args y **kwargs se lance una excepción de tipo excClass.

Ejemplo

```
import unittest
def cuadrado(num):
    """Calcula el cuadrado de un numero."""
    return num ** 2

class EjemploPruebas(unittest.TestCase):
    def test(self):
        l = [0, 1, 2, 3]
        r = [cuadrado(n) for n in l]
        self.assertEqual(r, [0, 1, 4, 9])

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

Preparación del contexto

- La clase TestCase proporciona un par de métodos que podemos sobreescribir para construir y desconstruir el entorno y que se ejecutan antes y después de las pruebas definidas en esa clase.

- Estos métodos son **setUp**() y **tearDown**().
  - **setUp**
    - Se lanza **antes de CADA test**
  - **tearDown**
    - Se lanza **después de CADA test**.

Ejemplo

```
class EjemploFixture(unittest.TestCase):
    def setUp(self):
        Print( “Preparando contexto” )
        self.lista = [0, 1, 2, 3]

    def test(self):
        Print( “Ejecutando prueba” )
        r = [cuadrado(n) for n in self.lista]
        self.assertEqual(r, [0, 1, 4, 9])

    def tearDown(self):
        print (“Desconstruyendo contexto” )
        del self.lista
```