UNIVERSIDAD DEL VALLE

INGENIERÍA EN ELECTRONICA

METODOLOGÍAS DE DESARROLLO DE SOFTWARE

Tarea Pruebas Unitarias

Presenta:

JUAN SEBASTIAN JOAQUI PARRA <<2410015-3744>>

ANA SOFIA JIMENEZ ZUÑIGA <<2410017-3744>>

JUAN DAVID LOAIZA JARAMILLO <<2410019-3744>>

Escuela de Ingeniería Eléctrica y Electrónica

Colombia, Santiago de Cali Diciembre del 2024

Tarea

1. Realizar un programa que tenga las 4 operaciones básicas, suma, resta, multiplicación y división.
2. Realizar 2 pruebas por cada operación (que incluyan valores negativos)
3. Generar un informe de cobertura y dar explicarlo.

Inicialmente, se crearon 2 archivos .py dentro de Visual Studio Code para realizar los códigos:

Tarea.py:

Tarea.py | Test_tarea.py

Tarea.py > Calculadora > multiply

```python
class Calculadora:
    def add(self, a, b):
        return a + b

    def subtract(self, a, b):
        return a - b

    def multiply(self, a, b):
        return a * b

    def divide(self, a, b):
        if b == 0:
            raise ValueError("No se puede dividir entre cero.")
        return a / b
```

Test_Tarea.py:

Tarea.py | Test_tarea.py

Test_tarea.py > ...

```python
import pytest
from Tarea import Calculadora

@pytest.fixture
def calculadora():
    return Calculadora()

def test_add(calculadora):
    assert calculadora.add(2, 3) == 5
    assert calculadora.add(-1, -3) == -4

def test_subtract(calculadora):
    assert calculadora.subtract(7, 3) == 4
    assert calculadora.subtract(-5, -5) == -10

def test_multiply(calculadora):
    assert calculadora.multiply(4, 3) == 12
    assert calculadora.multiply(-4, -5) == 20

def test_divide(calculadora):
    assert calculadora.divide(6, 3) == 2
    assert calculadora.divide(-6, -2) == 3
```

Explicación del código Tarea.py:

El archivo **Tarea.py** contiene una clase llamada **Calculadora**, que implementa las cuatro operaciones matemáticas básicas: **suma**, **resta**, **multiplicación** y **división**. La clase tiene los siguientes métodos:

- **add(a, b)**: Devuelve la suma de los dos números "a" y "b".
- **subtract(a, b)**: Devuelve la resta de "a" y "b".
- **multiply(a, b)**: Devuelve el producto de "a" y "b".
- **divide(a, b)**: Devuelve el resultado de dividir "a" entre "b", con una validación que lanza una excepción ValueError si "b" es igual a cero (0), para evitar la división por cero.

La clase es simple y permite realizar operaciones matemáticas estándar con validaciones básicas.

Explicación del código Test_Tarea.py

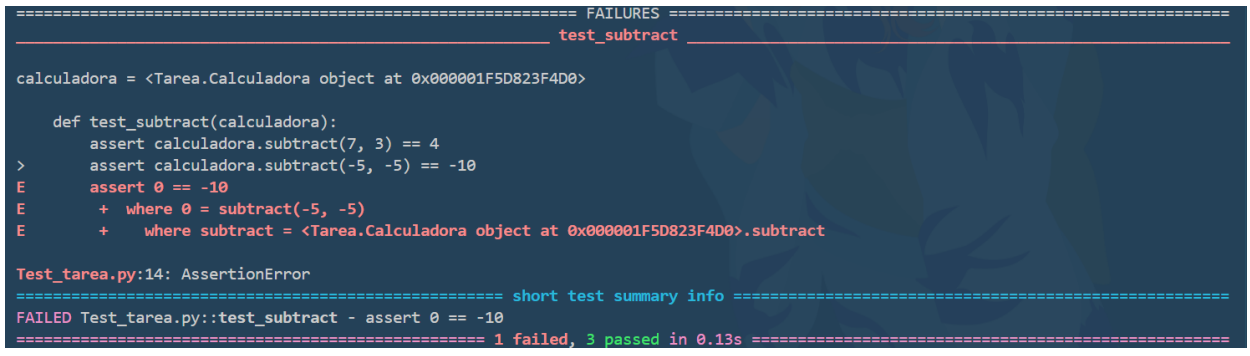
El archivo **Test_Tarea.py** contiene las pruebas unitarias para la clase **Calculadora**, utilizando el marco de pruebas **pytest**. Aquí se definen las siguientes pruebas:

- **test_add(calculadora)**: Verifica que la función **add** realice correctamente la **suma**, tanto con números positivos como negativos.
- **test_subtract(calculadora)**: Verifica que la función **subtract** realice correctamente la **resta**, tanto con números positivos como negativos.
- **test_multiply(calculadora)**: Verifica que la función **multiply** realice correctamente la **multiplicación**, con pruebas de números positivos y negativos.
- **test_divide(calculadora)**: Verifica que la función divide realice correctamente la **división**. Además, prueba el caso de **división por cero**, asegurando que se lance la excepción **ValueError** con el mensaje adecuado.

Estas pruebas permiten verificar que las operaciones matemáticas se realicen correctamente y que se manejen adecuadamente los errores **(como la división entre cero)** usando **pytest**.

Pruebas del código:

Por como se escribió la condición de la resta, debe marcar un error al usar el comando **"coverage run -m pytest"** debido a que apropósito se escribió un valor mal para verificar que esto ocurra:



Una vez corregido el código, observamos que todo se ejecuta como es debido:

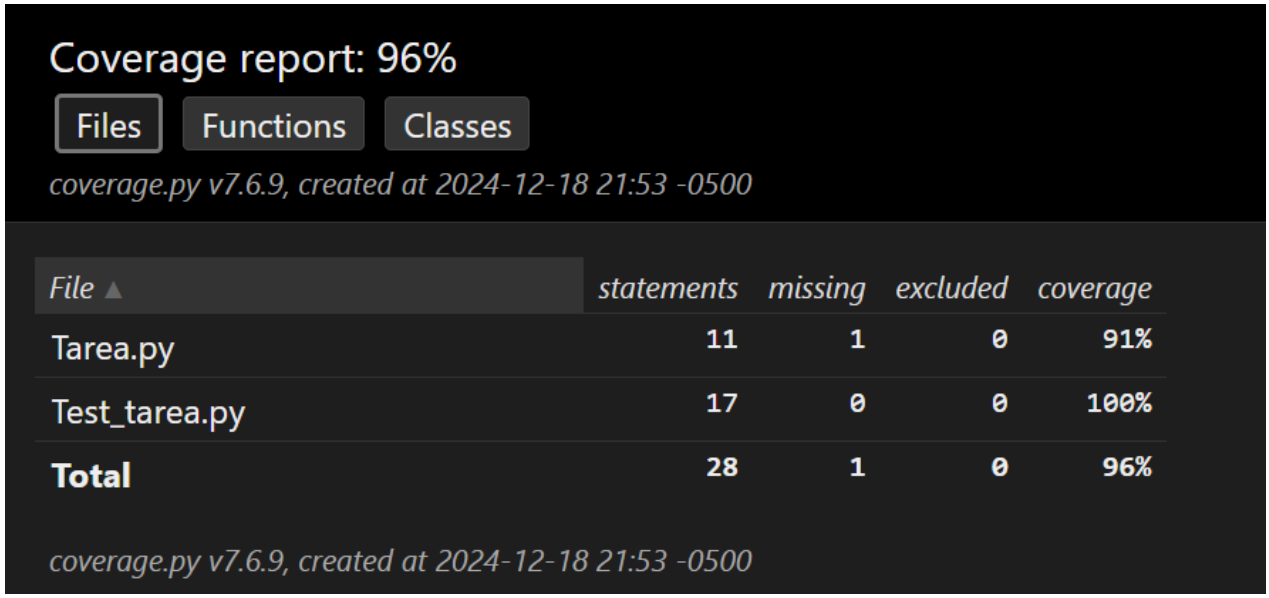
```
PS C:\Users\Juan Sebastian\Desktop\MDSO\Tarea Python> coverage run -m pytest
============================================================ test session starts ============================================================
platform win32 -- Python 3.11.3, pytest-8.3.4, pluggy-1.5.0
rootdir: C:\Users\Juan Sebastian\Desktop\MDSO\Tarea Python
collected 4 items

Test_tarea.py ....                                                                                                                  [100%]

============================================================= 4 passed in 0.06s =============================================================
```

Una vez mostrado este funcionamiento, se realizarán los reportes de cobertura de los códigos y sus pruebas unitarias:

Para esto, se usan los comandos **“coverage html”** y **“start htmlcov/index.html”:**

```
PS C:\Users\Juan Sebastian\Desktop\MDSO\Tarea Python> coverage html
Wrote HTML report to htmlcov\index.html
PS C:\Users\Juan Sebastian\Desktop\MDSO\Tarea Python> start htmlcov/index.html
PS C:\Users\Juan Sebastian\Desktop\MDSO\Tarea Python>
```

Con esto, podemos revisar el reporte de cobertura mediante una página en HTML:



Coverage report: 96%

Files | Functions | Classes

coverage.py v7.6.9, created at 2024-12-18 21:53 -0500

File ▲	function	statements	missing	excluded	coverage
Tarea.py	Calculadora.add	1	0	0	100%
Tarea.py	Calculadora.subtract	1	0	0	100%
Tarea.py	Calculadora.multiply	1	0	0	100%
Tarea.py	Calculadora.divide	3	1	0	67%
Tarea.py	*(no function)*	5	0	0	100%
Test_tarea.py	calculadora	1	0	0	100%
Test_tarea.py	test_add	2	0	0	100%
Test_tarea.py	test_subtract	2	0	0	100%
Test_tarea.py	test_multiply	2	0	0	100%
Test_tarea.py	test_divide	2	0	0	100%
Test_tarea.py	*(no function)*	8	0	0	100%
Total		28	1	0	96%

coverage.py v7.6.9, created at 2024-12-18 21:53 -0500

Cobertura global: El proyecto tiene un 96% de cobertura, lo que significa que el 96% de las líneas de código fueron ejecutadas durante las pruebas.

Funciones principales:

add, subtract, y multiply tienen una cobertura del 100%. Esto indica que todas las líneas de estas funciones fueron probadas con éxito.

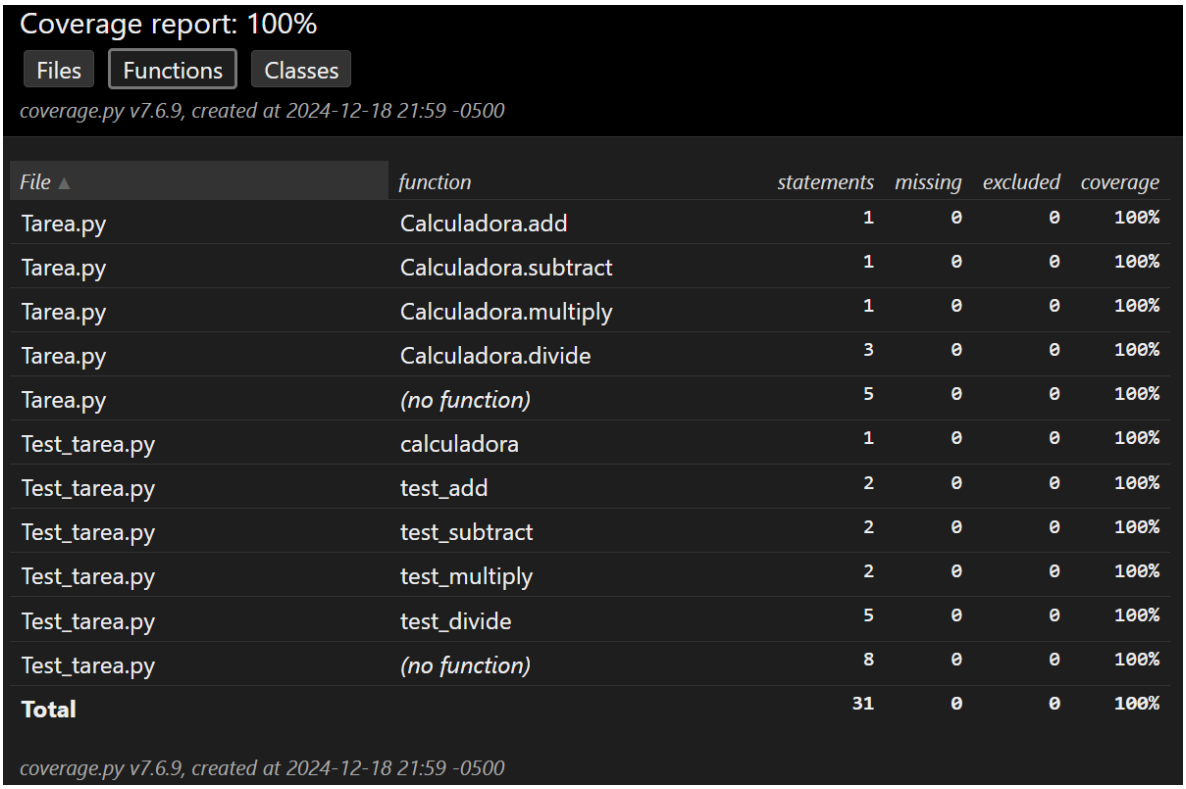
divide tiene una cobertura del 67%. Esto sugiere que una de las líneas (en este caso la que lanza la excepción por división entre cero) no fue cubierta durante las pruebas.

Todas las funciones de prueba (**test_add, test_subtract, test_multiply, test_divide**) tienen una cobertura del 100%, lo que confirma que las pruebas se ejecutaron correctamente para los casos cubiertos.

La cobertura general es alta, pero la línea de código que maneja la excepción por división entre cero no fue cubierta, para alcanzar el 100% de cobertura se le añadirá la siguiente línea de código:

```
# Prueba para división por cero con validación del mensaje
    with pytest.raises(ValueError) as exc_info:
        calculadora.divide(6, 0)
    assert str(exc_info.value) == "No se puede dividir entre cero."
```

Una vez agregada esta línea de código, podemos observar como la cobertura ahora sí alcanzó el **100% de las pruebas.**



Coverage report: 100%

Files | Functions | Classes

coverage.py v7.6.9, created at 2024-12-18 21:59 -0500

File ▲	*function*	*statements*	*missing*	*excluded*	*coverage*
Tarea.py	Calculadora.add	1	0	0	100%
Tarea.py	Calculadora.subtract	1	0	0	100%
Tarea.py	Calculadora.multiply	1	0	0	100%
Tarea.py	Calculadora.divide	3	0	0	100%
Tarea.py	*(no function)*	5	0	0	100%
Test_tarea.py	calculadora	1	0	0	100%
Test_tarea.py	test_add	2	0	0	100%
Test_tarea.py	test_subtract	2	0	0	100%
Test_tarea.py	test_multiply	2	0	0	100%
Test_tarea.py	test_divide	5	0	0	100%
Test_tarea.py	*(no function)*	8	0	0	100%
Total		31	0	0	100%

coverage.py v7.6.9, created at 2024-12-18 21:59 -0500