



Documentación de pruebas

Luz María Pérez Sarmiento

Tellez Zambrano Duncan
Gilberto

Alvarez Rugerio Marc Antony

Nolazco Hoyos Julio Cesar



Descripción General

El proyecto consiste en una **calculadora básica** programada en **Python**, acompañada de un conjunto de **pruebas unitarias** para verificar el correcto funcionamiento de cada una de sus operaciones matemáticas.

El sistema está compuesto por tres archivos principales:

1. `calculadora.py` → Contiene las funciones de la calculadora.
2. `test_calculadora.py` → Contiene las pruebas unitarias para cada función.
3. `prueba.py` → Ejecuta todas las pruebas con un formato visual detallado y organizado.

Archivo: `calculadora.py`

Este archivo contiene las funciones principales de la calculadora.

Cada función realiza una operación matemática básica.

```
# calculadora.py
```

```
def sumar(a, b):
```

```
    """Suma dos números."""
```

```
    return a + b
```

```
def restar(a, b):
```

```
    """Resta dos números."""
```

```
    return a - b
```

```
def multiplicar(a, b):
```

```
    """Multiplica dos números."""
```

```
    return a * b
```

```
def dividir(a, b):
```

```
"""Divide dos números. Lanza una excepción si b es 0."""
```

```
if b == 0:
```

```
    raise ValueError("No se puede dividir entre cero.")
```

```
return a / b
```

Explicación de cada función:

| Función | Descripción | Ejemplo | Resultado | Función |
|-------------------|---------------------------------|-------------------|-----------|-------------------|
| sumar(a, b) | Devuelve la suma de a y b. | sumar(3, 4) | 7 | sumar(a, b) |
| restar(a, b) | Devuelve la resta de a menos b. | restar(10, 5) | 5 | restar(a, b) |
| multiplicar(a, b) | Devuelve el producto de a y b. | multiplicar(2, 3) | 6 | multiplicar(a, b) |

Archivo: test_calculadora.py

Este archivo contiene las **pruebas unitarias** de cada método del archivo calculadora.py.

Se utiliza el módulo estándar de Python unittest para verificar que cada función produzca los resultados esperados.














Estructura del archivo:

- Se importa el módulo unittest.
- Se importa el archivo calculadora.
- Se crean varias clases: una por cada operación (TestSumar, TestRestar, etc.).
- Dentro de cada clase se definen distintos métodos que prueban casos específicos.


Ejemplo:

```
PS C:\Users\diego\calculadora> python prueba.py

=== Iniciando pruebas unitarias de calculadora ===

test_dividir_mixto (test_calculadora.TestDividir.test_dividir_mixto) ...  dividir_mixto -> OK
ok
test_dividir_negativos (test_calculadora.TestDividir.test_dividir_negativos) ...  dividir_negativos -> OK
ok
test_dividir_por_cero (test_calculadora.TestDividir.test_dividir_por_cero) ...  dividir_por_cero -> OK
ok
test_dividir_positivos (test_calculadora.TestDividir.test_dividir_positivos) ...  dividir_positivos -> OK
ok
test_multiplicar_mixto (test_calculadora.TestMultiplicar.test_multiplicar_mixto) ...  multiplicar_mixto -> OK
ok
test_multiplicar_negativos (test_calculadora.TestMultiplicar.test_multiplicar_negativos) ...  multiplicar_negativos -> OK
ok
test_multiplicar_positivos (test_calculadora.TestMultiplicar.test_multiplicar_positivos) ...  multiplicar_positivos -> OK
ok
test_restar_mixto (test_calculadora.TestRestar.test_restar_mixto) ...  restar_mixto -> OK
ok
test_restar_negativos (test_calculadora.TestRestar.test_restar_negativos) ...  restar_negativos -> OK
ok
test_restar_positivos (test_calculadora.TestRestar.test_restar_positivos) ...  restar_positivos -> OK
ok
test_sumar_mixto (test_calculadora.TestSumar.test_sumar_mixto) ...  sumar_mixto -> OK
ok
test_sumar_negativos (test_calculadora.TestSumar.test_sumar_negativos) ...  sumar_negativos -> OK
ok
test_sumar_positivos (test_calculadora.TestSumar.test_sumar_positivos) ...  sumar_positivos -> OK
```

Explicación:

- class TestSumar → define un grupo de pruebas para la función sumar().
- def test_sumar_positivos(self) → prueba un caso concreto.
- self.assertEqual(resultado, 7) → compara el valor calculado con el esperado.
- print("  ... OK") → imprime un mensaje de confirmación.

Clases incluidas:

| Clase | Función que prueba | Casos verificados |
|-----------------|--------------------|---|
| TestSumar | sumar(a, b) | Positivos, negativos, mixtos |
| TestRestar | restar(a, b) | Positivos, negativos, mixtos |
| TestMultiplicar | multiplicar(a, b) | Positivos, negativos, mixtos |
| TestDividir | dividir(a, b) | Positivos, negativos, mixtos, división por cero |

Métodos de prueba más destacados:

- assertEquals() → Verifica que el resultado sea igual al esperado.
- assertRaises(ValueError) → Verifica que una función lance una excepción.

Archivo: prueba.py

Este archivo ejecuta todas las pruebas de test_calculadora.py y muestra los resultados con un formato detallado y profesional.

prueba.py

```
import unittest
```

```
import test_calculadora # Importa el archivo de pruebas
```

```
if __name__ == "__main__":
```

```
    print("\n=== 🕒 Iniciando pruebas unitarias de calculadora ===\n")
```

```
    loader = unittest.TestLoader()
```

```
    suite = loader.loadTestsFromModule(test_calculadora)
```

```
    runner = unittest.TextTestRunner(verbosity=2)
```

```
    runner.run(suite)
```

```
    print("\n=== ✅ Todas las pruebas finalizaron correctamente ===\n")
```














Explicación:

| Elemento | Descripción |
|--------------------------------------|--|
| unittest.TestLoader() | Carga automáticamente todas las pruebas del archivo. |
| loadTestsFromModule() | Importa las pruebas desde el módulo test_calculadora. |
| unittest.TextTestRunner(verbosity=2) | Ejecuta las pruebas mostrando el nombre y el estado de cada una. |
| runner.run(suite) | Inicia la ejecución de todas las pruebas. |

Resultado Esperado

```
PS C:\Users\diego\calculadora> python prueba.py

=== Iniciando pruebas unitarias de calculadora ===

test_dividir_mixto (test_calculadora.TestDividir.test_dividir_mixto) ...  dividir_mixto -> OK
ok
test_dividir_negativos (test_calculadora.TestDividir.test_dividir_negativos) ...  dividir_negativos -> OK
ok
test_dividir_por_cero (test_calculadora.TestDividir.test_dividir_por_cero) ...  dividir_por_cero -> OK
ok
test_dividir_positivos (test_calculadora.TestDividir.test_dividir_positivos) ...  dividir_positivos -> OK
ok
test_multiplicar_mixto (test_calculadora.TestMultiplicar.test_multiplicar_mixto) ...  multiplicar_mixto -> OK
ok
test_multiplicar_negativos (test_calculadora.TestMultiplicar.test_multiplicar_negativos) ...  multiplicar_negativos -> OK
ok
test_multiplicar_positivos (test_calculadora.TestMultiplicar.test_multiplicar_positivos) ...  multiplicar_positivos -> OK
ok
test_restar_mixto (test_calculadora.TestRestar.test_restar_mixto) ...  restar_mixto -> OK
ok
test_restar_negativos (test_calculadora.TestRestar.test_restar_negativos) ...  restar_negativos -> OK
ok
test_restar_positivos (test_calculadora.TestRestar.test_restar_positivos) ...  restar_positivos -> OK
ok
test_sumar_mixto (test_calculadora.TestSumar.test_sumar_mixto) ...  sumar_mixto -> OK
ok
test_sumar_negativos (test_calculadora.TestSumar.test_sumar_negativos) ...  sumar_negativos -> OK
ok
test_sumar_positivos (test_calculadora.TestSumar.test_sumar_positivos) ...  sumar_positivos -> OK
```

Escribir un caso de prueba de integración

Caso de Prueba de Integración: Operaciones Encadenadas

Nombre del caso:

Prueba de integración de operaciones encadenadas en la calculadora.

Objetivo:

Verificar que las funciones de la calculadora (sumar, restar, multiplicar, dividir) funcionen correctamente al utilizarse de manera conjunta en una misma operación.

Descripción:

Se realizarán varias operaciones consecutivas utilizando los resultados intermedios como entrada para la siguiente operación, simulando una secuencia matemática completa.

Datos de entrada:

- $a = 5$
- $b = 3$
- $c = 2$
- $d = 4$

- $e = 2$

Operación a evaluar:

$$((a + b) - c) * d / e$$

Resultado esperado:

12

Caso de Prueba de Rendimiento: Repetición de Cálculos

Nombre del caso:

Prueba de rendimiento con operaciones repetidas.

Objetivo:

Evaluar el rendimiento de la calculadora al ejecutar un gran número de operaciones consecutivas y comprobar que mantiene la exactitud de los resultados sin errores ni degradación significativa del rendimiento.

Descripción:

Se repetirá una operación matemática simple $((a + b) * c / d)$ miles de veces para observar si la función mantiene un tiempo de respuesta aceptable y resultados correctos.

Datos de entrada:

- $a = 10$
- $b = 5$
- $c = 2$
- $d = 3$
- Número de repeticiones: 1000000 (un millón)

Resultado esperado:

Cada cálculo debe dar aproximadamente 10.0.

Resultado esperado en consola:

```
test_rendimiento_operaciones_repetidas  
(test_rendimiento.TestRendimientoCalculadora)
```

Prueba de rendimiento completada -> Tiempo: 2.841 segundos

ok