МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»

Кафедра інформаційних систем та мереж



Лабораторна робота №6

з дисципліни Спеціалізовані мови програмування

на тему

Розробка та Unit тестування Python додатку

Виконав:

студент групи РІ-21сп

Владислав ДМИТРЕНКО

Львів – 2024

**Мета:** Cтворення юніт-тестів для додатка-калькулятора на основі класів

**План роботи**

**Завдання 1:** Напишіть юніт-тест, щоб перевірити, що операція додавання в вашому додатку-калькуляторі працює правильно. Надайте тестові випадки як для позитивних, так і для негативних чисел.

**Завдання 2:** Створіть юніт-тести для переконання, що операція віднімання працює правильно. Тестуйте різні сценарії, включаючи випадки з від'ємними результатами.

**Завдання 3:** Напишіть юніт-тести, щоб перевірити правильність операції множення в вашому калькуляторі. Включіть випадки з нулем, позитивними та від'ємними числами.

**Завдання 4:** Розробіть юніт-тести для підтвердження точності операції ділення. Тести повинні охоплювати ситуації, пов'язані з діленням на нуль та різними числовими значеннями.

**Завдання 5:** Створіть юніт-тести, щоб перевірити, як ваш додаток-калькулятор обробляє помилки. Включіть тести для ділення на нуль та інших потенційних сценаріїв помилок. Переконайтеся, що додаток відображає відповідні повідомлення про помилки.

**Результати тестування:**

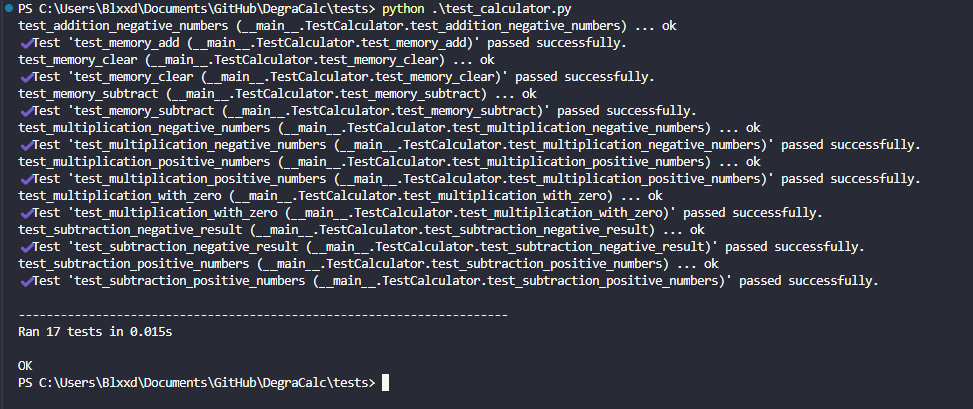
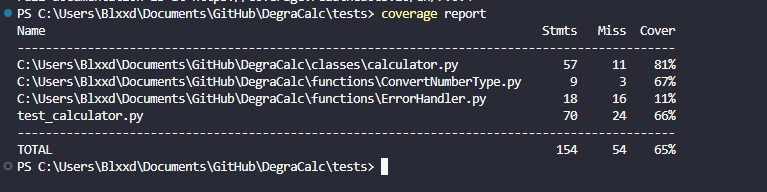
****

Рис. 1. Тестування проєкту

  
Рис. 2. Покриття тестами проєкту

Текст тест функцій

class TestCalculator(unittest.TestCase):

def setUp(self):

self.calculator = Calculator()

self.memory = Memory()

def test\_addition\_positive\_numbers(self):

self.assertEqual(self.calculator.Add(3, 5), 8)

def test\_addition\_negative\_numbers(self):

self.assertEqual(self.calculator.Add(-3, -5), -8)

def test\_addition\_positive\_and\_negative(self):

self.assertEqual(self.calculator.Add(-3, 5), 2)

def test\_subtraction\_positive\_numbers(self):

self.assertEqual(self.calculator.Subtract(10, 5), 5)

def test\_subtraction\_negative\_result(self):

self.assertEqual(self.calculator.Subtract(5, 10), -5)

def test\_multiplication\_with\_zero(self):

self.assertEqual(self.calculator.Multiply(5, 0), 0)

def test\_multiplication\_positive\_numbers(self):

self.assertEqual(self.calculator.Multiply(3, 5), 15)

def test\_multiplication\_negative\_numbers(self):

self.assertEqual(self.calculator.Multiply(-3, -5), 15)

def test\_division\_positive\_numbers(self):

self.assertEqual(self.calculator.Divide(10, 2), 5)

def test\_division\_with\_zero(self):

self.assertEqual(self.calculator.Divide(10, 0), "Error: x/0")

def test\_division\_negative\_result(self):

self.assertEqual(self.calculator.Divide(-10, 2), -5)

def test\_division\_by\_zero\_error\_handling(self):

self.assertEqual(self.calculator.Divide(5, 0), "Error: x/0")

def test\_invalid\_number\_conversion(self):

with self.assertRaises(ValueError):

ConvertNumberType("invalid")

def test\_get\_history(self):

self.calculator.Add(1, 1)

with open("History.txt", "r") as file:

self.assertIn("1 + 1 = 2", file.read())

def test\_memory\_add(self):

self.memory.Add(10)

self.assertEqual(self.memory.Read(), 10)

def test\_memory\_subtract(self):

self.memory.Add(10)

self.memory.Subtract(5)

self.assertEqual(self.memory.Read(), 5)

def test\_memory\_clear(self):

self.memory.Add(10)

self.memory.Clear()

self.assertEqual(self.memory.Read(), 0)

class CustomTextTestResult(unittest.TextTestResult):

def addSuccess(self, test):

super().addSuccess(test)

print(f"✔️ Test '{test}' passed successfully.")

def addFailure(self, test, err):

super().addFailure(test, err)

print(f"❌ Test '{test}' failed. Error: {err}")

def addError(self, test, err):

super().addError(test, err)

print(f"⚠️ Test '{test}' encountered an error. Error: {err}")

class CustomTextTestRunner(unittest.TextTestRunner):

def \_makeResult(self):

return CustomTextTestResult(self.stream, self.descriptions, self.verbosity)

if \_\_name\_\_ == '\_\_main\_\_':

runner = CustomTextTestRunner(verbosity=2)

unittest.main(testRunner=runner)

**Висновки:** В ході виконання лабораторної роботи було створено юніт-тестування на основі лабораторної роботи 2, який перевіряє правильність основних арифметичних операцій.