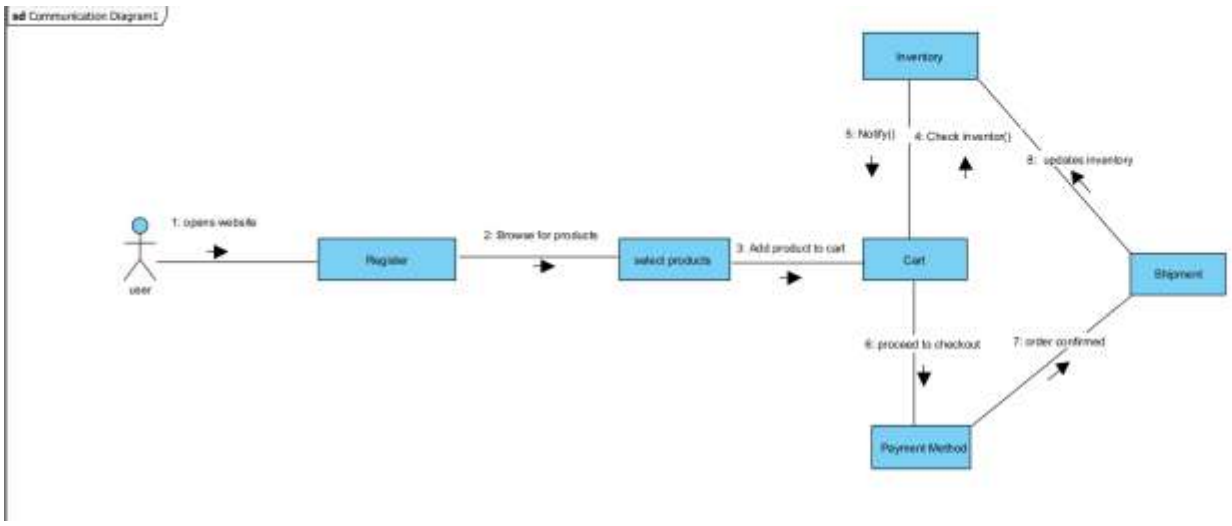
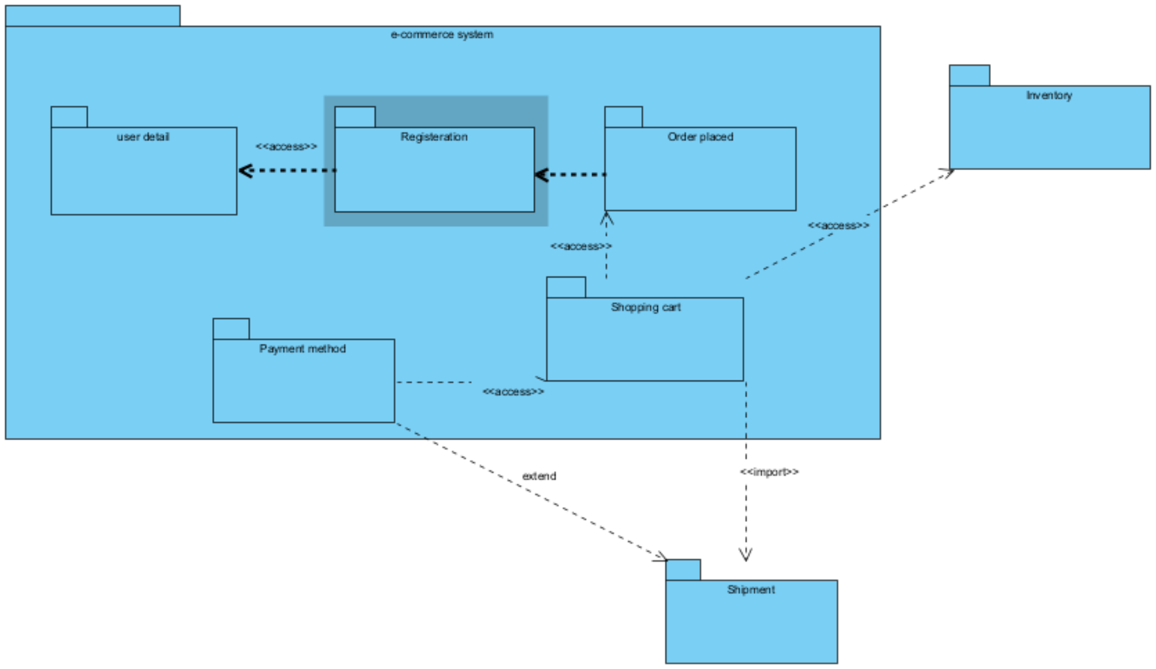
Lab Exam

Question 1:

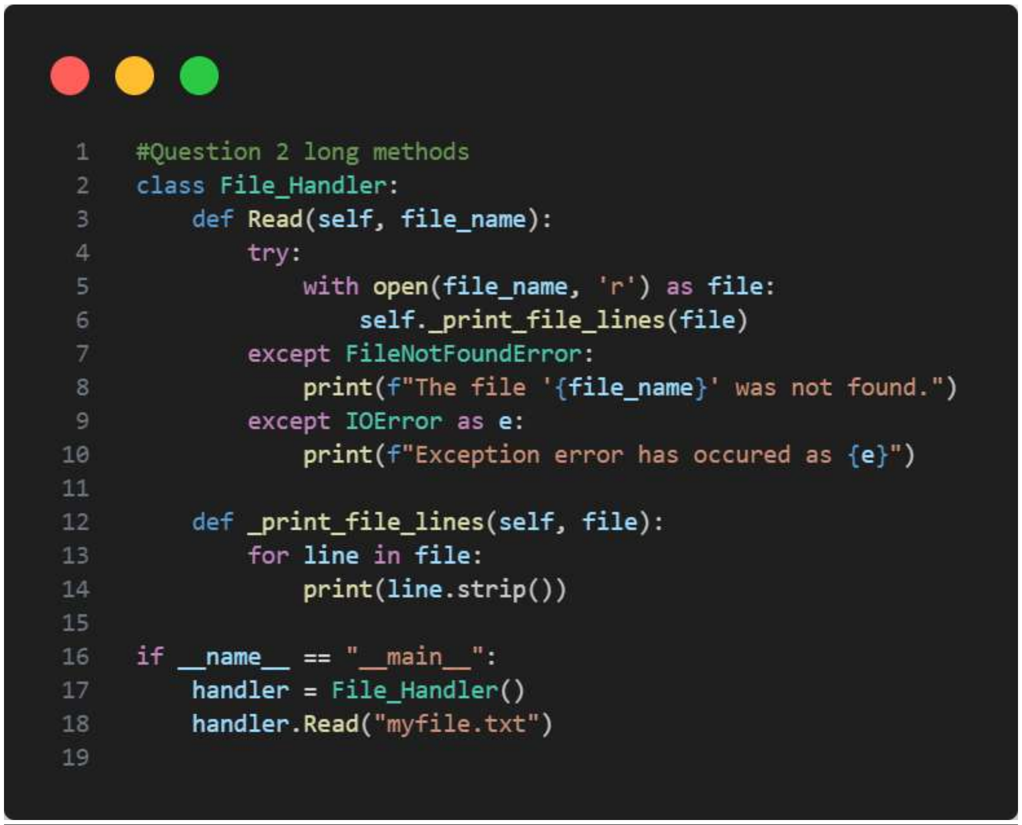
Communication diagram:



Package diagram:



Question:2



```python
#Question 2 long methods
class File_Handler:
    def Read(self, file_name):
        try:
            with open(file_name, 'r') as file:
                self._print_file_lines(file)
        except FileNotFoundError:
            print(f"The file '{file_name}' was not found.")
        except IOError as e:
            print(f"Exception error has occured as {e}")

    def _print_file_lines(self, file):
        for line in file:
            print(line.strip())

if __name__ == "__main__":
    handler = File_Handler()
    handler.Read("myfile.txt")

```

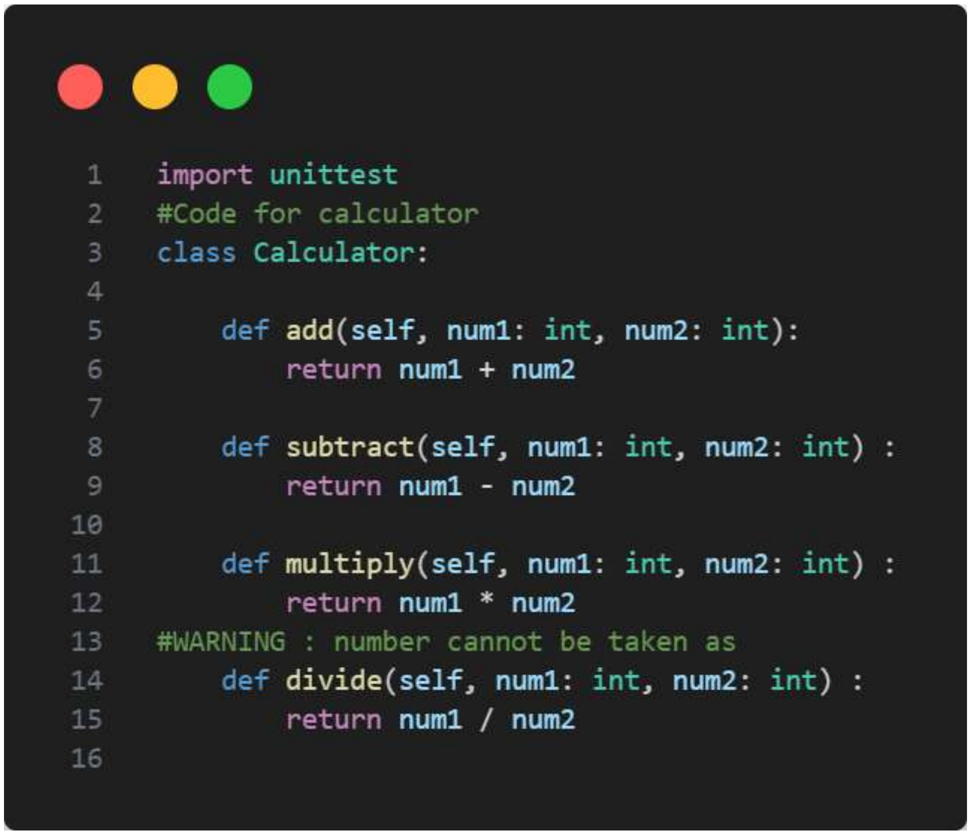
Question: 3

1. **Identify naming violations (class name, method names, parameters).**
2. **Identify missing or poor comments.**
3. **Explain why the division function is dangerous and how to handle it properly.**
4. **Write unit tests first for each operation (addition, subtraction, multiplication, division).**
5. **Handle edge cases such as division by zero.**

Answers:

1. The class name should be meaning full and should be in camelCase. Similarly, the method names convey no meaning and the datatype have error such as the datatype used for “d” is double but the parameters are in integers. There is no declaration of parameters and they are used a lot

2. Comment should have been added to explain the usage of code. Warning comment should be used to address that in 4th method division by zero can make the program to crash

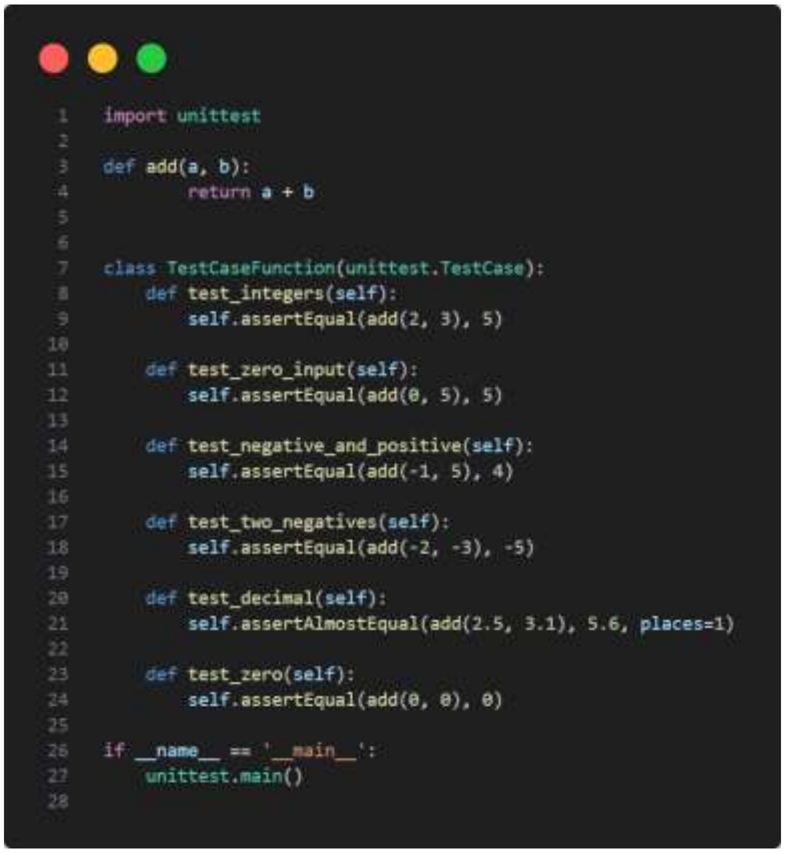


```python
import unittest
#Code for calculator
class Calculator:

    def add(self, num1: int, num2: int):
        return num1 + num2

    def subtract(self, num1: int, num2: int) :
        return num1 - num2

    def multiply(self, num1: int, num2: int) :
        return num1 * num2
#WARNING : number cannot be taken as
    def divide(self, num1: int, num2: int) :
        return num1 / num2

```

3. Division by zero can cause the program to crash. There should be exception handling for division by zero

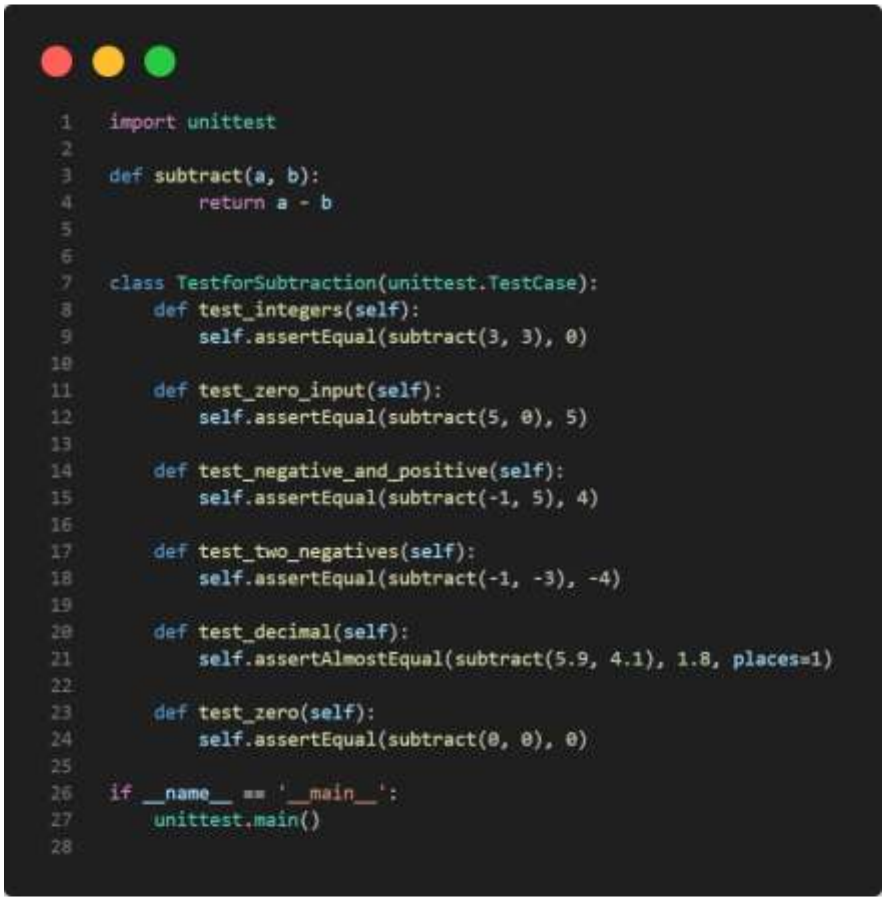
4. Test cases

Addition:



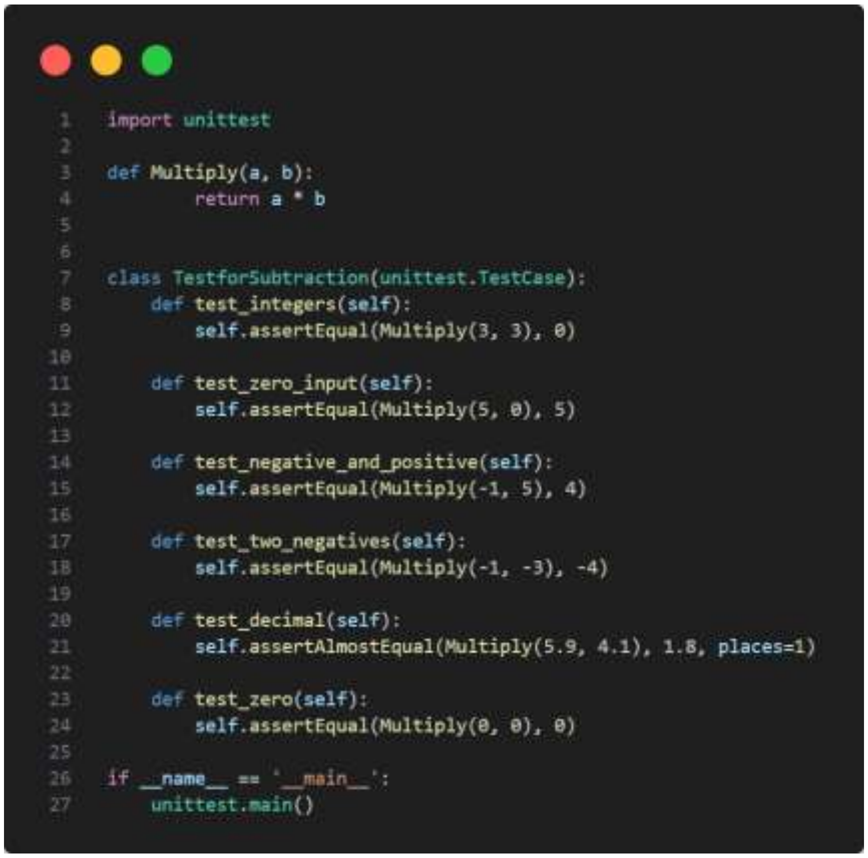
```python
import unittest

def add(a, b):
        return a + b


class TestCaseFunction(unittest.TestCase):
    def test_integers(self):
        self.assertEqual(add(2, 3), 5)

    def test_zero_input(self):
        self.assertEqual(add(0, 5), 5)

    def test_negative_and_positive(self):
        self.assertEqual(add(-1, 5), 4)

    def test_two_negatives(self):
        self.assertEqual(add(-2, -3), -5)

    def test_decimal(self):
        self.assertAlmostEqual(add(2.5, 3.1), 5.6, places=1)

    def test_zero(self):
        self.assertEqual(add(0, 0), 0)

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

```

Subtraction:



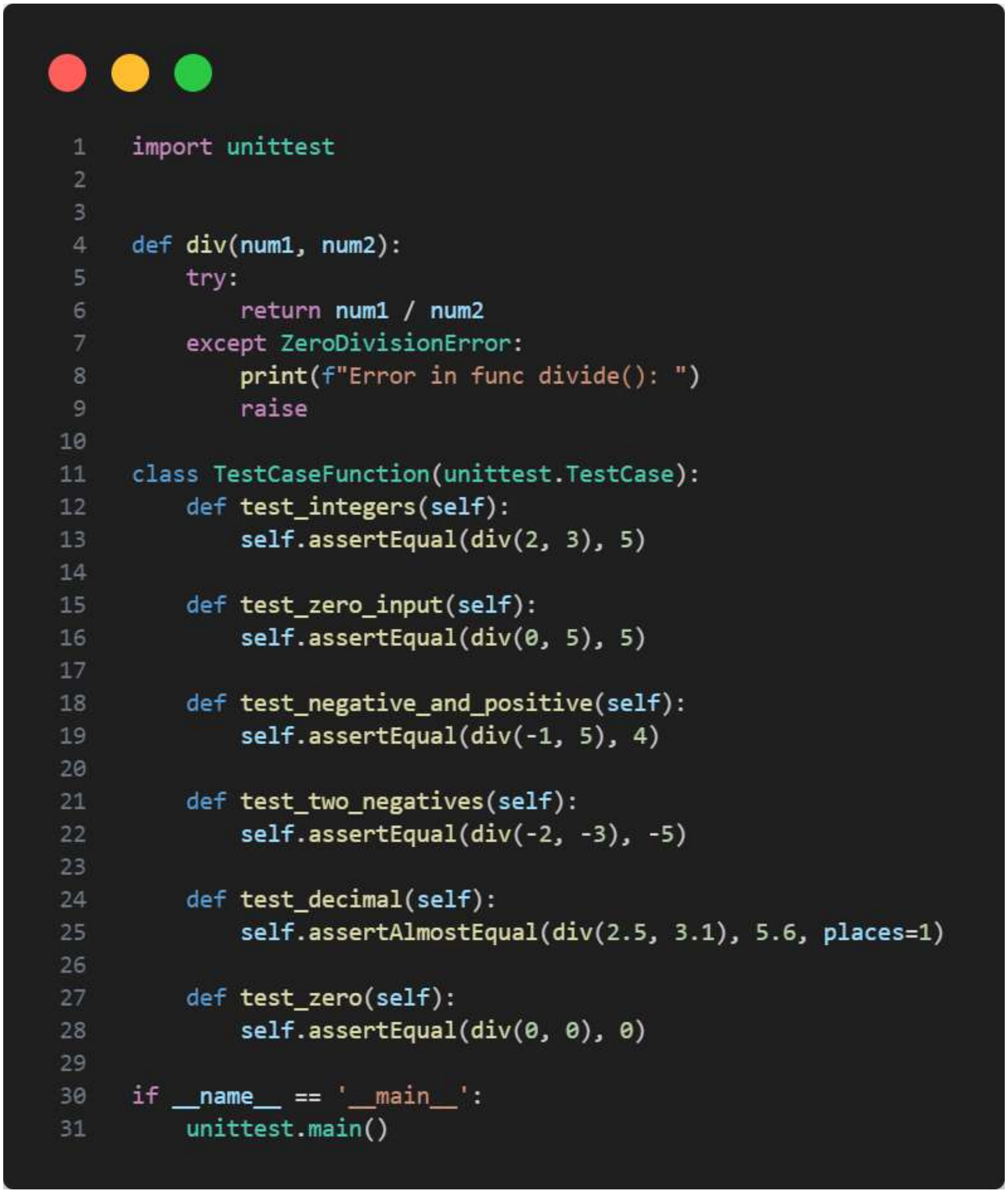
```python
import unittest

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


class TestforSubtraction(unittest.TestCase):
    def test_integers(self):
        self.assertEqual(subtract(3, 3), 0)

    def test_zero_input(self):
        self.assertEqual(subtract(5, 0), 5)

    def test_negative_and_positive(self):
        self.assertEqual(subtract(-1, 5), 4)

    def test_two_negatives(self):
        self.assertEqual(subtract(-1, -3), -4)

    def test_decimal(self):
        self.assertAlmostEqual(subtract(5.9, 4.1), 1.8, places=1)

    def test_zero(self):
        self.assertEqual(subtract(0, 0), 0)

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

```

Multiply:



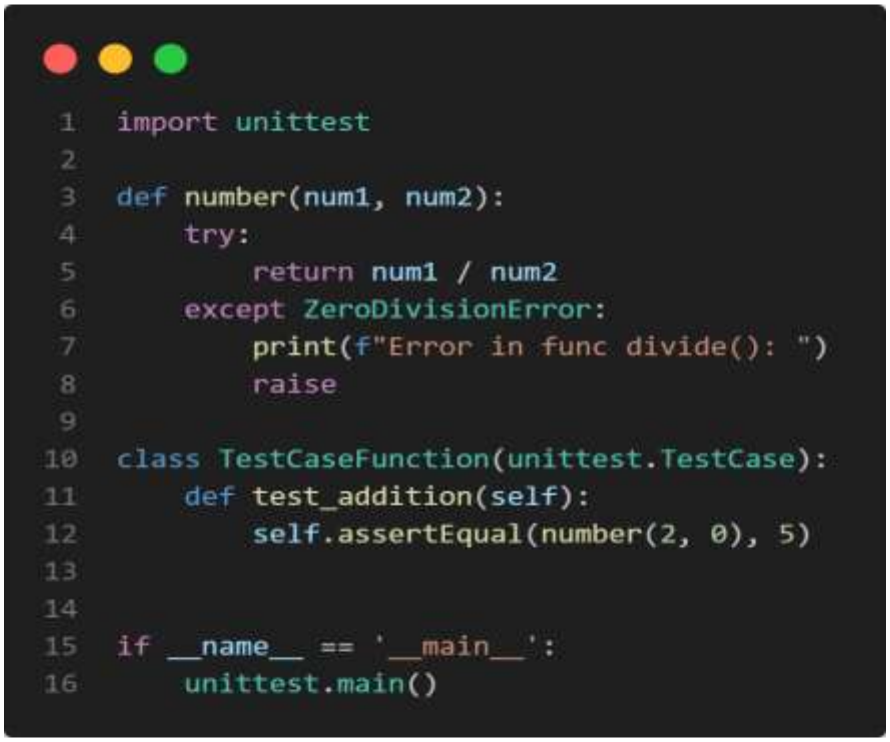
```
import unittest

def Multiply(a, b):
        return a * b


class TestforSubtraction(unittest.TestCase):
    def test_integers(self):
        self.assertEqual(Multiply(3, 3), 0)

    def test_zero_input(self):
        self.assertEqual(Multiply(5, 0), 5)

    def test_negative_and_positive(self):
        self.assertEqual(Multiply(-1, 5), 4)

    def test_two_negatives(self):
        self.assertEqual(Multiply(-1, -3), -4)

    def test_decimal(self):
        self.assertAlmostEqual(Multiply(5.9, 4.1), 1.8, places=1)

    def test_zero(self):
        self.assertEqual(Multiply(0, 0), 0)

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

Divide:



```python
import unittest


def div(num1, num2):
    try:
        return num1 / num2
    except ZeroDivisionError:
        print(f"Error in func divide(): ")
        raise

class TestCaseFunction(unittest.TestCase):
    def test_integers(self):
        self.assertEqual(div(2, 3), 5)

    def test_zero_input(self):
        self.assertEqual(div(0, 5), 5)

    def test_negative_and_positive(self):
        self.assertEqual(div(-1, 5), 4)

    def test_two_negatives(self):
        self.assertEqual(div(-2, -3), -5)

    def test_decimal(self):
        self.assertAlmostEqual(div(2.5, 3.1), 5.6, places=1)

    def test_zero(self):
        self.assertEqual(div(0, 0), 0)

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

5.



```python
import unittest

def number(num1, num2):
    try:
        return num1 / num2
    except ZeroDivisionError:
        print(f"Error in func divide(): ")
        raise

class TestCaseFunction(unittest.TestCase):
    def test_addition(self):
        self.assertEqual(number(2, 0), 5)


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