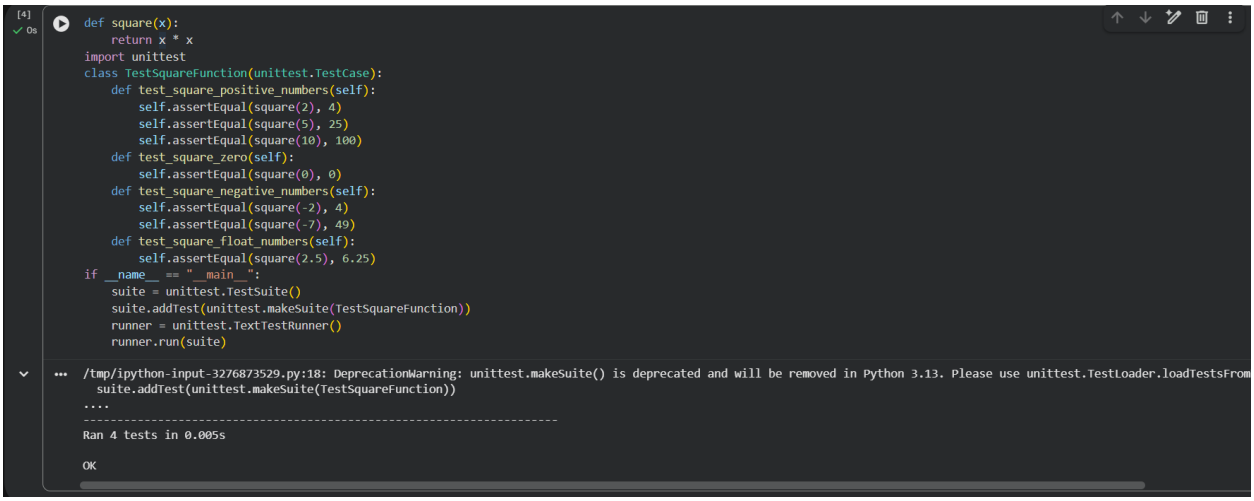
ASSIGNMENT (8..4)

Task 1: Developing a Utility Function Using TDD
Scenario
You are working on a small utility library for a larger software system. One of the required functions should calculate the square of a given number, and correctness is critical because other modules depend on it.



```
[4]
0s
def square(x):
    return x * x
import unittest
class TestSquareFunction(unittest.TestCase):
    def test_square_positive_numbers(self):
        self.assertEqual(square(2), 4)
        self.assertEqual(square(5), 25)
        self.assertEqual(square(10), 100)
    def test_square_zero(self):
        self.assertEqual(square(0), 0)
    def test_square_negative_numbers(self):
        self.assertEqual(square(-2), 4)
        self.assertEqual(square(-7), 49)
    def test_square_float_numbers(self):
        self.assertEqual(square(2.5), 6.25)
if __name__ == "__main__":
    suite = unittest.TestSuite()
    suite.addTest(unittest.makeSuite(TestSquareFunction))
    runner = unittest.TextTestRunner()
    runner.run(suite)

/tmp/ipython-input-3276873529.py:18: DeprecationWarning: unittest.makeSuite() is deprecated and will be removed in Python 3.13. Please use unittest.TestLoader.loadTestsFrom
  suite.addTest(unittest.makeSuite(TestSquareFunction))
....
----------------------------------------------------------------------
Ran 4 tests in 0.005s

OK
```

Task 2: Email Validation for a User Registration System
Scenario
You are developing the backend of a user registration system. One requirement is to validate user email addresses before storing them in the database

```python
import unittest
import re
def validate_email(email):
    if not isinstance(email, str):
        return False
    if not email:
        return False
    regex = r"^[a-zA-Z0-9](?:[._%+-](?![._%+-])|[a-zA-Z0-9])*[a-zA-Z0-9]@[a-zA-Z0-9](?:[.-](?![.-])|[a-zA-Z0-9])*[a-zA-Z0-9]\.[a-zA-Z]{2,}$"
    return re.match(regex, email) is not None

class TestEmailValidation(unittest.TestCase):
    def test_valid_email_standard(self):
        self.assertTrue(validate_email("test@example.com"))
        self.assertTrue(validate_email("john.doe@sub.domain.co.uk"))
        self.assertTrue(validate_email("user123_456@email-provider.net"))
        self.assertTrue(validate_email("test+alias@example.com"))
    def test_invalid_email_no_at(self):
        self.assertFalse(validate_email("testexample.com"))
        self.assertFalse(validate_email("johndoeexample.com"))
    def test_invalid_email_no_domain(self):
        self.assertFalse(validate_email("test@.com"))
        self.assertFalse(validate_email("test@example"))
    def test_invalid_email_no_username(self):
        self.assertFalse(validate_email("@example.com"))
    def test_invalid_email_bad_chars_username(self):
        self.assertFalse(validate_email("te st@example.com")) # Space
        self.assertFalse(validate_email("test!@example.com")) # Exclamation mark (not in regex)
        self.assertFalse(validate_email("test#@example.com")) # Hash (not in regex)
```

```python
    def test_empty_email(self):
        self.assertFalse(validate_email(""))
    def test_none_email(self):
        self.assertFalse(validate_email(None))
if __name__ == '__main__':
    suite = unittest.TestSuite()
    suite.addTest(unittest.makeSuite(TestEmailValidation))
    runner = unittest.TextTestRunner(verbosity=2)
    runner.run(suite)
```

```
/tmp/ipython-input-1106252505.py:79: DeprecationWarning: unittest.makeSuite() is deprecated and will be removed in Python 3.13. Please use unittest.TestLoader.loadTestsFrom
  suite.addTest(unittest.makeSuite(TestEmailValidation))
test_empty_email (__main__.TestEmailValidation.test_empty_email) ... ok
test_invalid_email_bad_chars_domain (__main__.TestEmailValidation.test_invalid_email_bad_chars_domain) ... ok
test_invalid_email_bad_chars_username (__main__.TestEmailValidation.test_invalid_email_bad_chars_username) ... ok
test_invalid_email_consecutive_dots_domain (__main__.TestEmailValidation.test_invalid_email_consecutive_dots_domain) ... ok
test_invalid_email_consecutive_dots_username (__main__.TestEmailValidation.test_invalid_email_consecutive_dots_username) ... ok
test_invalid_email_dot_start_end_domain (__main__.TestEmailValidation.test_invalid_email_dot_start_end_domain) ... ok
test_invalid_email_dot_start_end_username (__main__.TestEmailValidation.test_invalid_email_dot_start_end_username) ... ok
test_invalid_email_multiple_at (__main__.TestEmailValidation.test_invalid_email_multiple_at) ... ok
test_invalid_email_no_at (__main__.TestEmailValidation.test_invalid_email_no_at) ... ok
test_invalid_email_no_domain (__main__.TestEmailValidation.test_invalid_email_no_domain) ... ok
test_invalid_email_no_username (__main__.TestEmailValidation.test_invalid_email_no_username) ... ok
test_none_email (__main__.TestEmailValidation.test_none_email) ... ok
test_valid_email_standard (__main__.TestEmailValidation.test_valid_email_standard) ... ok

----------------------------------------------------------------------
Ran 13 tests in 0.024s

OK
```

Task 3: Decision Logic Development Using TDD

Scenario

In a grading or evaluation module, a function is required to determine the maximum value among three inputs. Accuracy is essential, as incorrect results could affect downstream decision logic.

```
[39]
def max_of_three(a, b, c):
    return max(a, b, c)
class TestMaxOfThree(unittest.TestCase):
    def test_all_positive_numbers(self):
        self.assertEqual(max_of_three(3, 7, 5), 7)
        self.assertEqual(max_of_three(10, 2, 8), 10)
    def test_with_negative_numbers(self):
        self.assertEqual(max_of_three(-1, -5, -3), -1)
        self.assertEqual(max_of_three(-10, 5, 0), 5)
    def test_with_equal_numbers(self):
        self.assertEqual(max_of_three(4, 4, 4), 4)
        self.assertEqual(max_of_three(2, 5, 5), 5)
    def test_with_zero(self):
        self.assertEqual(max_of_three(0, -1, -2), 0)
        self.assertEqual(max_of_three(0, 0, 3), 3)
    def test_with_float_numbers(self):
        self.assertEqual(max_of_three(2.5, 3.1, 1.9), 3.1)
if __name__ == "__main__":
    suite = unittest.TestSuite()
    suite.addTest(unittest.makeSuite(TestMaxOfThree))
    runner = unittest.TextTestRunner()
    runner.run(suite)

/tmp/ipython-input-1327369122.py:29: DeprecationWarning: unittest.makeSuite() is deprecated and will be removed in Python 3.13. Please use unittest.TestLoader.loadTestsFro
  suite.addTest(unittest.makeSuite(TestMaxOfThree))
.....
----------------------------------------------------------------------
Ran 5 tests in 0.006s

OK
```

Task 4: Shopping Cart Development with AI-Assisted TDD

Scenario

You are building a simple shopping cart module for an e-commerce application. The cart must support adding items, removing items, and calculating the total price accurately

```
[50]
✓ 0s
import unittest
class ShoppingCart:
    def __init__(self):
        self.items = {}
    def add_item(self, item_name, quantity, price):
        if quantity <= 0 or price < 0:
            return
        if item_name in self.items:
            self.items[item_name]['quantity'] += quantity
        else:
            self.items[item_name] = {'quantity': quantity, 'price': price}
    def remove_item(self, item_name, quantity):
        if item_name in self.items:
            if self.items[item_name]['quantity'] <= quantity:
                del self.items[item_name]
            else:
                self.items[item_name]['quantity'] -= quantity
    def calculate_total(self):
        total = 0.0
        for item_name, data in self.items.items():
            total += data['quantity'] * data['price']
        return total
class TestShoppingCart(unittest.TestCase):
    def test_add_single_item(self):
        cart = ShoppingCart()
        cart.add_item("Apple", 1, 1.00)
        self.assertEqual(cart.calculate_total(), 1.00)
    def test_add_multiple_of_same_item(self):
        cart = ShoppingCart()
        cart.add_item("Banana", 2, 0.50)
```

```
[50]
✓ 0s
            self.assertEqual(cart.calculate_total(), 4.00)
        def test_remove_partial_quantity(self):
            cart = ShoppingCart()
            cart.add_item("Pear", 5, 1.20)
            cart.remove_item("Pear", 2)
            self.assertAlmostEqual(cart.calculate_total(), 3.60) # 3 Pears * 1.20
        def test_calculate_total_empty_cart(self):
            cart = ShoppingCart()
            self.assertEqual(cart.calculate_total(), 0.00)
        def test_calculate_total_after_add_remove(self):
            cart = ShoppingCart()
            cart.add_item("Milk", 1, 3.00)
            cart.add_item("Bread", 2, 2.50)
            cart.remove_item("Bread", 1)
            # Expected: 1 Milk (3.00) + 1 Bread (2.50) = 5.50
            self.assertAlmostEqual(cart.calculate_total(), 5.50)
        def test_add_item_with_zero_quantity(self):
            cart = ShoppingCart()
            cart.add_item("Soda", 0, 1.50)
            self.assertEqual(cart.calculate_total(), 0.00)
        def test_add_item_with_negative_price(self):
            cart = ShoppingCart()
            cart.add_item("Discounted Item", 1, -2.00) # Negative price should be handled (e.g., ignore or error)
            self.assertEqual(cart.calculate_total(), 0.00) # Assuming negative price items are not added or calculated
    if __name__ == '__main__':
        suite = unittest.TestSuite()
        suite.addTest(unittest.makeSuite(TestShoppingCart))
        runner = unittest.TextTestRunner(verbosity=2)
        runner.run(suite)
```

```
        self.assertEqual(cart.calculate_total(), 0.00) # Assuming negative price items are not added or calculated
if __name__ == '__main__':
    suite = unittest.TestSuite()
    suite.addTest(unittest.makeSuite(TestShoppingCart))
    runner = unittest.TextTestRunner(verbosity=2)
    runner.run(suite)

/tmp/ipython-input-3267011651.py:72: DeprecationWarning: unittest.makeSuite() is deprecated and will be removed in Python 3.13. Please use unittest.TestLoader.loadTestsFromT
  suite.addTest(unittest.makeSuite(TestShoppingCart))
test_add_different_items (__main__.TestShoppingCart.test_add_different_items) ... ok
test_add_item_with_negative_price (__main__.TestShoppingCart.test_add_item_with_negative_price) ... ok
test_add_item_with_zero_quantity (__main__.TestShoppingCart.test_add_item_with_zero_quantity) ... ok
test_add_multiple_of_same_item (__main__.TestShoppingCart.test_add_multiple_of_same_item) ... ok
test_add_single_item (__main__.TestShoppingCart.test_add_single_item) ... ok
test_calculate_total_after_add_remove (__main__.TestShoppingCart.test_calculate_total_after_add_remove) ... ok
test_calculate_total_empty_cart (__main__.TestShoppingCart.test_calculate_total_empty_cart) ... ok
test_remove_existing_item_full_quantity (__main__.TestShoppingCart.test_remove_existing_item_full_quantity) ... ok
test_remove_non_existing_item (__main__.TestShoppingCart.test_remove_non_existing_item) ... ok
test_remove_partial_quantity (__main__.TestShoppingCart.test_remove_partial_quantity) ... ok

----------------------------------------------------------------------
Ran 10 tests in 0.018s

OK
```

Task 5: String Validation Module Using TDD

Scenario

You are working on a text-processing module where a function is required to identify whether a given string is a palindrome. The function must handle different cases and inputs reliably

```
[67]
✓ 0s
import unittest
import re
def is_palindrome(text):
    if not isinstance(text, str):
        return False
    normalized_text = re.sub(r'[^a-zA-Z0-9]', '', text).lower()
    return normalized_text == normalized_text[::-1]
class TestIsPalindrome(unittest.TestCase):
    def test_simple_palindrome(self):
        self.assertTrue(is_palindrome("madam"))
        self.assertTrue(is_palindrome("level"))
    def test_simple_non_palindrome(self):
        self.assertFalse(is_palindrome("hello"))
        self.assertFalse(is_palindrome("world"))
    def test_with_spaces(self):
        self.assertTrue(is_palindrome("nurses run"))
        self.assertTrue(is_palindrome("a man a plan a canal panama"))
    def test_with_punctuation(self):
        self.assertTrue(is_palindrome("A man, a plan, a canal: Panama."))
        self.assertTrue(is_palindrome("Eva, can I see bees in a cave?"))
    def test_case_insensitivity(self):
        self.assertTrue(is_palindrome("Racecar"))
        self.assertTrue(is_palindrome("Madam"))
    def test_empty_string(self):
        self.assertTrue(is_palindrome(""))
    def test_single_character(self):
        self.assertTrue(is_palindrome("a"))
        self.assertTrue(is_palindrome("Z"))
    def test_numbers_as_string(self):
        self.assertTrue(is_palindrome("121"))
        self.assertTrue(is_palindrome("1001"))
    def test_mixed_case_and_punctuation_non_palindrome(self):
        self.assertFalse(is_palindrome("This is not a palindrome!"))
        self.assertFalse(is_palindrome("Hello, World!"))
    def test_none_input(self):
        self.assertFalse(is_palindrome(None))
if __name__ == '__main__':
    suite = unittest.TestSuite()
    suite.addTest(unittest.makeSuite(TestIsPalindrome))
    runner = unittest.TextTestRunner(verbosity=2)
    runner.run(suite)
```

```
/tmp/ipython-input-2648525962.py:39: DeprecationWarning: unittest.makeSuite() is deprecated and will be removed in Python 3.13. Please use unittest.TestLoader.loadTestsFromT
  suite.addTest(unittest.makeSuite(TestIsPalindrome))
test_case_insensitivity (__main__.TestIsPalindrome.test_case_insensitivity) ... ok
test_empty_string (__main__.TestIsPalindrome.test_empty_string) ... ok
test_mixed_case_and_punctuation_non_palindrome (__main__.TestIsPalindrome.test_mixed_case_and_punctuation_non_palindrome) ... ok
test_none_input (__main__.TestIsPalindrome.test_none_input) ... ok
test_numbers_as_string (__main__.TestIsPalindrome.test_numbers_as_string) ... ok
test_simple_non_palindrome (__main__.TestIsPalindrome.test_simple_non_palindrome) ... ok
test_simple_palindrome (__main__.TestIsPalindrome.test_simple_palindrome) ... ok
test_single_character (__main__.TestIsPalindrome.test_single_character) ... ok
test_with_punctuation (__main__.TestIsPalindrome.test_with_punctuation) ... ok
test_with_spaces (__main__.TestIsPalindrome.test_with_spaces) ... ok

----------------------------------------------------------------------
Ran 10 tests in 0.029s

OK
```