**Test 1:**

**Test adding a book to the library.**

Code for Test:

import unittest

from library\_management.library import Library  # Import the Library class from the main code

class TestLibrary(unittest.TestCase):

    def test\_add\_book(self):

        """Test adding a book to the library."""

        library = Library()

        library.add\_book(1,"The Jungle Book", "Rudyard Kipling", 3)

        self.assertIn(1, library.books)  # Check if the book is added

        self.assertEqual(library.books[1], ["The Jungle Book", "Rudyard Kipling", 3])  # Verify book details

if \_\_name\_\_ == "\_\_main\_\_":

    unittest.main()

To check the code for testing:

class Library:

    def \_\_init\_\_(self):

        self.books = {}  # Dictionary initiated

    def add\_book(self, book\_id, title, author, copies):

        """Add a book to the library."""

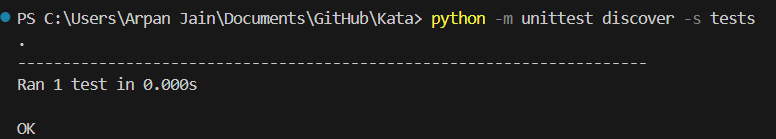
        if book\_id in self.books:

            self.books[book\_id][2] += copies  # Update number of copies if book exists

        else:

            self.books[book\_id] = [title, author, copies]  # Add new book

Screenshot of checking test running:



**Test 2:**

**Test viewing all books in the library.**

Code for Test:

import unittest

from library\_management.library import Library  # Import the Library class from the main code

class TestLibrary(unittest.TestCase):

    def test\_add\_book(self):

        """Test adding a book to the library."""

        library = Library()

        library.add\_book(1,"The Jungle Book", "Rudyard Kipling", 3)

        self.assertIn(1, library.books)  # Check if the book is added

        self.assertEqual(library.books[1], ["The Jungle Book", "Rudyard Kipling", 3])  # Verify book details

    def test\_view\_books(self):

        """Test viewing all books in the library."""

        library = Library()

        library.add\_book(1, "The Jungle Book", "Rudyard Kipling", 3)

        self.assertEqual(len(library.books), 1)  # Should have 1 book in the library

if \_\_name\_\_ == "\_\_main\_\_":

    unittest.main()

To check the code for testing:

class Library:

    def \_\_init\_\_(self):

        self.books = {}  # Dictionary initiated

    def add\_book(self, book\_id, title, author, copies):

        """Add a book to the library."""

        if book\_id in self.books:

            self.books[book\_id][2] += copies  # Update number of copies if book exists

        else:

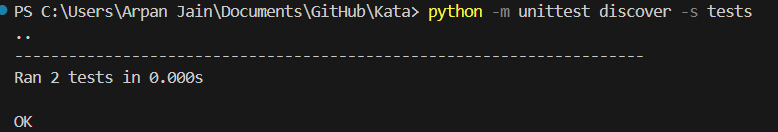
            self.books[book\_id] = [title, author, copies]  # Add new book

    def view\_books(self):

        """View all books in the library."""

        return self.books  # Simply return the books dictionary

Screenshot of checking test running:



**Test 3:**

**Test borrowing a book successfully.**

Code for Test:

import unittest

from library\_management.library import Library  # Import the Library class from the main code

class TestLibrary(unittest.TestCase):

    def test\_add\_book(self):

        """Test adding a book to the library."""

        library = Library()

        library.add\_book(1,"The Jungle Book", "Rudyard Kipling", 3)

        self.assertIn(1, library.books)  # Check if the book is added

        self.assertEqual(library.books[1], ["The Jungle Book", "Rudyard Kipling", 3])  # Verify book details

    def test\_view\_books(self):

        """Test viewing all books in the library."""

        library = Library()

        library.add\_book(1, "The Jungle Book", "Rudyard Kipling", 3)

        self.assertEqual(len(library.books), 1)  # Should have 1 book in the library

    def test\_borrow\_book\_success(self):

        """Test borrowing a book successfully."""

        library = Library()

        library.add\_book(1, "The Jungle Book", "Rudyard Kipling", 3)

        library.borrow\_book(1)

        self.assertEqual(library.books[1][2], 2)  # Copies should decrease by 1

if \_\_name\_\_ == "\_\_main\_\_":

    unittest.main()

To check the code for testing:

class Library:

    def \_\_init\_\_(self):

        self.books = {}  # Dictionary initiated

    def add\_book(self, book\_id, title, author, copies):

        """Add a book to the library."""

        if book\_id in self.books:

            self.books[book\_id][2] += copies  # Update number of copies if book exists

        else:

            self.books[book\_id] = [title, author, copies]  # Add new book

    def view\_books(self):

        """View all books in the library."""

        return self.books  # Simply return the books dictionary

    def borrow\_book(self, book\_id):

        """Borrow a book from the library if available."""

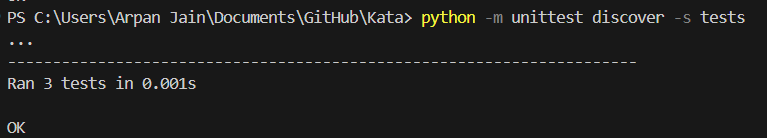
        if book\_id in self.books and self.books[book\_id][2] > 0:

            self.books[book\_id][2] -= 1  # Decrease available copies by 1

        else:

            "Book is not available in library"

Screenshot of checking test running:



**Test 4:**

**Test returning a book successfully.**

Code for Test:

import unittest

from library\_management.library import Library  # Import the Library class from the main code

class TestLibrary(unittest.TestCase):

    def test\_add\_book(self):

        """Test adding a book to the library."""

        library = Library()

        library.add\_book(1,"The Jungle Book", "Rudyard Kipling", 3)

        self.assertIn(1, library.books)  # Check if the book is added

        self.assertEqual(library.books[1], ["The Jungle Book", "Rudyard Kipling", 3])  # Verify book details

    def test\_view\_books(self):

        """Test viewing all books in the library."""

        library = Library()

        library.add\_book(1, "The Jungle Book", "Rudyard Kipling", 3)

        self.assertEqual(len(library.books), 1)  # Should have 1 book in the library

    def test\_borrow\_book\_success(self):

        """Test borrowing a book successfully."""

        library = Library()

        library.add\_book(1, "The Jungle Book", "Rudyard Kipling", 3)

        library.borrow\_book(1)

        self.assertEqual(library.books[1][2], 2)  # Copies should decrease by 1

    def test\_return\_book(self):

        """Test returning a book successfully."""

        library = Library()

        library.add\_book(1, "The Jungle Book", "Rudyard Kipling", 1)

        library.borrow\_book(1)  # Borrow the book

        library.return\_book(1)  # Return the book

        self.assertEqual(library.books[1][2], 1)  # Copies should return to original count

if \_\_name\_\_ == "\_\_main\_\_":

    unittest.main()

To check the code for testing:

class Library:

    def \_\_init\_\_(self):

        self.books = {}  # Dictionary initiated

    def add\_book(self, book\_id, title, author, copies):

        """Add a book to the library."""

        if book\_id in self.books:

            self.books[book\_id][2] += copies  # Update number of copies if book exists

        else:

            self.books[book\_id] = [title, author, copies]  # Add new book

    def view\_books(self):

        """View all books in the library."""

        return self.books  # Simply return the books dictionary

    def borrow\_book(self, book\_id):

        """Borrow a book from the library if available."""

        if book\_id in self.books and self.books[book\_id][2] > 0:

            self.books[book\_id][2] -= 1  # Decrease available copies by 1

        else:

            "Book is not available in library"

    def return\_book(self, book\_id):

        """Return a borrowed book to the library."""

        if book\_id in self.books:

            self.books[book\_id][2] += 1  # Increase available copies by 1

Screenshot of checking test running:

