```
import pandas as pd
import numpy as np
class BookLover:
    # Initializer
    def __init__(self, name, email, fav_genre, num_books = None, book_list = None):
        self.name = name
        self.email = email
        self.fav_genre = fav_genre
        self.num_books = 0 if num_books is None else num_books
        self.book_list = pd.DataFrame({'book_name': [], 'book_rating': []}) if book_list is None else book_list


    def add_book(self, book_name, rating):
        if (book_name in set(self.book_list['book_name'])) == True:
            print('This book already exists in the book list. Try a new book.')
            return False
        else:
            new_book = pd.DataFrame({'book_name': [book_name], 'book_rating': [rating]})
            self.book_list = pd.concat([self.book_list, new_book], ignore_index = True)
            self.num_books += 1


    def has_read(self, book_name):
        return book_name in set(self.book_list['book_name'])

    def fav_books(self):
        return self.book_list[self.book_list['book_rating'] > 3]from booklover import BookLover
import unittest

class BookLoverTestSuite(unittest.TestCase):

    def test_1_add_book(self):

        person1 = BookLover('Rishi','rs@gmail.com','humor')
        person1.add_book('Three body problem', 4)
        test = 'Three body problem'
        res = test in set(person1.book_list['book_name'])
        print(res)
        self.assertTrue(res, 'Book was not in list')


    def test_2_add_book(self):

        person1 = BookLover('Rishi','rs@gmail.com','humor')
        person1.add_book('Three body problem', 4)
        person1.add_book('Three body problem', 4)
        count = len(person1.book_list[person1.book_list['book_name'] == 'Three body problem'])
        print('Current List: ' )
        print(person1.book_list)
        self.assertEqual(count, 1, 'Book was added more than once')

    def test_3_has_read(self):

        person1 = BookLover('Rishi','rs@gmail.com','humor')
        person1.add_book('Three body problem', 4)
        person1.add_book('Book2', 2)
        person1.add_book('Book3', 3)
        test = person1.has_read('Book2')
        print(person1.has_read('Book2'))
        self.assertTrue(test, 'This book has not been read')

    def test_4_has_read(self):

        person1 = BookLover('Rishi','rs@gmail.com','humor')
        person1.add_book('Three body problem', 4)
```

```
        person1.add_book('Book2', 2)
        person1.add_book('Book3', 3)
        test = person1.has_read('Book4')
        self.assertFalse(test, 'Test is not False')

    def test_5_num_books_read(self):

        person1 = BookLover('Rishi','rs@gmail.com','humor')
        person1.add_book('Three body problem', 4)
        person1.add_book('Book2', 2)
        person1.add_book('Book3', 3)
        self.assertEqual(person1.num_books, 3, 'Number of books read does not match expected value!')

    def test_6_fav_books(self):
        person1 = BookLover('Rishi','rs@gmail.com','humor')
        person1.add_book('Three body problem', 4)
        person1.add_book('Book2', 2)
        person1.add_book('Book3', 3)
        person1.add_book('Book4', 5)
        test = person1.fav_books()

        for x in set(test.book_rating):
            if x <= 3:
                valid = False
                break
            else:
                valid = True

        print(valid)
        self.assertTrue(valid, 'One of the returned ratings is not greater than 3')


if __name__ == '__main__':
    unittest.main(verbosity=3)test_1_add_book (__main__.BookLoverTestSuite) ... ok
test_2_add_book (__main__.BookLoverTestSuite) ... ok
test_3_has_read (__main__.BookLoverTestSuite) ... ok
test_4_has_read (__main__.BookLoverTestSuite) ... ok
test_5_num_books_read (__main__.BookLoverTestSuite) ... ok
test_6_fav_books (__main__.BookLoverTestSuite) ... ok

----------------------------------------------------------------------
Ran 6 tests in 0.047s

OK
```

```
import pandas as pd
import numpy as np
class BookLover:
    # Initializer
    def __init__(self, name, email, fav_genre, num_books = None, book_list = None):
        self.name = name
        self.email = email
        self.fav_genre = fav_genre
        self.num_books = 0 if num_books is None else num_books
        self.book_list = pd.DataFrame({'book_name': [], 'book_rating': []}) if book_list is None else book_list


    def add_book(self, book_name, rating):
        if (book_name in set(self.book_list['book_name'])) == True:
            print('This book already exists in the book list. Try a new book.')
            return False
        else:
            new_book = pd.DataFrame({'book_name': [book_name], 'book_rating': [rating]})
            self.book_list = pd.concat([self.book_list, new_book], ignore_index = True)
            self.num_books += 1


    def has_read(self, book_name):
        return book_name in set(self.book_list['book_name'])

    def fav_books(self):
        return self.book_list[self.book_list['book_rating'] > 3]from booklover import BookLover
import unittest

class BookLoverTestSuite(unittest.TestCase):

    def test_1_add_book(self):

        person1 = BookLover('Rishi','rs@gmail.com','humor')
        person1.add_book('Three body problem', 4)
        test = 'Three body problem'
        res = test in set(person1.book_list['book_name'])
        print(res)
        self.assertTrue(res, 'Book was not in list')


    def test_2_add_book(self):

        person1 = BookLover('Rishi','rs@gmail.com','humor')
        person1.add_book('Three body problem', 4)
        person1.add_book('Three body problem', 4)
        count = len(person1.book_list[person1.book_list['book_name'] == 'Three body problem'])
        print('Current List: ' )
        print(person1.book_list)
        self.assertEqual(count, 1, 'Book was added more than once')

    def test_3_has_read(self):

        person1 = BookLover('Rishi','rs@gmail.com','humor')
        person1.add_book('Three body problem', 4)
        person1.add_book('Book2', 2)
        person1.add_book('Book3', 3)
        test = person1.has_read('Book2')
        print(person1.has_read('Book2'))
        self.assertTrue(test, 'This book has not been read')

    def test_4_has_read(self):

        person1 = BookLover('Rishi','rs@gmail.com','humor')
        person1.add_book('Three body problem', 4)
```

```
        person1.add_book('Book2', 2)
        person1.add_book('Book3', 3)
        test = person1.has_read('Book4')
        self.assertFalse(test, 'Test is not False')

    def test_5_num_books_read(self):

        person1 = BookLover('Rishi','rs@gmail.com','humor')
        person1.add_book('Three body problem', 4)
        person1.add_book('Book2', 2)
        person1.add_book('Book3', 3)
        self.assertEqual(person1.num_books, 3, 'Number of books read does not match expected value!')

    def test_6_fav_books(self):
        person1 = BookLover('Rishi','rs@gmail.com','humor')
        person1.add_book('Three body problem', 4)
        person1.add_book('Book2', 2)
        person1.add_book('Book3', 3)
        person1.add_book('Book4', 5)
        test = person1.fav_books()

        for x in set(test.book_rating):
            if x <= 3:
                valid = False
                break
            else:
                valid = True

        print(valid)
        self.assertTrue(valid, 'One of the returned ratings is not greater than 3')


if __name__ == '__main__':
    unittest.main(verbosity=3)test_1_add_book (__main__.BookLoverTestSuite) ... ok
test_2_add_book (__main__.BookLoverTestSuite) ... ok
test_3_has_read (__main__.BookLoverTestSuite) ... ok
test_4_has_read (__main__.BookLoverTestSuite) ... ok
test_5_num_books_read (__main__.BookLoverTestSuite) ... ok
test_6_fav_books (__main__.BookLoverTestSuite) ... ok

----------------------------------------------------------------------
Ran 6 tests in 0.047s

OK
```