```
1  import pandas as pd
2
3  class BookLover:
4
5      def __init__(self,name,email,fav_genre,num_books=0,book_list = pd.DataFrame({'book_name':[],
'book_rating':[]})):
6          self.name = name
7          self.email = email
8          self.fav_genre = fav_genre
9          self.num_books = num_books
10         self.book_list = book_list
11
12     def add_book(self,book_name,book_rating):
13
14         if self.book_list['book_name'].isin([book_name]).any():
15             print("Book Name already Present")
16         else:
17             new_book = pd.DataFrame({'book_name': [book_name], 'book_rating': [book_rating]})
18             self.book_list = pd.concat([self.book_list, new_book], ignore_index=True)
19
20     def has_read(self,book_name):
21         return self.book_list['book_name'].isin([book_name]).any()
22
23     def num_books_read(self):
24         return len(self.book_list)
25
26     def fav_books(self):
27         return self.book_list[self.book_list['book_rating'] > 3]import unittest
28 from booklover import BookLover
29
30 class BookLoverTestSuite(unittest.TestCase):
31
32
33     def test_1_add_book(self):
34         self.booklover.add_book("Percy Jackson", 2)
35         self.assertTrue(self.booklover.has_read("Percy Jackson"))
36
37     def test_2_add_book(self):
38         # add the same book twice. Test if it's in 'book_list' only once.
39         self.booklover.add_book("Percy Jackson", 2)
40         self.booklover.add_book("Percy Jackson", 2)
41         self.assertEqual(len(self.booklover.book_list), 1)
42
43     def test_3_has_read(self):
44         # pass a book in the list and test if the answer is 'True'.
45         self.booklover.add_book("Percy Jackson", 2)
46         self.assertTrue(self.booklover.has_read("Percy Jackson"))
47
48     def test_4_has_read(self):
49         # pass a book NOT in the list and use 'assert False' to test the answer is 'True'
50         self.assertFalse(self.booklover.has_read("Percy Jackson Part 2"))
51
52     def test_5_num_books_read(self):
53         # add some books to the list, and test num_books matches expected.
54         self.booklover.add_book("Percy Jackson", 2)
55         self.booklover.add_book("Tale of Two Cities", 1)
56         self.booklover.add_book("Eragon", 5)
57         self.booklover.add_book("Life of Pi", 4)
58         self.booklover.add_book("A new book", 1)
59         self.booklover.add_book("A bad book", 1)
60         self.booklover.add_book("An ok book", 3)
61
62         assert self.booklover.num_books_read() == 7
63
64     def test_6_fav_books(self):
65         # add some books with ratings to the list, making sure some of them have rating > 3.Â
```

```
        self.booklover.add_book("Percy Jackson", 2)
        self.booklover.add_book("Tale of Two Cities", 1)
        self.booklover.add_book("Eragon", 5)
        self.booklover.add_book("Life of Pi", 4)

        fav_books = self.booklover.fav_books()
        self.assertEqual(len(fav_books), 2)
        self.assertTrue((fav_books['book_rating'] > 3).all())

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 0.023s

OK
```