```
1
        import pandas as pd
  2
        import numpy as np
   3
        class BookLover:
   4
           # Initializer
  5
           def __init__(self, name, email, fav_genre, num_books = None, book_list = None):
             self.name = name
  6
  7
             self.email = email
  8
             self.fav_genre = fav_genre
  9
             self.num_books = 0 if num_books is None else num_books
  10
              self.book_list = pd.DataFrame({'book_name': [], 'book_rating': []}) if book_list is None else
book list
  11
  12
  13
           def add_book(self, book_name, rating):
  14
             if (book_name in set(self.book_list['book_name'])) == True:
  15
               print('This book already exists in the book list. Try a new book.')
  16
               return False
  17
             else:
  18
               new_book = pd.DataFrame({'book_name': [book_name], 'book_rating': [rating]})
  19
               self.book_list = pd.concat([self.book_list, new_book], ignore_index = True)
  20
               self.num books += 1
  21
  22
  23
           def has_read(self, book_name):
  24
             return book_name in set(self.book_list['book_name'])
  25
  26
           def fav_books(self):
  27
             return self.book_list[self.book_list['book_rating'] > 3]from booklover import BookLover
  28
        import unittest
  29
  30
        class BookLoverTestSuite(unittest.TestCase):
  31
  32
           def test_1_add_book(self):
  33
  34
             person1 = BookLover('Rishi','rs@gmail.com','humor')
  35
             person1.add_book('Three body problem', 4)
  36
             test = 'Three body problem'
  37
             res = test in set(person1.book_list['book_name'])
  38
             print(res)
  39
             self.assertTrue(res, 'Book was not in list')
  40
  41
  42
           def test_2_add_book(self):
  43
  44
             person1 = BookLover('Rishi','rs@gmail.com','humor')
  45
             person1.add_book('Three body problem', 4)
             person1.add_book('Three body problem', 4)
  46
  47
             count = len(person1.book_list[person1.book_list['book_name'] == 'Three body problem'])
  48
             print('Current List: ')
  49
             print(person1.book_list)
  50
             self.assertEqual(count, 1, 'Book was added more than once')
  51
  52
           def test_3_has_read(self):
  53
  54
             person1 = BookLover('Rishi','rs@gmail.com','humor')
  55
             person1.add_book('Three body problem', 4)
  56
             person1.add_book('Book2', 2)
  57
             person1.add_book('Book3', 3)
  58
             test = person1.has read('Book2')
  59
             print(person1.has_read('Book2'))
  60
             self.assertTrue(test, 'This book has not been read')
  61
  62
           def test_4_has_read(self):
  63
  64
             person1 = BookLover('Rishi','rs@gmail.com','humor')
```

person1.add_book('Three body problem', 4)

```
66
           person1.add_book('Book2', 2)
67
           person1.add_book('Book3', 3)
68
           test = person1.has_read('Book4')
69
           self.assertFalse(test, 'Test is not False')
70
71
        def test_5_num_books_read(self):
72
73
           person1 = BookLover('Rishi','rs@gmail.com','humor')
74
           person1.add_book('Three body problem', 4)
75
           person1.add book('Book2', 2)
76
           person1.add book('Book3', 3)
77
           self.assertEqual(person1.num_books, 3, 'Number of books read does not match expected value!')
78
79
         def test_6_fav_books(self):
           person1 = BookLover('Rishi','rs@gmail.com','humor')
80
81
           person1.add_book('Three body problem', 4)
82
           person1.add book('Book2', 2)
83
           person1.add_book('Book3', 3)
84
           person1.add_book('Book4', 5)
85
           test = person1.fav_books()
86
           for x in set(test.book_rating):
87
88
             if x \le 3:
89
                valid = False
90
                break
91
             else:
92
                valid = True
93
94
           print(valid)
95
           self.assertTrue(valid, 'One of the returned ratings is not greater than 3')
96
97
98
      if name == ' main ':
99
        unittest.main(verbosity=3)test_1_add_book (__main__.BookLoverTestSuite) ... ok
      test_2_add_book (__main__.BookLoverTestSuite) ... ok
100
101
      test_3_has_read (__main__.BookLoverTestSuite) ... ok
102
      test 4 has read ( main .BookLoverTestSuite) ... ok
      test_5_num_books_read (__main__.BookLoverTestSuite) ... ok
103
104
      test 6 fav books ( main .BookLoverTestSuite) ... ok
105
106
107
      Ran 6 tests in 0.047s
108
109
      OK
```

```
1
        import pandas as pd
  2
        import numpy as np
   3
        class BookLover:
   4
           # Initializer
  5
           def __init__(self, name, email, fav_genre, num_books = None, book_list = None):
             self.name = name
  6
  7
             self.email = email
  8
             self.fav_genre = fav_genre
  9
             self.num_books = 0 if num_books is None else num_books
  10
              self.book_list = pd.DataFrame({'book_name': [], 'book_rating': []}) if book_list is None else
book list
  11
  12
  13
           def add_book(self, book_name, rating):
  14
             if (book_name in set(self.book_list['book_name'])) == True:
  15
               print('This book already exists in the book list. Try a new book.')
  16
               return False
  17
             else:
  18
               new_book = pd.DataFrame({'book_name': [book_name], 'book_rating': [rating]})
  19
               self.book_list = pd.concat([self.book_list, new_book], ignore_index = True)
  20
               self.num books += 1
  21
  22
  23
           def has_read(self, book_name):
  24
             return book_name in set(self.book_list['book_name'])
  25
  26
           def fav_books(self):
  27
             return self.book_list[self.book_list['book_rating'] > 3]from booklover import BookLover
  28
        import unittest
  29
  30
        class BookLoverTestSuite(unittest.TestCase):
  31
  32
           def test_1_add_book(self):
  33
  34
             person1 = BookLover('Rishi','rs@gmail.com','humor')
  35
             person1.add_book('Three body problem', 4)
  36
             test = 'Three body problem'
  37
             res = test in set(person1.book_list['book_name'])
  38
             print(res)
  39
             self.assertTrue(res, 'Book was not in list')
  40
  41
  42
           def test_2_add_book(self):
  43
  44
             person1 = BookLover('Rishi','rs@gmail.com','humor')
  45
             person1.add_book('Three body problem', 4)
             person1.add_book('Three body problem', 4)
  46
  47
             count = len(person1.book_list[person1.book_list['book_name'] == 'Three body problem'])
  48
             print('Current List: ')
  49
             print(person1.book_list)
  50
             self.assertEqual(count, 1, 'Book was added more than once')
  51
  52
           def test_3_has_read(self):
  53
  54
             person1 = BookLover('Rishi','rs@gmail.com','humor')
  55
             person1.add_book('Three body problem', 4)
  56
             person1.add_book('Book2', 2)
  57
             person1.add_book('Book3', 3)
  58
             test = person1.has read('Book2')
  59
             print(person1.has_read('Book2'))
  60
             self.assertTrue(test, 'This book has not been read')
  61
  62
           def test_4_has_read(self):
  63
  64
             person1 = BookLover('Rishi','rs@gmail.com','humor')
```

person1.add_book('Three body problem', 4)

```
66
           person1.add_book('Book2', 2)
67
           person1.add_book('Book3', 3)
68
           test = person1.has_read('Book4')
69
           self.assertFalse(test, 'Test is not False')
70
71
        def test_5_num_books_read(self):
72
73
           person1 = BookLover('Rishi','rs@gmail.com','humor')
74
           person1.add_book('Three body problem', 4)
75
           person1.add book('Book2', 2)
76
           person1.add book('Book3', 3)
77
           self.assertEqual(person1.num_books, 3, 'Number of books read does not match expected value!')
78
79
         def test_6_fav_books(self):
           person1 = BookLover('Rishi','rs@gmail.com','humor')
80
81
           person1.add_book('Three body problem', 4)
82
           person1.add book('Book2', 2)
83
           person1.add_book('Book3', 3)
84
           person1.add_book('Book4', 5)
85
           test = person1.fav_books()
86
           for x in set(test.book_rating):
87
88
             if x \le 3:
89
                valid = False
90
                break
91
             else:
92
                valid = True
93
94
           print(valid)
95
           self.assertTrue(valid, 'One of the returned ratings is not greater than 3')
96
97
98
      if name == ' main ':
99
        unittest.main(verbosity=3)test_1_add_book (__main__.BookLoverTestSuite) ... ok
      test_2_add_book (__main__.BookLoverTestSuite) ... ok
100
101
      test_3_has_read (__main__.BookLoverTestSuite) ... ok
102
      test 4 has read ( main .BookLoverTestSuite) ... ok
      test_5_num_books_read (__main__.BookLoverTestSuite) ... ok
103
104
      test 6 fav books ( main .BookLoverTestSuite) ... ok
105
106
107
      Ran 6 tests in 0.047s
108
109
      OK
```