```
'''
Module for class BookLover
'''

from BookAlreadyExistsInBookListException import BookAlreadyExistsInBookListException
import pandas as pd

class BookLover:
    '''
    A BookLover represents a person with a name, an email address, a favorite genre, a number of read-books, and a read-book list
    who can add a read book, return a number of indicator of whether this BookLover has read a book, return a number of read books, and return a list of favorite books.

    Instance variables:
        name: str -- the name of this BookLover
        email: str -- the email and unique identifier of this BookLover
        fav_genre: str -- the favorite genre of this BookLover (e.g., 'mystery', 'fantasy', 'historical fiction')
        num_books: int -- the number of books that this BookLover has read
        book_list: pd.DataFrame --
            a data frame with columns labeled 'book_name' and 'book_rating' containing read books (i.e., titles of books this BookLover has read and
            this BookLover's ratings of those books on a scale of 1 to 5, where 1 means this BookLover did not like the book at all, and 5 means this BookLover loved the book)
            (e.g., pd.DataFrame(columns =
                ['book_name', 'book_rating'], data = [
                ['Jane Eyre', 4],
                ['Fight Club', 3],
                ['The Divine Comedy', 5],
                ['The Popol Vuh', 5]]
            ))

    Public methods:
        __init__
    '''

    def __init__(
        self,
        name,
        email,
        fav_genre,
        num_books = 0,
         book_list = pd.DataFrame({'book_name':[], 'book_rating':[]}).astype(dtype = {'book_name': str, 'book_rating': int})
    ):
        '''
        Initializes a BookLover

        Keyword arguments:
            name: str -- a name with which to initialize this BookLover
            email: str --an email address and unique identifier with which to initialize this BookLover
            fav_genre: str -- a favorite genre with which to initialize this BookLover
            num_books: int -- a number of read books with which to initialize this BookLover. Default 0.
            book_list: pd.DataFrame --
                a data frame with which to initialize this BookLover. Has columns labeled 'book_name' and 'book_rating' and contains read books (i.e., titles of books this BookLover
                has read and this BookLover's ratings of those books on a scale of 0 to 5, where 0 means this BookLover did not like the book at all, and 5 means this BookLover loved
                the book). Default pd.DataFrame({'book_name':[], 'book_rating':[]}).

        Return values:
            none

        Side effects:
            Initializes this BookLover's name, email address, favorite genre, number of read books, and read-book list
```

```
59
60          Exceptions raised:
61              none
62
63          Restrictions on when this method can be called:
64              May not be called directly
65          '''
66
67          self.name = name
68          self.email = email
69          self.fav_genre = fav_genre
70          self.num_books = num_books
71          # book_list references an empty data frame of the list __defaults__ of this method of BookLover; this
data frame is a class attribute. self.book_list references a instance attribute.
72          self.book_list = book_list.copy()
73
74      def _get_list_of_book_names(self):
75          self.series_of_book_names = self.book_list['book_name']
76          self.list_of_book_names = self.series_of_book_names.to_list()
77          return self.list_of_book_names
78
79      def add_book(self, book_name, rating):
80          '''
81          Adds a read book to this BookLover's read-book list
82
83          Keyword arguments:
84              book_name: str -- the name of a book to add to this BookLover's read-book list
85              rating: int -- a book rating from 0 to 5 to add to this BookLover's read-book list
86
87          Return values:
88              none
89
90          Side effects:
91              Adds a book to this BookLover's read-book list
92
93          Exceptions raised:
94              BookAlreadyExistsException if a book already exists in this BookLover's read-book list
95
96          Restrictions on when this method can be called:
97              none
98          '''
99
100         if (book_name in self._get_list_of_book_names()):
101             raise BookAlreadyExistsInBookListException(book_name + ' already exists in the read-book list of
' + self.name + '.')
102         else:
103             # Raf Alvarado's solution:
104             #new_book = pd.DataFrame({
105             #    'book_name': [book_name],
106             #    'book_rating': [rating]
107             #})
108             #self.book_list = pd.concat([self.book_list, new_book], ignore_index = True)
109             self.book_list.loc[len(self.book_list.index)] = [book_name, rating]
110             self.num_books += 1
111
112     def has_read(self, book_name):
113         '''
114         Returns an indicator of whether this BookLover has read a book with a specified name
115
116         Keyword arguments:
117             book_name: str -- the name of a book for which to return an indicator of whether this BookLover
has read that book
118
119         Return values:
120             True if this BookLover has read a book with a specified name
121             False if this BookLover has not read a book with a specified name
```

```
122
123         Side effects:
124             none
125
126         Exceptions raised:
127             none
128
129         Restrictions on when this method can be called:
130             none
131         '''
132
133         if (book_name in self._get_list_of_book_names()):
134             return True
135         else:
136             return False
137
138     def num_books_read(self):
139         '''
140         Returns the number of books this BookLover has read
141
142         Keyword arguments:
143             none
144
145         Return values:
146             the total number of books this BookLover has read
147
148         Side effects:
149             none
150
151         Exceptions raised:
152             none
153
154         Restrictions on when this method can be called:
155             none
156         '''
157
158         return self.num_books
159
160     def fav_books(self):
161         '''
162         Returns a data frame of books to which this BookLover has given ratings greater than 3
163
164         Keyword arguments:
165             none
166
167         Return values:
168             a data frame of books to which this BookLover has given ratings greater than 3
169
170         Side effects:
171             none
172
173         Exceptions raised:
174             none
175
176         Restrictions on when this method can be called:
177             none
178         '''
179
180         return self.book_list[self.book_list['book_rating'] > 3]'''
181 Module for class BookLoverTestSuite, which tests the methods of a BookLover
182 '''
183
184 import unittest
185 from booklover import *
186
187 class BookLoverTestSuite(unittest.TestCase):
```

```
    '''
    Tests the methods of a BookLover

    Instance variables:
        none

    Public methods:
        test_1_init
    '''

    def setUp(self):
        '''
        Creates a instance variable with a value of a BookLover

        Keyword arguments:
            none

        Return values:
            none

        Side effects:
            Creates an instance variable with a value of a BookLover

        Exceptions raised:
            none

        Restrictions on when this method can be called:
            none
        '''

        self.book_lover = BookLover('Han Solo', 'hsolo@millenniumfalcon.com', 'scifi')

    def test_0_init(self):
        '''
        Tests BookLover.__init__

        Keyword arguments:
            none

        Return values:
            none

        Side effects:
            Tests whether a BookLover's name, email, favorite genre, number of read books, and book list are equal to expected values

        Exceptions raised:
            AssertionError if a BookLover's name, email, favorite genre, number of read books, or book list is not equal to an expected value

        Restrictions on when this method can be called:
            none
        '''

        self.assertEqual(self.book_lover.name, 'Han Solo')
        self.assertEqual(self.book_lover.email, 'hsolo@millenniumfalcon.com')
        self.assertEqual(self.book_lover.fav_genre, 'scifi')
        self.assertEqual(self.book_lover.num_books, 0)
            self.assertTrue(self.book_lover.book_list.equals(pd.DataFrame({'book_name':[], 'book_rating':[]}).astype(dtype = {'book_name': 'str', 'book_rating': 'int'})))


    def test_1_add_book(self):
        '''
        Tests BookLover.add_book by confirming that a book was successfully added to a BookLover's read book list
```

```
250
251         Keyword arguments:
252             none
253
254         Return values:
255             none
256
257         Side effects:
258             Adds a book to the read-book list of a BookLover and tests whether the book is in the BookLover's
read-book list
259
260         Exceptions raised:
261             AssertionError if a book, including book name and rating, is not correctly added to the read-book
list of a BookLover
262
263         Restrictions on when this method can be called:
264             none
265         '''
266
267         self.book_lover.add_book('Star Wars: A New Hope', 5)
268          self.assertTrue(self.book_lover.book_list.equals(pd.DataFrame({'book_name':['Star Wars: A New
Hope'], 'book_rating':[5]})))
269
270     def test_2_add_book(self):
271         '''
272         Tests BookLover.add_book by running test_1_add_book twice, catching a BookAlreadyExistsInBook-
ListException thrown when an attempt is made to add a book already in a BookLover's read-book list, and confirm-
ing that only one book was added to the BookLover's read-book list
273
274         Keyword arguments:
275             none
276
277         Return values:
278             none
279
280         Side effects:
281             Runs test_1_add_book twice, catches a BookAlreadyExistsInBookListException thrown when an
attempt is made to add a book already in a BookLover's read book list, and confirms that only one book was added
to the BookLover's read-book list
282
283         Exceptions raised:
284             AssertionError if test_1_add_book succeeds or a BookLover's read-book list does not contain one
book
285
286         Restrictions on when this method can be called:
287             none
288         '''
289
290         self.test_1_add_book()
291         try:
292             self.test_1_add_book()
293             self.fail()
294         except BookAlreadyExistsInBookListException as e:
295             pass
296          self.assertTrue(self.book_lover.book_list.equals(pd.DataFrame({'book_name':['Star Wars: A New
Hope'], 'book_rating':[5]})))
297
298     def test_3_has_read(self):
299         '''
300          Tests BookLover.has_read by ensuring that has_read returns True when the name of a book in a
BookLover's read-book list is provided to has_read
301
302         Keyword arguments:
303             none
304
305         Return values:
```

```
306             none
307
308         Side effects:
309             Ensures that has_read returns True when the name of a book in a BookLover's read-book list is pro-
vided to has_read
310
311         Exceptions raised:
312             AssertionError if has_read returns False when the name of a book in a BookLover's read-book list is
provided to has_read
313
314         Restrictions on when this method can be called:
315             none
316         '''
317
318         self.test_1_add_book()
319         self.assertTrue(self.book_lover.has_read('Star Wars: A New Hope'))
320
321     def test_4_has_read(self):
322         '''
323         Tests BookLover.has_read by ensuring that has_read returns False when the name of a book not in a
BookLover's read-book list is provided to has_read
324
325         Keyword arguments:
326             none
327
328         Return values:
329             none
330
331         Side effects:
332             Ensures that has_read returns False when the name of a book not in a BookLover's read-book list is
provided to has_read
333
334         Exceptions raised:
335             AssertionError if has_read returns True when the name of a book not in a BookLover's read-book
list is provided to has_read
336
337         Restrictions on when this method can be called:
338             none
339         '''
340
341         self.test_1_add_book()
342         self.assertFalse(self.book_lover.has_read('Star Wars: Empire Strikes Back'))
343
344     def test_5_num_books_read(self):
345         '''
346         Tests BookLover.num_books_read by ensuring that num_books_read returns 2 when a BookLover's
read-book list contains two books
347
348         Keyword arguments:
349             none
350
351         Return values:
352             none
353
354         Side effects:
355             Ensures that num_books_read returns 2 when a BookLover's read-book list contains two books
356
357         Exceptions raised:
358             AssertionError if num_books_read returns a number other than 2 when a BookLover's read-book
list contains two books
359
360         Restrictions on when this method can be called:
361             none
362         '''
363
364         self.test_1_add_book()
```

```
        self.book_lover.add_book('Star Wars: Empire Strikes Back', 4)
        self.assertEqual(self.book_lover.num_books_read(), 2)

    def test_6_fav_books(self):
        '''
        Tests BookLover.fav_books by ensuring that fav_books returns a data frame of two books with ratings greater than 3 when a BookLover's read-book list contains two books with rating greater than 3 and one book with rating 3

        Keyword arguments:
            none

        Return values:
            none

        Side effects:
            Ensures that fav_books returns a data frame of two books when a BookLover's read-book list contains two books with rating greater than 3 and one book with rating 3

        Exceptions raised:
            AssertionError if fav_book does not return a data frame of two books with ratings greater than 3
            when a BookLover's read-book list contains two books with rating greater than 3 and one book with rating 3

        Restrictions on when this method can be called:
            none
        '''

        self.test_1_add_book()
        self.book_lover.add_book('Star Wars: Empire Strikes Back', 4)
        self.book_lover.add_book('Star Wars: Return of the Jedi', 3)
        data_frame_of_favorite_books = self.book_lover.fav_books()
        self.assertTrue(
            data_frame_of_favorite_books.equals(
                pd.DataFrame(
                    {
                        'book_name': [
                            'Star Wars: A New Hope',
                            'Star Wars: Empire Strikes Back'
                        ],
                        'book_rating': [
                            5,
                            4
                        ]
                    }
                )
            )
        )
        series_of_indicators_of_whether_ratings_are_greater_than_3 = (data_frame_of_favorite_books['book_rating'] > 3)
        self.assertTrue(series_of_indicators_of_whether_ratings_are_greater_than_3.all())

    def tearDown(self):
        '''
        Deletes an instance variable with a value of a BookLover

        Keyword arguments:
            none

        Return values:
            none

        Side effects:
            Deletes an instance variable with a value of a BookLover

        Exceptions raised:
```

```
            none

        Restrictions on when this method can be called:
            none
        '''

        del self.book_lover

if __name__ == '__main__':
    verbose = 3
    unittest.main(verbosity = verbose)test_0_init (__main__.BookLoverTestSuite) ... ok
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 7 tests in 0.046s

OK
```

```
'''
Module for class BookLover
'''

from BookAlreadyExistsInBookListException import BookAlreadyExistsInBookListException
import pandas as pd

class BookLover:
    '''
    A BookLover represents a person with a name, an email address, a favorite genre, a number of read-books, and a read-book list
    who can add a read book, return a number of indicator of whether this BookLover has read a book, return a number of read books, and return a list of favorite books.

    Instance variables:
        name: str -- the name of this BookLover
        email: str -- the email and unique identifier of this BookLover
        fav_genre: str -- the favorite genre of this BookLover (e.g., 'mystery', 'fantasy', 'historical fiction')
        num_books: int -- the number of books that this BookLover has read
        book_list: pd.DataFrame --
            a data frame with columns labeled 'book_name' and 'book_rating' containing read books (i.e., titles of books this BookLover has read and
            this BookLover's ratings of those books on a scale of 1 to 5, where 1 means this BookLover did not like the book at all, and 5 means this BookLover loved the book)
            (e.g., pd.DataFrame(columns =
                ['book_name', 'book_rating'], data = [
                ['Jane Eyre', 4],
                ['Fight Club', 3],
                ['The Divine Comedy', 5],
                ['The Popol Vuh', 5]]
            ))

    Public methods:
        __init__
    '''

    def __init__(
        self,
        name,
        email,
        fav_genre,
        num_books = 0,
         book_list = pd.DataFrame({'book_name':[], 'book_rating':[]}).astype(dtype = {'book_name': str, 'book_rating': int})
    ):
        '''
        Initializes a BookLover

        Keyword arguments:
            name: str -- a name with which to initialize this BookLover
            email: str --an email address and unique identifier with which to initialize this BookLover
            fav_genre: str -- a favorite genre with which to initialize this BookLover
            num_books: int -- a number of read books with which to initialize this BookLover. Default 0.
            book_list: pd.DataFrame --
                a data frame with which to initialize this BookLover. Has columns labeled 'book_name' and 'book_rating' and contains read books (i.e., titles of books this BookLover
                has read and this BookLover's ratings of those books on a scale of 0 to 5, where 0 means this BookLover did not like the book at all, and 5 means this BookLover loved
                the book). Default pd.DataFrame({'book_name':[], 'book_rating':[]}).

        Return values:
            none

        Side effects:
            Initializes this BookLover's name, email address, favorite genre, number of read books, and read-book list
```

```
59
60          Exceptions raised:
61              none
62
63          Restrictions on when this method can be called:
64              May not be called directly
65          '''
66
67          self.name = name
68          self.email = email
69          self.fav_genre = fav_genre
70          self.num_books = num_books
71          # book_list references an empty data frame of the list __defaults__ of this method of BookLover; this
data frame is a class attribute. self.book_list references a instance attribute.
72          self.book_list = book_list.copy()
73
74      def _get_list_of_book_names(self):
75          self.series_of_book_names = self.book_list['book_name']
76          self.list_of_book_names = self.series_of_book_names.to_list()
77          return self.list_of_book_names
78
79      def add_book(self, book_name, rating):
80          '''
81          Adds a read book to this BookLover's read-book list
82
83          Keyword arguments:
84              book_name: str -- the name of a book to add to this BookLover's read-book list
85              rating: int -- a book rating from 0 to 5 to add to this BookLover's read-book list
86
87          Return values:
88              none
89
90          Side effects:
91              Adds a book to this BookLover's read-book list
92
93          Exceptions raised:
94              BookAlreadyExistsException if a book already exists in this BookLover's read-book list
95
96          Restrictions on when this method can be called:
97              none
98          '''
99
100         if (book_name in self._get_list_of_book_names()):
101             raise BookAlreadyExistsInBookListException(book_name + ' already exists in the read-book list of
' + self.name + '.')
102         else:
103             # Raf Alvarado's solution:
104             #new_book = pd.DataFrame({
105             #    'book_name': [book_name],
106             #    'book_rating': [rating]
107             #})
108             #self.book_list = pd.concat([self.book_list, new_book], ignore_index = True)
109             self.book_list.loc[len(self.book_list.index)] = [book_name, rating]
110             self.num_books += 1
111
112     def has_read(self, book_name):
113         '''
114         Returns an indicator of whether this BookLover has read a book with a specified name
115
116         Keyword arguments:
117             book_name: str -- the name of a book for which to return an indicator of whether this BookLover
has read that book
118
119         Return values:
120             True if this BookLover has read a book with a specified name
121             False if this BookLover has not read a book with a specified name
```

```
122
123         Side effects:
124             none
125
126         Exceptions raised:
127             none
128
129         Restrictions on when this method can be called:
130             none
131         '''
132
133         if (book_name in self._get_list_of_book_names()):
134             return True
135         else:
136             return False
137
138     def num_books_read(self):
139         '''
140         Returns the number of books this BookLover has read
141
142         Keyword arguments:
143             none
144
145         Return values:
146             the total number of books this BookLover has read
147
148         Side effects:
149             none
150
151         Exceptions raised:
152             none
153
154         Restrictions on when this method can be called:
155             none
156         '''
157
158         return self.num_books
159
160     def fav_books(self):
161         '''
162         Returns a data frame of books to which this BookLover has given ratings greater than 3
163
164         Keyword arguments:
165             none
166
167         Return values:
168             a data frame of books to which this BookLover has given ratings greater than 3
169
170         Side effects:
171             none
172
173         Exceptions raised:
174             none
175
176         Restrictions on when this method can be called:
177             none
178         '''
179
180         return self.book_list[self.book_list['book_rating'] > 3]'''
181 Module for class BookLoverTestSuite, which tests the methods of a BookLover
182 '''
183
184 import unittest
185 from booklover import *
186
187 class BookLoverTestSuite(unittest.TestCase):
```

```
    '''
    Tests the methods of a BookLover

    Instance variables:
        none

    Public methods:
        test_1_init
    '''

    def setUp(self):
        '''
        Creates a instance variable with a value of a BookLover

        Keyword arguments:
            none

        Return values:
            none

        Side effects:
            Creates an instance variable with a value of a BookLover

        Exceptions raised:
            none

        Restrictions on when this method can be called:
            none
        '''

        self.book_lover = BookLover('Han Solo', 'hsolo@millenniumfalcon.com', 'scifi')

    def test_0_init(self):
        '''
        Tests BookLover.__init__

        Keyword arguments:
            none

        Return values:
            none

        Side effects:
            Tests whether a BookLover's name, email, favorite genre, number of read books, and book list are equal to expected values

        Exceptions raised:
            AssertionError if a BookLover's name, email, favorite genre, number of read books, or book list is not equal to an expected value

        Restrictions on when this method can be called:
            none
        '''

        self.assertEqual(self.book_lover.name, 'Han Solo')
        self.assertEqual(self.book_lover.email, 'hsolo@millenniumfalcon.com')
        self.assertEqual(self.book_lover.fav_genre, 'scifi')
        self.assertEqual(self.book_lover.num_books, 0)
            self.assertTrue(self.book_lover.book_list.equals(pd.DataFrame({'book_name':[], 'book_rating':[]}).astype(dtype = {'book_name': 'str', 'book_rating': 'int'})))


    def test_1_add_book(self):
        '''
        Tests BookLover.add_book by confirming that a book was successfully added to a BookLover's read book list
```

```
250
251         Keyword arguments:
252             none
253
254         Return values:
255             none
256
257         Side effects:
258             Adds a book to the read-book list of a BookLover and tests whether the book is in the BookLover's
read-book list
259
260         Exceptions raised:
261             AssertionError if a book, including book name and rating, is not correctly added to the read-book
list of a BookLover
262
263         Restrictions on when this method can be called:
264             none
265         '''
266
267         self.book_lover.add_book('Star Wars: A New Hope', 5)
268          self.assertTrue(self.book_lover.book_list.equals(pd.DataFrame({'book_name':['Star Wars: A New
Hope'], 'book_rating':[5]})))
269
270     def test_2_add_book(self):
271         '''
272         Tests BookLover.add_book by running test_1_add_book twice, catching a BookAlreadyExistsInBook-
ListException thrown when an attempt is made to add a book already in a BookLover's read-book list, and confirm-
ing that only one book was added to the BookLover's read-book list
273
274         Keyword arguments:
275             none
276
277         Return values:
278             none
279
280         Side effects:
281             Runs test_1_add_book twice, catches a BookAlreadyExistsInBookListException thrown when an
attempt is made to add a book already in a BookLover's read book list, and confirms that only one book was added
to the BookLover's read-book list
282
283         Exceptions raised:
284             AssertionError if test_1_add_book succeeds or a BookLover's read-book list does not contain one
book
285
286         Restrictions on when this method can be called:
287             none
288         '''
289
290         self.test_1_add_book()
291         try:
292             self.test_1_add_book()
293             self.fail()
294         except BookAlreadyExistsInBookListException as e:
295             pass
296          self.assertTrue(self.book_lover.book_list.equals(pd.DataFrame({'book_name':['Star Wars: A New
Hope'], 'book_rating':[5]})))
297
298     def test_3_has_read(self):
299         '''
300          Tests BookLover.has_read by ensuring that has_read returns True when the name of a book in a
BookLover's read-book list is provided to has_read
301
302         Keyword arguments:
303             none
304
305         Return values:
```

```
306             none
307
308         Side effects:
309             Ensures that has_read returns True when the name of a book in a BookLover's read-book list is pro-
vided to has_read
310
311         Exceptions raised:
312             AssertionError if has_read returns False when the name of a book in a BookLover's read-book list is
provided to has_read
313
314         Restrictions on when this method can be called:
315             none
316         '''
317
318         self.test_1_add_book()
319         self.assertTrue(self.book_lover.has_read('Star Wars: A New Hope'))
320
321     def test_4_has_read(self):
322         '''
323         Tests BookLover.has_read by ensuring that has_read returns False when the name of a book not in a
BookLover's read-book list is provided to has_read
324
325         Keyword arguments:
326             none
327
328         Return values:
329             none
330
331         Side effects:
332             Ensures that has_read returns False when the name of a book not in a BookLover's read-book list is
provided to has_read
333
334         Exceptions raised:
335             AssertionError if has_read returns True when the name of a book not in a BookLover's read-book
list is provided to has_read
336
337         Restrictions on when this method can be called:
338             none
339         '''
340
341         self.test_1_add_book()
342         self.assertFalse(self.book_lover.has_read('Star Wars: Empire Strikes Back'))
343
344     def test_5_num_books_read(self):
345         '''
346         Tests BookLover.num_books_read by ensuring that num_books_read returns 2 when a BookLover's
read-book list contains two books
347
348         Keyword arguments:
349             none
350
351         Return values:
352             none
353
354         Side effects:
355             Ensures that num_books_read returns 2 when a BookLover's read-book list contains two books
356
357         Exceptions raised:
358             AssertionError if num_books_read returns a number other than 2 when a BookLover's read-book
list contains two books
359
360         Restrictions on when this method can be called:
361             none
362         '''
363
364         self.test_1_add_book()
```

```
        self.book_lover.add_book('Star Wars: Empire Strikes Back', 4)
        self.assertEqual(self.book_lover.num_books_read(), 2)

    def test_6_fav_books(self):
        '''
        Tests BookLover.fav_books by ensuring that fav_books returns a data frame of two books with ratings greater than 3 when a BookLover's read-book list contains two books with rating greater than 3 and one book with rating 3

        Keyword arguments:
            none

        Return values:
            none

        Side effects:
            Ensures that fav_books returns a data frame of two books when a BookLover's read-book list contains two books with rating greater than 3 and one book with rating 3

        Exceptions raised:
            AssertionError if fav_book does not return a data frame of two books with ratings greater than 3
            when a BookLover's read-book list contains two books with rating greater than 3 and one book with rating 3

        Restrictions on when this method can be called:
            none
        '''

        self.test_1_add_book()
        self.book_lover.add_book('Star Wars: Empire Strikes Back', 4)
        self.book_lover.add_book('Star Wars: Return of the Jedi', 3)
        data_frame_of_favorite_books = self.book_lover.fav_books()
        self.assertTrue(
            data_frame_of_favorite_books.equals(
                pd.DataFrame(
                    {
                        'book_name': [
                            'Star Wars: A New Hope',
                            'Star Wars: Empire Strikes Back'
                        ],
                        'book_rating': [
                            5,
                            4
                        ]
                    }
                )
            )
        )
        series_of_indicators_of_whether_ratings_are_greater_than_3 = (data_frame_of_favorite_books['book_rating'] > 3)
        self.assertTrue(series_of_indicators_of_whether_ratings_are_greater_than_3.all())

    def tearDown(self):
        '''
        Deletes an instance variable with a value of a BookLover

        Keyword arguments:
            none

        Return values:
            none

        Side effects:
            Deletes an instance variable with a value of a BookLover

        Exceptions raised:
```

```
            none

        Restrictions on when this method can be called:
            none
        '''

        del self.book_lover

if __name__ == '__main__':
    verbose = 3
    unittest.main(verbosity = verbose)test_0_init (__main__.BookLoverTestSuite) ... ok
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 7 tests in 0.046s

OK
```