```
#DS5100 Module 08 Homework
#ID : VVK6RD
#Name: Ade Faparusi

import pandas as pd

class BookLover():
    """
    Just another Python class.
    """


    def __init__(self, name, email, fav_genre):
        self.name = name
        self.email = email
        self.fav_genre = fav_genre
        self.num_books = 0
        self.book_list = pd.DataFrame({'book_name':[], 'book_rating':[]})


    def add_book(self, book_name, rating):
        self.book_name = book_name
        self.rating = rating

        if (len(self.book_list) > 0)  and (self.book_name in self.book_list.book_name.values):
            print(self.book_name, " already on book list")
            return False
        else:
            new_book = pd.DataFrame({
                'book_name': [self.book_name],
                'book_rating': [self.rating]
            })
            self.book_list = pd.concat([self.book_list, new_book], ignore_index=True)


    def has_read(self, book_name):
        self.book_name = book_name
        if self.book_name in self.book_list.book_name.values:
            print ("You have read the book: ", self.book_name)
            return True
        else:
            print ("You have not read the book: ", self.book_name)
            return False


    def num_books_read(self):
        actual = len(self.book_list)
        print ('num_books_read: ', actual)
        return actual

    def fav_books(self):
        fav_list = [num for num in self.book_list[self.book_list.book_rating>3].book_name.values]
        print ("List of favorite books : ",fav_list)
        return fav_list


if __name__ == '__main__':

    test_object = BookLover("Han Solo", "hsolo@millenniumfalcon.com", "scifi")
    test_object.add_book("War of the Worlds", 4)
    test_object.add_book("War of the Worlds", 4)
    test_object.add_book("Jane Eyre", 4)
    test_object.add_book("Fight Club", 3)
    test_object.add_book("The Divine Comedy", 5)
    test_object.add_book("The Popol Vuh", 5)
    test_object.has_read('The Popol Vuh')
```

```
    test_object.has_read('The Roll')
    test_object.num_books_read()
    test_object.fav_books()
```

```
#DS5100 Module 08 Homework
#ID : VVK6RD
#Name: Ade Faparusi


import unittest
from booklover import BookLover


class BookLoverTestSuite(unittest.TestCase):

    def test_1_add_book(self):
        # add a book and test if it is in 'book_list'.
        booklover1 = BookLover("Han Solo", "hsolo@millenniumfalcon.com", "scifi")
        test_book_name = "Oliver Twist"
        test_rating = 4
        booklover1.add_book(test_book_name , test_rating)
        self.assertTrue(booklover1.has_read(test_book_name))


    def test_2_add_book(self):
        # add the same book twice. Test if it's in 'book_list' only once.
        booklover1 = BookLover("Han Solo", "hsolo@millenniumfalcon.com", "scifi")
        test_book_name = "Oliver Twist"
        test_rating = 4
        booklover1.add_book(test_book_name , test_rating)
        booklover1.add_book(test_book_name , test_rating)
        count_book = len (booklover1.book_list[booklover1.book_list.book_name ==test_book_name])
        self.assertEqual(count_book, 1)

    def test_3_has_read(self):
        # pass a book in the list and test if the answer is 'True'.
        booklover1 = BookLover("Han Solo", "hsolo@millenniumfalcon.com", "scifi")
        test_book_name = "Oliver Twist"
        test_rating = 4
        booklover1.add_book(test_book_name , test_rating)
        self.assertTrue(booklover1.has_read(test_book_name))

    def test_4_has_read(self):
        # pass a book NOT in the list and use 'assert False' to test the answer is 'True'
        booklover1 = BookLover("Han Solo", "hsolo@millenniumfalcon.com", "scifi")
        test_book_name = "Oliver Twist"
        test_rating = 4
        booklover1.add_book(test_book_name , test_rating)
        test_book_name2 = "Monte Cristo"
        self.assertFalse(booklover1.has_read(test_book_name2))

    def test_5_num_books_read(self):
        # add some books to the list, and test num_books matches expected.
        booklover1 = BookLover("Han Solo", "hsolo@millenniumfalcon.com", "scifi")
        add_books = [("Oliver Twist",4),
                ("Jane Eyre", 4),
                ("Fight Club", 3),
```

```
                    ("The Divine Comedy", 5),
                    ("The Popol Vuh", 5)
                   ]
        for book_info in add_books:
            booklover1.add_book(*book_info)

        self.assertEqual(booklover1.num_books_read(),len(add_books))


    def test_6_fav_books(self):
        # add some books with ratings to the list, making sure some of them have rating > 3.
        # Your test should check that the returned books have rating  > 3
        booklover1 = BookLover("Han Solo", "hsolo@millenniumfalcon.com", "scifi")
        add_books = [("Oliver Twist",4),
                    ("Jane Eyre", 4),
                    ("Fight Club", 3),
                    ("The Divine Comedy", 5),
                    ("The Popol Vuh", 5)
                   ]
        for book_info in add_books:
            booklover1.add_book(*book_info)

        self.assertEqual(len(booklover1.fav_books()),len([a for a,b in add_books if b>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 0.030s

OK
```

```
#DS5100 Module 08 Homework
#ID : VVK6RD
#Name: Ade Faparusi

import pandas as pd

class BookLover():
    """
    Just another Python class.
    """


    def __init__(self, name, email, fav_genre):
        self.name = name
        self.email = email
        self.fav_genre = fav_genre
        self.num_books = 0
        self.book_list = pd.DataFrame({'book_name':[], 'book_rating':[]})


    def add_book(self, book_name, rating):
        self.book_name = book_name
        self.rating = rating

        if (len(self.book_list) > 0)  and (self.book_name in self.book_list.book_name.values):
            print(self.book_name, " already on book list")
            return False
        else:
            new_book = pd.DataFrame({
                'book_name': [self.book_name],
                'book_rating': [self.rating]
            })
            self.book_list = pd.concat([self.book_list, new_book], ignore_index=True)


    def has_read(self, book_name):
        self.book_name = book_name
        if self.book_name in self.book_list.book_name.values:
            print ("You have read the book: ", self.book_name)
            return True
        else:
            print ("You have not read the book: ", self.book_name)
            return False


    def num_books_read(self):
        actual = len(self.book_list)
        print ('num_books_read: ', actual)
        return actual

    def fav_books(self):
        fav_list = [num for num in self.book_list[self.book_list.book_rating>3].book_name.values]
        print ("List of favorite books : ",fav_list)
        return fav_list


if __name__ == '__main__':

    test_object = BookLover("Han Solo", "hsolo@millenniumfalcon.com", "scifi")
    test_object.add_book("War of the Worlds", 4)
    test_object.add_book("War of the Worlds", 4)
    test_object.add_book("Jane Eyre", 4)
    test_object.add_book("Fight Club", 3)
    test_object.add_book("The Divine Comedy", 5)
    test_object.add_book("The Popol Vuh", 5)
    test_object.has_read('The Popol Vuh')
```

```
    test_object.has_read('The Roll')
    test_object.num_books_read()
    test_object.fav_books()
```

```
#DS5100 Module 08 Homework
#ID : VVK6RD
#Name: Ade Faparusi


import unittest
from booklover import BookLover


class BookLoverTestSuite(unittest.TestCase):

    def test_1_add_book(self):
        # add a book and test if it is in 'book_list'.
        booklover1 = BookLover("Han Solo", "hsolo@millenniumfalcon.com", "scifi")
        test_book_name = "Oliver Twist"
        test_rating = 4
        booklover1.add_book(test_book_name , test_rating)
        self.assertTrue(booklover1.has_read(test_book_name))


    def test_2_add_book(self):
        # add the same book twice. Test if it's in 'book_list' only once.
        booklover1 = BookLover("Han Solo", "hsolo@millenniumfalcon.com", "scifi")
        test_book_name = "Oliver Twist"
        test_rating = 4
        booklover1.add_book(test_book_name , test_rating)
        booklover1.add_book(test_book_name , test_rating)
        count_book = len (booklover1.book_list[booklover1.book_list.book_name ==test_book_name])
        self.assertEqual(count_book, 1)

    def test_3_has_read(self):
        # pass a book in the list and test if the answer is 'True'.
        booklover1 = BookLover("Han Solo", "hsolo@millenniumfalcon.com", "scifi")
        test_book_name = "Oliver Twist"
        test_rating = 4
        booklover1.add_book(test_book_name , test_rating)
        self.assertTrue(booklover1.has_read(test_book_name))

    def test_4_has_read(self):
        # pass a book NOT in the list and use 'assert False' to test the answer is 'True'
        booklover1 = BookLover("Han Solo", "hsolo@millenniumfalcon.com", "scifi")
        test_book_name = "Oliver Twist"
        test_rating = 4
        booklover1.add_book(test_book_name , test_rating)
        test_book_name2 = "Monte Cristo"
        self.assertFalse(booklover1.has_read(test_book_name2))

    def test_5_num_books_read(self):
        # add some books to the list, and test num_books matches expected.
        booklover1 = BookLover("Han Solo", "hsolo@millenniumfalcon.com", "scifi")
        add_books = [("Oliver Twist",4),
                ("Jane Eyre", 4),
                ("Fight Club", 3),
```

```
                    ("The Divine Comedy", 5),
                    ("The Popol Vuh", 5)
                   ]
        for book_info in add_books:
            booklover1.add_book(*book_info)

        self.assertEqual(booklover1.num_books_read(),len(add_books))


    def test_6_fav_books(self):
        # add some books with ratings to the list, making sure some of them have rating > 3.
        # Your test should check that the returned books have rating  > 3
        booklover1 = BookLover("Han Solo", "hsolo@millenniumfalcon.com", "scifi")
        add_books = [("Oliver Twist",4),
                    ("Jane Eyre", 4),
                    ("Fight Club", 3),
                    ("The Divine Comedy", 5),
                    ("The Popol Vuh", 5)
                   ]
        for book_info in add_books:
            booklover1.add_book(*book_info)

        self.assertEqual(len(booklover1.fav_books()),len([a for a,b in add_books if b>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 0.030s

OK
```