Metadata Course: DS 5100 Module: 08 Python Testing Topic: HW Unit Testing a Book Lover Class Author: R.C. Alvarado (adapted) Date: 19 March 2024

* Name: Alanna Hazlett * Net UD: uwa6xv * URL of this file in GitHub: https://github.com/AlannaHazlett/DS5100--uwa6xv-/tree/main/lessons/M08

```
import pandas as pd
import numpy as np
class BookLover():
    '''Stores data about books users have read.'''



    num_books = 0
    book_list = pd.DataFrame({'book_name':[], 'book_rating':[]})

    def __init__(self,name,email,fav_genre):
        self.name = name
        self.email = email
        self.fav_genre = fav_genre


    def add_book(self,book_name,book_rating):
        # Check if value book_name exists in any rows of any columns
        if self.book_list.isin([book_name]).any().any():
            print("Book already exists in the DataFrame")
        else:
            self.num_books += 1
            new_book = pd.DataFrame({
            'book_name': [book_name],
            'book_rating': [book_rating]
            })
            self.book_list = pd.concat([self.book_list, new_book], ignore_index=True)


    def has_read(self,book_name):
        #The method should return True if the person has read the book, False otherwise.
        if self.book_list.isin([book_name]).any().any():
            return True
        else:
            return False


    def num_books_read(self):
        #return self.book_list.shape[0]
        return self.num_books


    def fav_books(self):
        return self.book_list[self.book_list.book_rating > 3]


import unittest
from booklover import BookLover

class BookLoverTestSuite(unittest.TestCase):


    def test_1_add_book(self):
        # Create instance
        test_object = BookLover("Han Solo", "hsolo@millenniumfalcon.com", "scifi")
        # add a book and test if it is in `book_list`.
        test_object.add_book("War of the Worlds", 4)
        self.assertEqual(1,len(test_object.book_list))


    def test_2_add_book(self):
        # Create instance
        test_object = BookLover("Han Solo", "hsolo@millenniumfalcon.com", "scifi")
        # add the same book twice. Test if it’s in ‘book_list‘ only once.
        test_object.add_book("War of the Worlds", 4)
        expected = len(test_object.book_list)
        test_object.add_book("War of the Worlds", 4)
        actual = len(test_object.book_list)
        self.assertEqual(actual, expected)


    def test_3_has_read(self):
        # Create instance
        test_object = BookLover("Han Solo", "hsolo@millenniumfalcon.com", "scifi")
        # pass a book in the list and test if the answer is ‘True‘.
        test_object.add_book("War of the Worlds", 4)
        self.assertTrue(test_object.has_read("War of the Worlds"))


    def test_4_has_read(self):
        # Create instance
        test_object = BookLover("Han Solo", "hsolo@millenniumfalcon.com", "scifi")
        # pass a book NOT in the list and use ‘assert False‘ to test the answer is ‘True‘
        test_object.add_book("War of the Worlds", 4)
        self.assertFalse(test_object.has_read("Barbie"))


    def test_5_num_books_read(self):
        # Create instance
        test_object = BookLover("Han Solo", "hsolo@millenniumfalcon.com", "scifi")
        # add some books to the list, and test num_books matches expected.
        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)
        # Give expected value
        books_in_list = 4
        # Compare expected value with num_books
        self.assertEqual(books_in_list,test_object.num_books)


    def test_6_fav_books(self):
        # Create instance
        test_object = BookLover("Han Solo", "hsolo@millenniumfalcon.com", "scifi")
        # add some books with ratings to the list, making sure some of them have rating > 3.
        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)
        # Your test should check that the returned books have rating  > 3
        expected_value2 = 3
        self.assertEqual(expected_value2, len(test_object.fav_books()))


if __name__ == ’__main__’:

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

----------------------------------------------------------------------
```

```
Ran 6 tests in 0.013s

OK
```