Assignment 5 Part I

CS 2000: Python Programming Language

Wassnaa AL-mawee

Western Michigan University

March 20, 2018

In this assignment, you will work with Python's built-in unittest framework to perform Python unit testing. It is worth 5 points and is due (03/28/2018).

Given an accounting routine used in a bookshop. It works on a list with sublists, which look like this:

orders= [bookshop $order_1$, [$book_1$ order number, quantity, price per item], [$book_2$ order number, quantity, price per item], ...[..,..,...]], ...[bookshop $order_n$, [...,..,...], [,...,...,..]]

```
>>> orders = [ [1, ("5464", 4, 9.99), ("8274",18,12.99), ("9744", 9, 44.95)],

              [2, ("5464", 9, 9.99), ("9744", 9, 44.95)],

              [3, ("5464", 9, 9.99), ("88112", 11, 24.99)],

              [4, ("8732", 7, 11.99), ("7733", 11,18.99), ("88112", 5, 39.95)] ]
```

In this assignment, you will implement two classes: Bookshop class contains all the necessary user's defined methods for orders list operations (using lambda, map, filter, and reduce), and TestBookshop class, that inherits from unittest.TestCase, includes all the necessary unit tests to test all the defined methods in Bookshop class. All the test cases must be passed or ok.

1. (0.5 point) Write a method, which returns the same given list, but the tuples have 2-items only. Each tuple consists of the order number and the product of the price per items and the quantity. The product should be increased by $10 if the value of the order is less than $100.
   Implement a unit test test1(), using assertEqual() to check if (actual value == expected value).
   Print out the actual value and the expected value in test1.

2. (0.5 point) Write a method, which filters out the minimum price of product (price per item *quantity) per bookshop order. It returns a list with 4-tuples. Each tuple has 2 items (bookshop order number, book order number).
   Implement a unit test test2(), using assertEqual() to check if (actual value == expected value).
   Print out the actual value and the expected value in test2.

3. (0.5 point) Write a method, which filters out the maximum price of product (price per item* quantity) per bookshop order. It returns a list with 4-tuples. Each tuple has 2 items (bookshop order number, book order number).
   Implement a unit test `test3()`, using `assertNotEqual()` to check if (actual value != wrong expected value).
   Print out the actual value and the expected value in test3.

4. (0.5 point) Write a method, which returns a list with tuples. Each tuple has two items: (bookshop order number, total amount of order).
   Implement a unit test `test4()`, using `assertEqual()` to check if (actual value == expected value).
   Print out the actual value and the expected value in test4.

5. (0.5 point) Write a method, which returns a list with two items [book order number, total amount of the product in all orders]. This method returns the book order number that has the maximum total amount of product in all orders.
   Implement a unit test `test5()`, using `assertIn()` to check if (a book order number in the returned list).
   Print out the book order number and the returned list in test5.

6. (0.5 point) Write a method, which returns a list with two items [book order number, total number of its quantity in all orders]. This method returns the book order number that has the maximum total number of quantity in all orders
   Implement a unit test `test6()`, using `assertEqual()` to check if (actual value == expected value).
   Print out the actual value and the expected value in test6.

7. (0.5 point) Write a method, which returns an ordered list based on (bookshop order number) per maximum total quantity. [(max bookshop order number, total quantity), …..(min bookshop order number, total quantity)]
   Implement a unit test `test7()`, using `assertEqual()` to check if (actual value == expected value).
   Print out the actual value and the expected value in test7.

8. (0.5 point) Write a method, which returns a total quantity of all books that have been ordered.
   Implement a unit test `test8()`, using `assertEqual()` to check if (actual value == expected value).
   Print out the actual value and the expected value in test8.

9. (0.5 point) Write a method, which returns a list with two items [the most ordered (book order number), and the least ordered (book order number]. To figure out that, you need to count the occurrence of book order number in all orders.
   Implement a unit test `test9()`, using `assertEqual()` to check if (actual value == expected value).
   Print out the actual value and the expected value in test9.

10. (0.5 point) Write a method, which returns a list with 4 items. Each item represents the length of the sublists from index 0 to index 3.
    Implement a unit test `test10()`, using `assertEqual()` to check if (actual value == expected value).
    Print out the actual value and the expected value in test10.

- **What to turn in:**

(Remember READABILITY COUNTS!)

- On e-Learning, submit your Python file whose name <hw#5PartI_LastName.py>. Please format your Python code in the following manner:

Name: <your name here>

Date: <#/#/#>

Homework: <#>

Your code