A Simple Grocery Store Program

This assignment description is copied from the COMPSCI 101 assignment. I've attached the pdf for that assignment specification which may be helpful to read.

For this program, you will be tasked to complete the implementation for a **simple grocery store program** written in python. The program is a console application which supports tasks such as:

- Viewing items on sale
- Add/remove items to/from shopping cart
- Checkout cart items

You are provided with the following python files: Files which require completion have **TODO** comments in the python file and are also mentioned below.

- **GroceryStore.py** - Grocery store file which contain information about the available products. **TODO**
- **GroceryStoreMain.py** - Logic to initialise and startup the program. This file already takes care of control flow of the program. **TODO**
- **Product.py** - Product file which contain information about an item sold at the GroceryStore. **TODO**
- **User.py** - User file which allows products/items to be added/removed from cart. **TODO**
- **TestGroceryStore.py** - Use this test file to verify the correctness of your implemention.

Currently, the program will not function and requires completion. Before beginning, it is advised to understand all the files and how they interact. The methods have been commented to describe their purpose and the requirements of the tasks are described below.

Installation

The software requires `python3` to run. If you haven't installed python3 yet, follow the below instructions to get started.

Mac

Instructions (http://python-guide-pt-br.readthedocs.io/en/latest/starting/install3/osx/)

Windows

Download site (https://www.python.org/downloads/)

Running the software

All commands are executed on the same directory as the python files in a terminal/command line.

Grocery store

Run the program using `python3 GroceryStoreMain.py` After executing the above command, the program should display back the top-level menu:

```
Hello Chang Kon Han
Welcome to OrionStore
1. Show the list of items on sale
2. Start to shop online
3. Exit the system
Enter selection:
```

You can interact with the program through keyboard input. The program will detect invalid inputs and continue asking for correct input.

```
Enter selection: 4
Invalid option, please try again!
```

The overall control flow of the program can be outlined as follows:

- Option one of the top-level menu leads to displaying all the items on sale and then returning to the same top-level menu.
- Option two of the top-level menu leads to the second-level menu;
- Option three of the top-level menu leads to exiting from the program;
- Option one or two of the second-level menu leads to adding or removing an items to or from the shopping cart and then returning to the same second-level menu;
- Option 3 of the second-level menu returns back to the top-level menu after a successful checkout. Otherwise, the display remains at the second-level menu.
- Option 4 of the second-level menu leads to discarding the shopping cart and then returning back to the top-level menu.

```
Hello Chang Kon Han
Welcome to OrionStore
1. Show the list of items on sale
2. Start to shop online
3. Exit the system
Enter selection: 2
1. Add an item to the shopping cart
2. Delete an item from the shopping cart
3. Check out the shopping cart
4. Exit without buying
Enter selection: 1
Enter product id to buy: 1
```

```
Enter quantity to purchase: 1
1 Fresh toast bread white is added to the shopping cart
1. Add an item to the shopping cart
2. Delete an item from the shopping cart
3. Check out the shopping cart
4. Exit without buying
Enter selection: 3
Checking out items ...
Fresh toast bread white/1/$3.99
Total amount due: $3.99
1. Show the list of items on sale
2. Start to shop online
3. Exit the system
Enter selection: 3
Thank you for shopping with us!
```

Unit Tests

Run the unit tests by `python3 -m unittest TestGroceryStore.py -v`

All unit tests should pass for correct implementation of the program.

```
test_add_to_cart (TestGroceryStore.TestGroceryStore) ... ok
test_add_to_cart_cannot_add_more_than_available_stock (TestGroceryStore.TestGrocery
Store) ... ok
test_add_to_cart_cannot_add_out_of_stock_product (TestGroceryStore.TestGroceryStore
) ... ok
test_checkout (TestGroceryStore.TestGroceryStore) ... ok
test_checkout_is_insufficient (TestGroceryStore.TestGroceryStore) ... ok
test_checkout_prints_bill_on_empty_cart (TestGroceryStore.TestGroceryStore) ... ok
test_clear (TestGroceryStore.TestGroceryStore) ... ok
test_clear_on_empty_cart (TestGroceryStore.TestGroceryStore) ... ok
test_grocery_store_print_initial_products (TestGroceryStore.TestGroceryStore) ... o
k
test_grocery_store_print_products_does_not_print_out_of_stock_products (TestGrocery
Store.TestGroceryStore) ... ok
test_product_print (TestGroceryStore.TestGroceryStore) ... ok
test_remove_from_cart (TestGroceryStore.TestGroceryStore) ... ok
test_remove_from_cart_cannot_more_than_current_cart (TestGroceryStore.TestGrocerySt
ore) ... ok
test_remove_from_cart_cannot_remove_product_not_in_cart (TestGroceryStore.TestGroce
ryStore) ... ok
test_user_name (TestGroceryStore.TestGroceryStore) ... ok

----------------------------------------------------------------------
Ran 15 tests in 0.001s

OK
```

Failed tests should show a message.

Tasks

Changing user name

Change the `GroceryStoreMain.py` file and edit `user_name` variable inside the `initialise_user` method. Change it to your name `May Lin Tye`. After changing variable, the display should show at the start.

```
Hello May Lin Tye
Welcome to OrionStore
1. Show the list of items on sale
2. Start to shop online
3. Exit the system
Enter selection:
```

Print product information

Update the `print` method inside `Product.py`. It should print i.e. display into system output the properties of the product, comma-separated. It should display in the order: id, name, price, quantity. The price is prefixed by the `$` symbol. An example output is:

```
1, Fresh toast bread white, $3.99, 20
```

Print grocery store stock

Update the `print_available_products` method inside `GroceryStore.py`. It should display **currently available products i.e. quantity is 1 or greater**. Each product should show its product information. The products are sorted by its product id. Example:

```
1, Fresh toast bread white, $3.99, 20
2, Low-fat milk, $4.8, 10
3, V-energy drink, $2.75, 10
4, Fresh garlic, $1.98, 5
5, Pineapple, $3.6, 6
6, Mango, $1.89, 4
7, Snickers chocolate bar, $1.8, 20
8, Broccoli, $1.47, 11
9, Washed Potato, $2.98, 7
10, Good-morning cereal, $5.6, 10
11, Rose apple, $4.98, 5
12, Avocado, $4.99, 5
13, Bananas, $2.96, 4
14, Kiwi fruit green, $2.45, 10
15, Rock melon, $7.98, 2
16, Lettuce, $2.99, 12
17, Chocolate block, $3.59, 10
```

Add to cart

Edit the `add_to_cart` method inside `User.py`. The method takes in product and quantity argument. Successful addition to the cart should update the cart dictionary variable. **Successful addition of a product into shopping cart should use the product as the key and its quantity as the value**. The method should handle the scenarios where:

1. Can't add out of stock products
2. Can't add more than what's available

If the current product is out of stock. The software should display a message indicating that the **product id** is out of stock.

```
Enter product id to buy: 1
Enter quantity to purchase: 1
Sorry, item 1 is out of stock. Please select a different item
```

If the user tries to add more than what's currently available, the program should also display a message.

```
Enter product id to buy: 1
Enter quantity to purchase: 25
Cannot add more than what is available in stock
```

Successful addition should display the number of products successfully added to the cart. This should update the quantity of the product in the cart variable. Furthermore, the product quantity should be subtracted by the quantity amount.

```
Enter product id to buy: 1
Enter quantity to purchase: 1
1 Fresh toast bread white is added to the shopping cart
```

Remove from cart

Edit the `remove_from_cart` method inside `User.py`. The method takes in product and quantity argument. Successful removal from the cart should update the cart dictionary variable. Furthermore, the product quantity value should be increased after successful removal from cart. The method should handle the scenarios where:

1. Cannot remove more than what's currently in the cart
2. Cannot remove a product which is not in the cart

If the user attempts to remove a product with a quantity value more than what is currently in the cart, the software should display a message.

```
Enter product id to remove: 1
Enter quantity to remove: 3
Cannot remove more than what is in the cart
```

If the user tries to remove a product which is not in the cart, the software should display a message.

```
Enter product id to remove: 5
Enter quantity to remove: 1
This item is not in the shopping cart
```

Successful removal from the cart should display the quantity of the product removed from the cart.

```
Enter product id to remove: 1
Enter quantity to remove: 1
1 Fresh toast bread white is removed from the shopping cart
```

Clear the cart

Edit the `clear` method inside `User.py`. This method clears the shopping cart and displays a message to the user. Clearing a cart works even if the shopping cart is already empty. Clearing the cart should restore the product quantity.

```
All items are cleared from the shopping cart
```

Checkout

Edit the `checkout` method inside `User.py`. Checkout attempts to checkout the shopping cart. Checkout should return a **boolean value** i.e. True or False. Checkout should handle these scenarios:

1. Attempting to checkout an empty cart
2. Insufficient funds to purchase items
3. Successful checkout

If the user attempts to checkout an empty cart, the program should display a message and return False.

```
Cannot checkout empty cart
```

If the user does not have sufficient funds to purchase what's currently in the cart, a message will be shown to the user and the method should return False. It displays the amount required to purchase the money and the current user balance. The money is prefixed with the `$` symbol and rounded to 2 d.p.

```
Insufficient funds to purchase item(s)
$84.37 required to purchase but current balance is $50
```

If there is a successful checkout, a bill should be displayed to the user and the total amount due. The method returns True. The amount is prefixed with the `$` symbol and rounded to 2 d.p.

The bill is formatted to show its name, quantity purchased and the price. It is `/` i.e. forward slash separated. The total amount is summed and displayed at the bottom and subtracted from the users current `balance` variable. Furthermore, the bill is sorted by its product id.

```
Checking out items ...
Fresh toast bread white/3/$11.97
Mango/1/$1.89
Total amount due: $13.86
```