

Description

The main purpose of the project was to design and implement an inventory management system of types of items to handle. For my case, I was assigned to handle the items of a Restaurant inventory(food, beverages etc). One of the main components of the project was the usage of Object Oriented Programming through Classes and Objects, as well as the implementation of key principles such as Encapsulation, Inheritance etc. The 3 main classes of my program were “Item”, “Inventory” and “Report”, with each one being responsible for a certain part of the system. The item types which I worked with (food, beverages) were handled primarily by the Item Class(responsible for setting the category of an item, or getting its name) and the Inventory Class(responsible for adding, removing things from Inventory). The Report class was also used for some other functions.

Instructions

The program contains 3 classes; Inventory, Item and Report. Each class has a specific function within the program, and some inherit from some others. There are also categories which the program uses, such as Foods, Beverages etc. You can take a look at these by using the ‘list_items’ method from the Inventory class. To use the Inventory Class, you must create an object, and then you can access any of the 4 methods from the Inventory class. Each method has a specified number of arguments which you must pass. For instance, the add_item and remove_item methods have 2, the update_inventory has 1 etc. Moving on, the Item class has a very similar logic, with the addition of class attributes, all of whom are private. The number of arguments is specified for each method. The same logic is also applied for the Report Class.

Structure

Class Inventory; no class attributes

add_item method:

2 arguments; the item which needs to be added and the category to be added.

Remove_item method:

2 arguments; the item which needs to be removed and the category from which it is removed from

List_items method:

No arguments; prints the inventory items

Class Item

2 class attributes; the name of the item and the category

Set_category method

No arguments; prints the category of the object which is created from the Item class

Get_details method

No arguments; prints the name and the category of the object created from the Item class

Class Report; no class attributes

Items_in method;

1 argument, the category which you want to check the items for; prints the number of items in a category

Recipes method;

No arguments; prints a list of the recipes in the inventory

Create_new_recipe method;

2 arguments, the name and items of the recipes(must be passed as a list!!) prints the recipe created

Brief Summary on Developed Classes:

Inventory Class: This is the class which is responsible for adding, removing and listing the items of the item system. It has a designated method for each of the functions listed above, and it utilizes categories of foods and beverages. In addition, it allows one to create an entirely new category via the 'update_inventory' method.

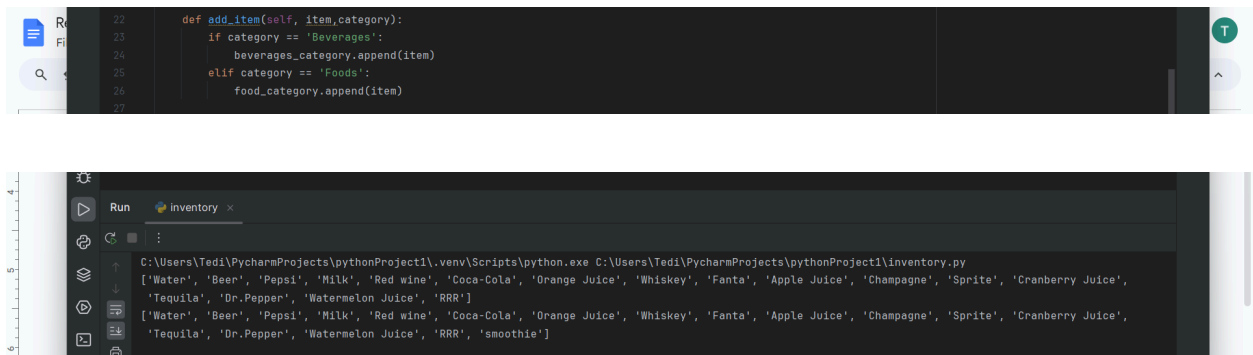
Item Class: The item class has 3 class attributes; name, category and expiration, all of whom are private variables which can't be modified by a client using the program. This class has more to do with individual items, with methods which set the private attributes of an item. The set_name, set_category and set_expiry do so respectively.

Report Class: The Report Class is responsible for sharing some details about the inventory. For instance, the items_in_method prints the number of items which a category contains. The recipes method contains the recipes of the inventory. NOTE: the create_new_recipe method's 'items' argument must be passed as a list.

Verification of Sanity of Code

Scenario 1: Adding a new item to a category

Use the 'add_item' method under the Inventory class. The method takes 2 arguments; the item which you want to add and the category to which you want to add it to. In the following case, I want to add an item 'smoothie' under the 'Beverages' category. The before and after demonstrates that the item has successfully been added to the category class.



The first screenshot shows the 'add_item' method in the Inventory class. It takes 'self', 'item', and 'category' as arguments. It checks if the category is 'Beverages' and appends the item to the 'beverages_category' list, or if it's 'Foods' and appends it to the 'food_category' list.

```
22 def add_item(self, item, category):
23     if category == 'Beverages':
24         beverages_category.append(item)
25     elif category == 'Foods':
26         food_category.append(item)
27
```

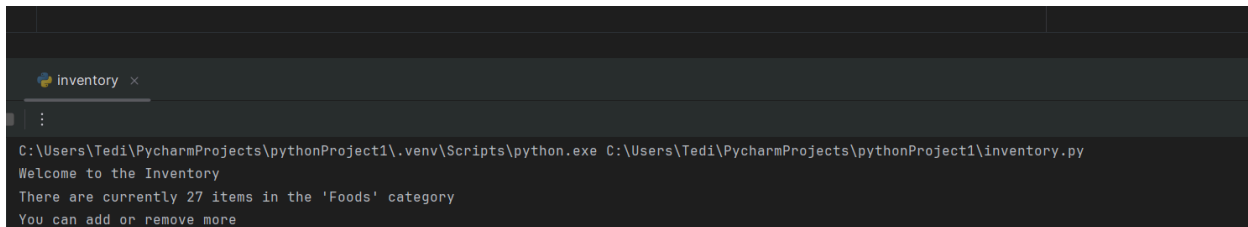
The second screenshot shows the output of the 'add_item' method. It displays the 'beverages_category' list before and after adding the 'smoothie' item. The list before was ['Water', 'Beer', 'Pepsi', 'Milk', 'Red wine', 'Coca-Cola', 'Orange Juice', 'Whiskey', 'Fanta', 'Apple Juice', 'Champagne', 'Sprite', 'Cranberry Juice', 'Tequila', 'Dr.Pepper', 'Watermelon Juice', 'RRR']. The list after is the same, but with 'smoothie' added at the end.

```
Run inventory x
C:\Users\Tedi\PycharmProjects\pythonProject1\venv\Scripts\python.exe C:\Users\Tedi\PycharmProjects\pythonProject1\inventory.py
['Water', 'Beer', 'Pepsi', 'Milk', 'Red wine', 'Coca-Cola', 'Orange Juice', 'Whiskey', 'Fanta', 'Apple Juice', 'Champagne', 'Sprite', 'Cranberry Juice', 'Tequila', 'Dr.Pepper', 'Watermelon Juice', 'RRR']
['Water', 'Beer', 'Pepsi', 'Milk', 'Red wine', 'Coca-Cola', 'Orange Juice', 'Whiskey', 'Fanta', 'Apple Juice', 'Champagne', 'Sprite', 'Cranberry Juice', 'Tequila', 'Dr.Pepper', 'Watermelon Juice', 'RRR', 'smoothie']
```

Scenario 2: Check the number of items in a category

Use the 'itemsIn' method under the Report class. The method takes 1 argument; the category which you want to check the number of items for. In the following case, I want to check the number of items in the 'Beverages' category of my inventory. The printed output demonstrates that the method successfully prints the amount of items in the category. Of course, this can change depending on whether you later decide to add or remove items.

```
def items_in(self, category):
    #prints the number of items in the desired category
    if category == 'Foods':
        lengthOf = len(food_category)
    elif category == 'Beverages':
        lengthOf = len(beverages_category)
    print(f"There are currently {lengthOf} items in the '{category}' category")
    print("You can add or remove more")
```



```
inventory x
C:\Users\Tedi\PycharmProjects\pythonProject1\.venv\Scripts\python.exe C:\Users\Tedi\PycharmProjects\pythonProject1\inventory.py
Welcome to the Inventory
There are currently 27 items in the 'Foods' category
You can add or remove more
```

Conclusion

While working on the project, there were a few challenges which I came across. One of those was determining the type of data structure to use for the storing and manipulation of the items of the Inventory system. I ended up using the list data structure because I believe it is the easiest to use due to the abundance of in-built functions which it contains. For example, the ‘append’ methods allowed me to create the original list of the food and beverages categories. I did this by using a for loop in order to loop through the items of the CSV list and appending each item to the end of the respective lists. However, I think the challenge was overcome and I think the list was the best option. I found while working on the project that the utilization of Object Oriented principles made the project much smoother to work with. For example, encapsulation makes it easy to not have to worry about the name or category of an item being altered by a client who is using the program. To be more specific, I made the name and category attributes private class attributes for 2 reasons; 1) every item has its own class/category 2) once they are created, they can’t be changed. Overall, I feel the project was a good way for us to practice with using OOP and getting more used to knowing how to create classes/objects in order to make our programs more flexible. One area for improvement could be to perhaps use even more classes, and make the program have user input as well.