Polymorphism with Stores

Polymorphism in python defines methods in the child class that have the same name as the methods in the parent class. In inheritance, the child class inherits the methods from the parent class. Also, it is possible to modify a method in a child class that it has inherited from the parent class.

The goal of this program is to create classes that represent to a certain degree different types of stores in real life and the information tracked and services provided by these stores. In order to accomplish this the following classes must to be created:

I) Create a program called store.py which contains an abstract class called Store which is made up of the following:

- **Attributes/Properties**
  - Store name
  - Store address
  - Store availability/status (open or closed)
  - Sales tax percentage
- **Functions/Functionality**
  - Constructor which provides the ability to pass and set the values for the various attributes
  - Getter and setters for all the attributes mentioned above
  - *is_store_open* should return True if the store is open and False if the store is closed
  - Abstract function called: *calculate_total_sales_tax*
  - Abstract function called *calculate_total_sales*

II) Create a program called restaurant.py which contains a class called Restaurant which is type of Store. It should possess all the attributes and functions present in the Store class without having to re-implement those in the Restaurant class. The Restaurant class is made up of the following:

- **Attributes/Properties**
  - Total number of people served
  - Max occupancy
  - Current occupancy
  - Price per person
- **Functions/Functionality**
  - Constructor which provides the ability to pass and set the values for the various attributes
  - *seat_patrons* should do the following:
    - Take the number of people to be seated as input
    - Update the values of the appropriate attributes

        - If number of people to be seated does not exceed or equals the max occupancy
            - Print "Welcome to [replace with name of restaurant]"
            - Return True
        - If number of people to be seated exceeds the max occupancy
            - Print "We are at capacity, we appreciate your patience"
            - Return False
    - *serve_patrons* which should do the following:
        - Take the number of people to serve as input
        - Update the values of the appropriate attributes
        - Return the number of people being served currently
    - *checkout_patrons* (this is when the patrons are ready to leave the restaurant) which should do the following:
        - Take the number of people leaving as input
        - Update the values of the appropriate attributes
        - Return the current occupancy of the restaurant
    - Create a getter and setter for the attribute: Price per person

III) Create a program called grocery_store.py which contains a class called GroceryStore which is type of Store. It should possess all the attributes and functions present in the Store class without having to re-implement those in the GroceryStore class. The GroceryStore class is made up of the following:

- **Attributes/Properties**
    - Total revenue
    - Grocery store type (independent or chain)
- **Functions/Functionality**
    - Constructor which provides the ability to pass and set the values for the various attributes
    - *sell_item* which should do the following:
        - Takes the quantity and price of an item as input
        - Update the values of the appropriate attributes
        - Return the total revenue of the grocery store
    - Create a getter and setter for the attribute: Grocery Store type

IV) Create a program called *shopping.py* which will make use of the above-mentioned Restaurant and GroceryStore classes and call their various functions to simulate the actions which take place in real life restaurants and grocery stores in order to test your code.

V) Create and submit UML diagram.

Do not forget to implement the two abstract functions specified in the Store class within each of the classes mentioned above. As the developer, it is important to determine the best data types to be used for each of the attributes and the input and output for functionality being implemented within this class.

Write a Learning Report Summary (LRS)

Using Microsoft Word, write a summary report (not a bullet items) with a minimum of 100 words explaining how you completed your assignment. *Please describe your responses, not just yes/no answers.*

1. Did you successfully get your assignment done? Did it run? Any error? Did you get the correct result? Did you test your program thoroughly?
2. How much time did you spend to complete your assignment?
3. Did you find the assignment easy or challenging for you?
4. Did you write the program yourself? Did you get any help from anyone?
5. When you encountered obstacles to complete your program, how did you resolve the issues? Did you use Google to get help? Describe how Google was abled or not able to assist you?
6. What did you learn from doing this assignment?
7. Any other information you would like to share with your instructor? Make sure you provide program output on each option.

What to submit

Your program files.
Your program output with sufficient test samples
Your LRS.
Your UML diagram.