project-2

[70 pts] You will be writing code for recording the menu items and daily sales of a lemonade stand. It will have these classes: MenuItem, SalesForDay, and LemonadeStand. All data members of each class should be marked as **private** (a leading underscore in the name). Since they're private, if you need to access them from outside the class, you should do so via get or set methods.

Here are the method descriptions for the three classes:

MenuItem:

A MenuItem object represents a menu item to be offered for sale at the lemonade stand.

* init method - takes as parameters three values with which to initialize the MenuItem: its name, its wholesale cost, and its selling price
* get methods for each of the data members: get_name(), get_wholesale_cost(), and get_selling_price()

SalesForDay:

A SalesForDay object represents the sales for a particular day.

* init method - takes as parameters two values with which to initialize the SalesForDay: the day (an integer for the number of days the stand has been open so far), and a dictionary whose keys are the names of the items sold, and whose values are the numbers of those items sold that day
* get methods for each of the data members: get_day() and get_sales_dict()

LemonadeStand:
Remember that the LemonadeStand class must not directly access the private data members of MenuItem and SalesForDay objects, but instead must call the appropriate get methods

A LemonadeStand object represents a lemonade stand, which has four data members:
* a string for the name of the stand
* an integer representing the current day
* a dictionary of MenuItem objects, where the keys are the names of the items and the values are the corresponding MenuItem objects
* a list of SalesForDay objects

The Lemonade Stand methods are:
* init method - takes as a parameter the name of the stand; initializes the name to that value, initializes current day to zero, initializes the menu to an empty dictionary, and initializes the sales record to an empty list (**remember to not use any mutable default arguments**)
* a get method for the name: get_name()
* add_menu_item - takes as a parameter a MenuItem object and adds it to the menu dictionary
* enter_sales_for_today - takes as a parameter a dictionary where the keys are names of items sold and the corresponding values are how many of the item were sold. If the name of any item sold doesn't match the name of any MenuItem in the menu, it raises an **InvalidSalesItemError** (you'll need to define this exception class). Otherwise, it creates a new SalesForDay object, using the current day and the dictionary that was passed in, adds that object to the list of SalesForDay objects, and then increments the current day by 1
* get_sales_dict_for_day - takes as a parameter an integer representing a particular day, and returns the dictionary of sales for that day (**not** a SalesForDay object)
* total_sales_for_menu_item - takes as a parameter the name of a menu item and returns the total number of that item sold over the history of the stand
* total_profit_for_menu_item - takes as a parameter the name of a menu item and returns the total profit on that item over the history of the stand
* total_profit_for_stand - takes no parameters and returns the total profit on all items sold over the history of the stand

[10 pts] **You must include a main function** that runs if the file is run as a script, but not if the file is imported. The main function should try calling enter_sales_for_today() with a dictionary that contains an item name not in the menu. If an InvalidSalesItem is raised, it should be caught with a try/except that prints an explanatory message for the user (otherwise the function should proceed normally).

[20 pts] In addition to your file containing the code for the above classes, **you must also submit a file that contains unit tests for your classes. It must have at least five unit tests and use at least two different assert functions. This part (like the rest) must be your own work.

Gradescope will not test your main function or unit tests - the TAs will take care of that.

Here's a very simple example of how your classes could be used:
```
stand = LemonadeStand('Lemons R Us')
item1 = MenuItem('lemonade', .5, 1.5)
stand.add_menu_item(item1)
item2 = MenuItem('cookie', .2, 1
stand.add_menu_item(item2)
day0 = {
    'lemonade' : 5,
    'cookie'   : 2
}
stand.enter_sales_for_today(day0)
print(f"lemonade profit = {stand.total_profit_for_menu_item('lemonade')}")

```

Your files must be named: **LemonadeStand.py** and **LemonadeStandTester.py**