Read the instructions carefully. Not following the instructions will result in not getting credit for the assignment.

Make sure your output matches the example run.

Objectives

In Part 8, you will learn how to do the following:

- Implement an informal, implicit protocol to make dessert items comparable
- Sort a list of user-defined objects using the attributes of a class

Structure

Update the following:

- Module name: dessertshop
    - `main()`
- Module name: dessert - DessertItem - Order
- test files as-needed

Problem

Suppose you want to add functionality to your Dessert Shop application to sort the items on your receipt so that the least expensive is at the top and the most expensive is at the bottom. You will need to sort your order (conceptually a list of items) based on a pre-defined data member (attribute) of your class.

For Part 8 we will only consider the price, not other ways we could arrange items.

Changes to DessertItem class

- Implement `__eq__`, `__ne__`, `__lt__`, `__gt__`,`__ge__`, `__le__` operators using the price.

Changes to Order class

Add method:

- `sort(): Order` sort the items by price in ascending order.

Keep in mind that an Order is **not** a list--it is an object of type **Order** that has an attribute that is a list. We can access that attribute as-needed.

Changes to main

- sort the dessert items in an Order
    - after all items are added, but
    - before it is printed

Test Cases

Add pytest test cases to `test_dessert.py` that test all of the relational operators \(=,\lt, \le,\gt, \ge\). Test all operators once.

Add one pytest test case to `test_order.py` that tests the sort method.

Key Program Requirements

1. All relational operators are implemented in DessertItem.
2. All relational operators are tested at least once.
3. The receipt shows items sorted by price in ascending order.
4. Files modified:

- `dessert.py`
- `dessertshop.py`
- `test_order.py`
- `test_dessert.py`

Example Run

```
1: Candy
2: Cookie
3: Ice Cream
4: Sunday
What would you like to add to the order? (1-4, Enter for done): 2
Enter the type of cookie: Oatmeal Raisin
Enter the quantity purchased: 4
Enter the price per dozen: 3.45
```

```
1: Candy
2: Cookie
3: Ice Cream
4: Sunday
What would you like to add to the order? (1-4, Enter for done):
```

Example receipt

Correctness

From your terminal, run `ruff check` on all of your .py source files and test files, like:

```
ruff check dessert.py
```

This will check for syntax errors, violations and many issues that could lead to bugs in your code. Code will be maually graded, so any score received are partial.

Style

From your terminal, run `ruff format` on all of your .py source files to check the format of your code, like:

```
ruff format dessert.py
```

How to Submit

From your GitHub assignment repository page, click Submit and enter a nontrivial commit message.

Grading

This project is manually graded. Use the following rubric.

Criteria	Mastery (100)	Developing (85)	Beginning (70)	Low (50)
Implementation of relational operators	All relational operators are implemented in DessertItem.	The relational operators are implemented but with minor errors in methods or error handling.	The relational operators are implemented but with significant errors in methods or error handling.	The relational operators are either not implemented or incorrectly implemented.
Test Cases	All required test cases are implemented correctly and pass. Each relational operator is tested at least once.	Most required test cases are implemented and pass, but there may be a few missing or failing tests.	Some required test cases are implemented and pass, but there are significant gaps in test coverage.	Few or no required test cases are implemented, or most tests fail.
Receipt Output	The receipt correctly includes payment information for each item. Items appear in ascending order by price.	The receipt includes payment information for most items, but there may be a few errors or omissions. Some items may not be in sorted order.	The receipt includes payment information for some items, but there are significant errors or omissions.	The receipt does not include correct payment information for the items or has serious errors. Sorting was not implemented, or mostly fails.

Students should aim for the Mastery level in all categories to ensure they have fully understood and implemented the concepts covered in the project. The overall project score is the average of the individual criteria scores.