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

Make sure your output matches the example run.

Objectives

- Create a simple database of customers and what they have ordered.

Structure

Create or update the following:

- Module name: dessertshop
  - Customer class
- new file `test_customer.py`

Problem

Suppose you want to add functionality to your Dessert Shop application to keep track of customers and what they have ordered. You will need to begin by creating a Customer class to store all your customer information.

Note: Your instructor may modify this project and the next one to use a SQLAlchemy Object-Oriented database to hold Customer and Order history information.

Customer Class

The Customer Class has the following attributes and methods:

- attribute customer_name: str
- attribute order_history: list[Order]
- attribute customer_id: int
- a constructor that takes the customer's name as its only parameter. Initialize all other object attributes to reasonable values.
- method `add2history(order:Order)-> Customer` add the order to the history and return the Customer object

Add this Customer class to the dessertshop module.

Changes to main

- Add a yes/no loop so that the program asks the user if they want to start another order as soon as the current order is complete and printed. The user should enter "y" and then press "Enter" for yes, and anything else is no.

Test Cases

Add tests to test all attributes in the Customer class. Add these tests to the file `test_customer.py`.

You do not need to create a test for the reading the order history.

Key Program Requirements

1. Create the Customer class
2. pytests for the Customer class have been created
3. Workspace contains all files from Part 9 and
    - new file `test_cutomer.py`
    - `dessertshop.py` is modified

Example Run

There is no Example Run for Part 10. Run your `test_customer.py` to test your Customer class.

Correctness

From your terminal, run `ruff check` on all of your .py source files and tesst 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 above 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 pts)	Developing (85 pts)	Beginning (70 pts)	Low (50 pts)
Implementation of the Customer Class	All attributes defined correctly Constructor initializes with customer's name `add2history` method implemented and returns Customer object	All attributes defined Constructor partially correct `add2history` method present but issues exist	Missing one attribute Minimal methods defined	No or limited implementation of Customer class
Changes to main	Implemented yes/no loop correctly Accepts "y" as yes and terminates for other inputs	Yes/No loop implemented with minor errors Doesn't fully follow the "y" for yes rule	Attempted changes to main, but major errors exist	No changes made to main
Test Case Creation for Customer Class	Tests for all attributes except order history Tests located in `test_customer.py`	Tests for most attributes present Minor issues or incomplete tests	- Few tests present with several major issues	No or minimal tests related to Customer class

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.