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

Objectives

- Use a dictionary to store and retrieve information (alternatively a database)
- Use a class variable that is shared among all instances of a class

Structure

Update the following:

- Module name
  - deesertshop
- Classes
  - DessertShop
  - Customer
- Function
  - main

Problem

Now that you have a Customer class ready for use, add functionality to your DessertShop application to:

- Store a customer list
- Store an order history for each customer

Note: Your instructor may modify Part 11 requirements to use a SQLAlchemy OO database.

Changes to DessertShop class

- Add an attribute `customer_db: Dict[str,Customer]` to your DessertShop class. The key is the customer name and the value is the Customer object.
- For simplicity we assume that customer names are unique for distinct customers.

Note: Usually we would not use a customer's name as a key. We would normally want to use a key that we could ensure would be unique like a customer's id attribute. For simplicity we will assume that every customer of the Dessert Shop has a unique name.

Changes to console application user input

1. After the order is complete (the user hits enter indicating they don't want to add any more items), ask for the customer's name.
2. Check to see if the customer already exists as a key in the customer_db
3. If they don't exist in the customer_db, create a new Customer object and add it to the customer_db
4. Whether they already existed or not, get the Customer object associated with the customer name in the customer_db and add the order to the Customer object's order history.
5. Asking for the customer name should happen AFTER you finish adding items to the order but BEFORE you ask for the payment type.

Changes to the Customer class

Add a **class** attribute `id: int` to the Customer class. Give it a reasonable starting value, like 0.

Modify the Customer constructor to increment the id by 1 when a new customer object is created. Do not change the constructor method signature--this attribute is internal use only.

Changes to the Receipt output

Add the following new fields to the top of `receipt.pdf` and the string version of the receipt.

```
Customer Name: Apollo
Customer ID: 1000
Total Orders: 1
```

Test Cases

Add one test to `test_customer.py` to verify that customer id's are unique.

Key Program Requirements

1. Ensure that customer id's are assigned and unique for each customer.
2. The program has a yes/no loop for doing another order.

Example Run

The receipts below are not formatted to match the printed receipts we expect, but all the information you need is there.

```
1: Candy
2: Cookie
3: Ice Cream
4: Sunday
5: Admin Module
What would you like to add to the order? (1-5, Enter for done):

Enter the customer name: Darth Vader
What form of payment will be used? (CASH, CARD, PHONE): CARD

----------------------------------Receipt----------------------------------
Gummy Bears (Bag) 0.25 lbs. @ $0.35/lb.: $0.09 [Tax: $0.01]
Oatmeal Raisin Cookies (Box) 2 cookies @ $3.45/dozen: $0.58 [Tax: $0.04]
Pistachio Ice Cream (Bowl) 2 scoops @ $0.79/scoop: $1.58 [Tax: $0.11]
--------------------------------------------------------------------------
Total number of items in order: 3
Order Subtotals: $2.24 [Tax: $0.16]
Order Total: $2.41
---------------------------------------------------------------------------
Paid for with CARD.
--------------------------------------------------------------------------------
Customer Name: Darth          Customer ID: 1000         Total Orders: 1
Press y and Enter to start a new order.

1: Candy
2: Cookie
3: Ice Cream
4: Sunday
5: Admin Module
What would you like to add to the order? (1-5, Enter for done):

Enter the customer name: James
What form of payment will be used? (CASH, CARD, PHONE): Card

----------------------------------Receipt----------------------------------
Gummy Bears (Bag) 0.25 lbs. @ $0.35/lb.: $0.09 [Tax: $0.01]
Oatmeal Raisin Cookies (Box) 2 cookies @ $3.45/dozen: $0.58 [Tax: $0.04]
Pistachio Ice Cream (Bowl) 2 scoops @ $0.79/scoop: $1.58 [Tax: $0.11]
---------------------------------------------------------------------------
Total number of items in order: 3
Order Subtotals: $2.24 [Tax: $0.16]
Order Total: $2.41
------------------------------------------------------------------------------
```

```
Paid for with CARD.
-----------------------------------------------------------------------------------
Customer Name: James           Customer ID: 1001          Total Orders: 1
Press y and Enter to start a new order.

1: Candy
2: Cookie
3: Ice Cream
4: Sunday
5: Admin Module
What would you like to add to the order? (1-5, Enter for done):

Enter the customer name: Mei
What form of payment will be used? (CASH, CARD, PHONE): Phone

----------------------------------Receipt----------------------------------
Gummy Bears (Bag) 0.25 lbs. @ $0.35/lb.: $0.09 [Tax: $0.01]
Oatmeal Raisin Cookies (Box) 2 cookies @ $3.45/dozen: $0.58 [Tax: $0.04]
Pistachio Ice Cream (Bowl) 2 scoops @ $0.79/scoop: $1.58 [Tax: $0.11]
---------------------------------------------------------------------------
Total number of items in order: 3
Order Subtotals: $2.24 [Tax: $0.16]
Order Total: $2.41
---------------------------------------------------------------------------
Paid for with PHONE.
----------------------------------------------------------------------------------
Customer Name: Mei           Customer ID: 1002          Total Orders: 1
Press y and Enter to start a new order.

1: Candy
2: Cookie
3: Ice Cream
4: Sunday
5: Admin Module
What would you like to add to the order? (1-5, Enter for done):

Enter the customer name: Tony
What form of payment will be used? (CASH, CARD, PHONE): Cash

----------------------------------Receipt----------------------------------
Gummy Bears (Bag)
    0.25 lbs. @ $0.35/lb.: $0.09 [Tax: $0.01]
Oatmeal Raisin Cookies (Box)
    2 cookies @ $3.45/dozen: $0.58 [Tax: $0.04]
```

```
Pistachio Ice Cream (Bowl)
    2 scoops @ $0.79/scoop: $1.58 [Tax: $0.11]
---------------------------------------------------------------------------
Total number of items in order: 3
Order Subtotals: $2.24 [Tax: $0.16]
Order Total: $2.41
----------------------------------------------------------------------------------
Paid for with CASH.
-----------------------------------------------------------------------------------
Customer Name: Tony           Customer ID: 1000          Total Orders: 2
Press y and Enter to start a new order.
```

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)
Changes to DessertShop class	`customer_db` attribute added Assumes unique customer names	`customer_db` attribute present but not fully correct Minor errors in name uniqueness handling	Only basic addition of `customer_db`	No or limited changes to DessertShop class

Criteria	Mastery (100 pts)	Developing (85 pts)	Beginning (70 pts)	Low (50 pts)
Console Application User Input	Asks for customer name post order Properly checks and adds customer to `customer_db` Order added to Customer's history	Asks for customer name but issues with order/history sync Minor errors in `customer_db` handling	Basic functionality added but many errors	No or minimal changes to user input handling
Changes to Customer Class	Class attribute `id` added Unique id generation for each new Customer	`id` attribute present, minor errors in uniqueness Constructor updated but some errors	Basic implementation of `id` attribute	No or limited updates to Customer class
Changes to Receipt Output	All fields added correctly: Customer Name, ID, Total Orders	Most fields added with minor errors	Few fields added or major errors in display	No or minimal changes to receipt
Test Cases Creation	Test verifies uniqueness of customer id's	Test attempts to check uniqueness but with errors	Basic test present but not related to uniqueness	No or minimal tests related to Customer id uniqueness

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.