C9

Classes

CHAPTER ASSIGNMENT

Name _____ Class _____ No. _____ Date _____

> Answer every question in the space given. Show your working. You may use the chapter page to help you.

Questions from *Python Crash Course*, 2nd edition, Eric Matthes (No Starch Press), for classroom use.

Try It Yourself

9-1 **Restaurant:** Make a class called `Restaurant`. The `__init__()` method for `Restaurant` should store two attributes: a `restaurant_name` and a `cuisine_type`. Make a method called `describe_restaurant()` that prints these two pieces of information, and a method called `open_restaurant()` that prints a message indicating that the restaurant is open.

Make an instance called `restaurant` from your class. Print the two attributes individually, and then call both methods.

9-2 **Three Restaurants:** Start with your class from Exercise 9-1. Create three different instances from the class, and call `describe_restaurant()` for each instance.

9-3 **Users:** Make a class called `User`. Create two attributes called `first_name` and `last_name`, and then create several other attributes that are typically stored in a user profile. Make a method called `describe_user()` that prints a summary of the user's information. Make another method called `greet_user()` that prints a personalized greeting to the user.

Create several instances representing different users, and call both methods for each user.

9-4 **Number Served:** Start with your program from Exercise 9-1 (page 162). Add an attribute called `number_served` with a default value of 0. Create an instance called `restaurant` from this class. Print the number of customers the restaurant has served, and then change this value and print it again.

Add a method called `set_number_served()` that lets you set the number of customers that have been served. Call this method with a new number and print the value again.

Add a method called `increment_number_served()` that lets you increment the number of customers who've been served. Call this method with any number you like that could represent how many customers were served in, say, a day of business.

9-5 **Login Attempts:** Add an attribute called `login_attempts` to your `User` class from Exercise 9-3 (page 162). Write a method called `increment_login_attempts()` that increments the value of `login_attempts` by 1. Write another method called `reset_login_attempts()` that resets the value of `login_attempts` to 0.

Make an instance of the `User` class and call `increment_login_attempts()` several times. Print the value of `login_attempts` to make sure it was incremented properly, and then call `reset_login_attempts()`. Print `login_attempts` again to make sure it was reset to 0.

9-6 **Ice Cream Stand:** An ice cream stand is a specific kind of restaurant. Write a class called `IceCreamStand` that inherits from the `Restaurant` class you wrote in Exercise 9-1 (page 162) or Exercise 9-4 (page 167). Either version of the class will work; just pick the one you like better. Add an attribute called `flavors` that stores a list of ice cream flavors. Write a method that displays these flavors. Create an instance of `IceCreamStand`, and call this method.

9-7 **Admin:** An administrator is a special kind of user. Write a class called `Admin` that inherits from the `User` class you wrote in Exercise 9-3 (page 162) or Exercise 9-5 (page 167). Add an attribute, `privileges`, that stores a list of strings like `"can add post"`, `"can delete post"`, `"can ban user"`, and so on. Write a method called `show_privileges()` that lists the administrator's set of privileges. Create an instance of `Admin`, and call your method.

9-8 **Privileges:** Write a separate `Privileges` class. The class should have one attribute, `privileges`, that stores a list of strings as described in Exercise 9-7. Move the `show_privileges()` method to this class. Make a `Privileges` instance as an attribute in the `Admin` class. Create a new instance of `Admin` and use your method to show its privileges.

9-9 **Battery Upgrade:** Use the final version of *electric_car.py* from this section. Add a method to the `Battery` class called `upgrade_battery()`. This method should check the battery size and set the capacity to 100 if it isn't already. Make an electric car with a default battery size, call `get_range()` once, and then call `get_range()` a second time after upgrading the battery. You should see an increase in the car's range.

9-10 **Imported Restaurant:** Using your latest `Restaurant` class, store it in a module. Make a separate file that imports `Restaurant`. Make a `Restaurant` instance, and call one of `Restaurant`'s methods to show that the `import` statement is working properly.

9-11 **Imported Admin:** Start with your work from Exercise 9-8 (page 173). Store the classes `User`, `Privileges`, and `Admin` in one module. Create a separate file, make an `Admin` instance, and call `show_privileges()` to show that everything is working correctly.

9-12 **Multiple Modules:** Store the `User` class in one module, and store the `Privileges` and `Admin` classes in a separate module. In a separate file, create an `Admin` instance and call `show_privileges()` to show that everything is still working correctly.

9-13 **Dice:** Make a class `Die` with one attribute called `sides`, which has a default value of 6. Write a method called `roll_die()` that prints a random number between 1 and the number of sides the die has. Make a 6-sided die and roll it 10 times.

Make a 10-sided die and a 20-sided die. Roll each die 10 times.

9-14 **Lottery:** Make a list or tuple containing a series of 10 numbers and five letters. Randomly select four numbers or letters from the list and print a message saying that any ticket matching these four numbers or letters wins a prize.

9-15 **Lottery Analysis:** You can use a loop to see how hard it might be to win the kind of lottery you just modeled. Make a list or tuple called `my_ticket`. Write a loop that keeps pulling numbers until your ticket wins. Print a message reporting how many times the loop had to run to give you a winning ticket.

9-16 **Python Module of the Week:** One excellent resource for exploring the Python standard library is a site called *Python Module of the Week*. Go to *https://pymotw.com/* and look at the table of contents. Find a module that looks interesting to you and read about it, perhaps starting with the `random` module.

End of assignment · Check your answers against the chapter at gurujazzy.com/subjects/python-crash-course/chapter-9-classes/