**CSE 1321L: Programming and Problem Solving I Lab**

**Fall 2025**

**Assignment 6**

**Module 5**

# What students will learn

- Design classes including attributes, constructors, and methods
- Creating objects of classes

# Content

- Overview
- Assignment6A: Owl Rent-a-Car

# Overview

For this assignment, you are going to practice implementing and designing classes defining its attributes and behaviors by implementing a constructor function and other member functions.

**Final note: *Do not cheat***
If your temptation is to look online, don't. Come see us instead and ask questions – we are here to help. Remember, you are going to have to write code in your future job interviews, so learn it now to secure a high-paying job later.

Lastly, make sure you review the sample output and make sure the output of your program follows the exact same format including the input statements, print statement, etc. As always, user input is shown in **red** and **bold**.

# Assignment 6A: Owl Rent-a-Car

You will develop a program for a car rental company. The program will be used by the sales team to rent a car to a customer and to process the return of a rented car.

The program should implement two classes: Car and OwlRental. The Car class will handle the information and behaviors regarding a specific car. Moreover, the OwlRental class will handle the all of the cars owned by the rental company and will feature functions that handles the rental and return processes.

## Requirements

- **Car** class:
  - The constructor function takes 4 parameters, a parameter for the make, model, year, and the per day rate of the car.
  - The constructor should set the instance attributes **make**, **model**, **year**, and **per-day-rate** accordingly.
  - The constructor should also initialize a Boolean attribute called **isRented** which should be set to **False**.
  - Moreover, the class the following functions:
    - is_available()
    - set_rented()
    - set_returned()
    - info()
  - **is_available()** should not take in any parameters and will just return the car object's **isRented** attribute value.
  - **set_rented()** should take in a value for the number of days the car will be rented. The function should set the **isRented** car attribute as **True** and print the information of the car rented (review sample output).
  - **set_returned()** should not take any parameters. The function should set the **isRented** car parameter as **False** and print a message saying the car was returned (review sample output).
  - **info()** should not take any parameters. The function will print the car object's information. If the car is not rented it should print as available and it's the per-day-rate. If the car is currently being rented, it should print it as rented (review sample output)
- **OwlRental** class:
  - The constructor function does not take any parameters.
  - The constructor should initialize an instance attribute called **depot** and set it as an empty list.
    - This list will be used to "store" car objects.
  - The class should contain the following functions:
    - add_car()
    - list_cars()
    - rent_car()
    - return_car()

- **add_car()** should take 4 parameters, one parameter for the make, model, year, and per-day-rate. The function should use these values to append a car object in **depot**.
- **list_cars()** should take no parameters. The function should print all the cars in the **depot** list by calling each car's **info()** function. Moreover, it should also print the index location of each car (review sample output).
- **rent_car()** should take 2 parameters, one parameter that will serve as an index for depot, and the number of days the car will be rented.
    - The function should check if the car at the specified index is available. If not, it should print an error message.
    - If the car is available, call the car's **setAsRented()** function and pass as an argument the appropriate value.
- **return_car()** should take 1 parameter that will serve as an index for depot.
    - The function should check if the car at the specified index is not available (not rented). If so, print an error message.
    - Else, the program should call the car's **setAsReturned()** function.

o **Main** program:

- Start the program by creating an **OwlRental** object.
- Add these three cars by calling the **add_car()** function of your **OwlRental** object:
    - 2025 Toyota Corolla, rate: $49.99 per day
    - 2023 Honda Civic, rate: $45.50 per day
    - 2023 Tesla Model 3, rate: $119.00 per day
- Start the main loop of the program by asking the user to choose an option:
    - 1. Rent
    - 2. Return
    - 3. View cars
    - 4. Exit
- If the user chooses to rent, list the cars by calling the appropriate function of your OwlRental object, then prompt the user for an index and how many days they want to rent the car for. Then, call the appropriate function from your OwlRental object to rent the specified car.
- If the user chooses to return, list the cars by calling the appropriate function of your OwlRental object, then prompt the user for an index. Call the appropriate function from your OwlRental object to return the specified car.
- If the user chooses to view cars, list the cars by calling the appropriate function of your OwlRental object.
- Stop the loop if the user chooses option 4.

Example runs are shown below. The user input is shown in **red**

## Sample Output #1

```
[Owl Rent-a-Car]
1. Rent
2. Return
3. View cars
4. Exit
```

```
> 1

0 - 2025 Toyota Corolla - AVAILABLE, PER DAY RATE: $49.99
1 - 2023 Honda Civic - AVAILABLE, PER DAY RATE: $45.5
2 - 2023 Tesla Model 3 - AVAILABLE, PER DAY RATE: $119.0
Select an index: 0
How many day(s)?: 10
2025 Toyota Corolla has been rented for 10 days.
Customer will be charged $499.90 at return

1. Rent
2. Return
3. View cars
4. Exit
> 1

0 - 2025 Toyota Corolla - RENTED
1 - 2023 Honda Civic - AVAILABLE, PER DAY RATE: $45.5
2 - 2023 Tesla Model 3 - AVAILABLE, PER DAY RATE: $119.0
Select an index: 0
How many day(s)?: 14
Error: Car is already rented

1. Rent
2. Return
3. View cars
4. Exit
> 3

0 - 2025 Toyota Corolla - RENTED
1 - 2023 Honda Civic - AVAILABLE, PER DAY RATE: $45.5
2 - 2023 Tesla Model 3 - AVAILABLE, PER DAY RATE: $119.0

1. Rent
2. Return
3. View cars
4. Exit
> 2

0 - 2025 Toyota Corolla - RENTED
1 - 2023 Honda Civic - AVAILABLE, PER DAY RATE: $45.5
2 - 2023 Tesla Model 3 - AVAILABLE, PER DAY RATE: $119.0
Select an index: 1
Error: Car is not rented

1. Rent
2. Return
3. View cars
```

```
4. Exit
> 2

0 - 2025 Toyota Corolla - RENTED
1 - 2023 Honda Civic - AVAILABLE, PER DAY RATE: $45.5
2 - 2023 Tesla Model 3 - AVAILABLE, PER DAY RATE: $119.0
Select an index: 0
2025 Toyota Corolla has been returned

1. Rent
2. Return
3. View cars
4. Exit
> 3

0 - 2025 Toyota Corolla - AVAILABLE, PER DAY RATE: $49.99
1 - 2023 Honda Civic - AVAILABLE, PER DAY RATE: $45.5
2 - 2023 Tesla Model 3 - AVAILABLE, PER DAY RATE: $119.0

1. Rent
2. Return
3. View cars
4. Exit
> 4
```

## Submission Instructions:

- Programs must follow the output format provided. This includes each blank line, colons (:), and other symbols.
- Programs must be working correctly.
- Programs must be written in Python.
- Programs must be submitted with the correct **.py** format.
- Programs must be saved in files with the correct file name:
  - Assignment6A.py
- Programs (source code files) must be uploaded to Gradescope by the due date.