Advanced Python

Object oriented programming in Python

Assignment #1

Assignment on OOP:

1. Create a class called "Book" with the following data members:
    - title (string)
    - author (string)
    - ISBN (string)
    - publisher (string)
    - price (float)
2. Implement a constructor method for the class that initializes all the data members with the values passed as arguments.
3. Create a method called "display_book_info" that prints all the data members of the class in the following format:
    - "Title: [title]\nAuthor: [author]\nISBN: [ISBN]\nPublisher: [publisher]\nPrice: [price]"
4. Override the built-in `__str__` method in the class so that when a Book object is printed, it displays the same information as the "display_book_info" method.
5. Create a class called "Library" with the following data members:
    - name (string)
    - location (string)
    - books (list of Book objects)
6. Implement a constructor method for the class that initializes the name and location data members with the values passed as arguments and initializes the books data member as an empty list.
7. Create a method called "add_book" that takes a Book object as an argument and adds it to the books data member.
8. Create a method called "display_library_info" that prints the name and location of the library and then calls the "display_book_info" method on each book in the books data member.
9. Create a main function that creates a Library object and a few Book objects, adds the Book objects to the Library, and then calls the "display_library_info" method on the Library object.

10. Submit your code and explanation on how you approached the problem, what you learned and what challenges you faced.

Assignment #2

Assignment on inheritance:

1. Create a class called "Vehicle" with the following data members:
    - make (string)
    - model (string)
    - year (int)
2. Implement a constructor method for the class that initializes all the data members with the values passed as arguments.
3. Create a method called "display_vehicle_info" that prints all the data members of the class in the following format:
    - "Make: [make]\nModel: [model]\nYear: [year]"
4. Create a class called "Car" that inherits from the "Vehicle" class and has an additional data member called "num_doors" (int).
5. Implement a constructor method for the "Car" class that takes the same arguments as the "Vehicle" class constructor, and an additional argument for the num_doors data member.
6. Override the "display_vehicle_info" method in the "Car" class so that it also prints the num_doors data member.
7. Create a class called "Truck" that inherits from the "Vehicle" class and has an additional data member called "bed_size" (string).
8. Implement a constructor method for the "Truck" class that takes the same arguments as the "Vehicle" class constructor, and an additional argument for the bed_size data member.
9. Override the "display_vehicle_info" method in the "Truck" class so that it also prints the bed_size data member.
10. Create a main function that creates a Car and a Truck object, and then calls the "display_vehicle_info" method on each object.
11. Submit your code and explain how you approached the problem, what you learned, and what challenges you faced.

Assignment #3

Assignment on magic operator functions:

1. Create a class called "Student" with the following data members:
    - name (string)
    - age (int)
    - grade (float)
2. Implement a constructor method for the class that initializes all the data members with the values passed as arguments.
3. Override the built-in `__lt__` method in the class so that two Student objects can be compared using the < operator based on their grade. If the grade of one student is lower than the other, the method should return True, otherwise False.
4. Override the built-in `__le__` method in the class so that two Student objects can be compared using the <= operator based on their grade. If the grade of one student is lower or equal than the other, the method should return True, otherwise False.
5. Override the built-in `__gt__` method in the class so that two Student objects can be compared using the > operator based on their grade. If the grade of one student is greater than the other, the method should return True, otherwise False.
6. Override the built-in `__ge__` method in the class so that two Student objects can be compared using the >= operator based on their grade. If the grade of one student is greater or equal than the other, the method should return True, otherwise False.
7. Create a main function that creates several Student objects and compares them using the <, <=, >, >= operators.
8. Submit your code and explain how you approached the problem, what you learned, and what challenges you faced.
