Python Programming Problems

1. slope_calculator.py
    a. Contains a function called calc_slope with 3 input parameters and returns the slope
    b. Run the function with y=28, x=3, b=5
    c. Print the slope that was returned from the function with ‘Slope = ‘ in front of the number
2. slope_calculator_list.py
    a. Contains a function called calc_slope with 1 input parameter, which is a list that contains 4 elements:
        i. y
        ii. m
        iii. x
        iv. b
    b. The function should return the slope
    c. You should test it with 2 lists:
        i. lineList1 = [28, 999, 3, 5]
        ii. lineList2 = [135, 888, 21, -325]
    d. For both lists, print:
        i. The slope of the line as returned from the function
        ii. y, m, x, and b after running the function
            1. You should update the lists with the calculated slope
3. slope_calculator_dictionary.py
    a. Repeat the previous problem but use a dictionary instead of a list
4. line_calculator.py
    a. This file will only contain functions. The intent is to be able to import this into another file as a module.
    b. The functions are:
        i. calc_slope
        ii. calc_y
        iii. calc_x
        iv. calc b
    c. Each function has 1 input parameter, which is a list that contains 4 elements:
        i. y
        ii. m
        iii. x
        iv. b
    d. Each function should return the value that was calculated
    e. You will not test this function yet
5. slope_calculator_using_module.py
    a. This file should import your line_calculator and test each function by passing in a list into the module functions
        i. You can choose the lists, just verify that it works produces the correct outputs

6. line_calculator_using_class.py
    a. Create a class called LineCalculator with the following characteristics:
        i. Initialization parameters should be a name for the line and a lineData list
        ii. Member functions for calculating y, m, x, and b (as done in the previous problems)
    b. Create 4 instances of the class with different names and lists
    c. Test out the member functions with these instances to make sure they calculate the values correctly.
7. automobile_class.py
    a. Create the Automobile class described in the lecture
        i. You only need to implement the CalculateGallonsUsed member function, not the other functions
    b. Test the class with the following code:

```
vehicle1 = Automobile("Toyota", "Camry Hybrid", 48, 500, "regular")
v1Gallons = vehicle1.CalculateGallonsUsed(250)

vehicle2 = Automobile("Honda", "CRV Hybrid", 35, 450, "regular")
v2Gallons = vehicle2.CalculateGallonsUsed(250)

print(vehicle1.make, vehicle1.model, " used ", v1Gallons, " of gas")
print(vehicle2.make, vehicle2.model, " used ", v2Gallons, " of gas")
```

8. automobile_inheritance.py
    a. Use the Automobile class from the previous problem and create a child class called Truck
    b. Truck should have everything that Automobile has but it should add:
        i. cargoCapacity
        ii. cargoWeight
    c. To test, thefunction, use the same provided code from the previous problem and add the following to the bottom:

```
vehicle3 = Truck("Ford", "F-350", 10, 400, "diesel", 2000, 1500)
v3Gallons = vehicle3.CalculateGallonsUsed(250)
print(vehicle3.make, vehicle3.model, " used ", v3Gallons, " of gas while carrying ", vehicle3.cargoWeight,
" lbs of cargo")
```