Python Programming Problems

1. slope\_calculator.py
   1. Contains a function called calc\_slope with 3 input parameters and returns the slope
   2. Run the function with y=28, x=3, b=5
   3. Print the slope that was returned from the function with ‘Slope = ‘ in front of the number
2. slope\_calculator\_list.py
   1. Contains a function called calc\_slope with 1 input parameter, which is a list that contains 4 elements:
      1. y
      2. m
      3. x
      4. b
   2. The function should return the slope
   3. You should test it with 2 lists:
      1. lineList1 = [28, 999, 3, 5]
      2. lineList2 = [135, 888, 21, -325]
   4. For both lists, print:
      1. The slope of the line as returned from the function
      2. y, m, x, and b after running the function
         1. You should update the lists with the calculated slope
3. slope\_calculator\_dictionary.py
   1. Repeat the previous problem but use a dictionary instead of a list
4. line\_calculator.py
   1. This file will only contain functions. The intent is to be able to import this into another file as a module.
   2. The functions are:
      1. calc\_slope
      2. calc\_y
      3. calc\_x
      4. calc b
   3. Each function has 1 input parameter, which is a list that contains 4 elements:
      1. y
      2. m
      3. x
      4. b
   4. Each function should return the value that was calculated
   5. You will not test this function yet
5. slope\_calculator\_using\_module.py
   1. This file should import your line\_calculator and test each function by passing in a list into the module functions
      1. You can choose the lists, just verify that it works produces the correct outputs
6. line\_calculator\_using\_class.py
   1. Create a class called LineCalculator with the following characteristics:
      1. Initialization parameters should be a name for the line and a lineData list
      2. Member functions for calculating y, m, x, and b (as done in the previous problems)
   2. Create 4 instances of the class with different names and lists
   3. Test out the member functions with these instances to make sure they calculate the values correctly.
7. automobile\_class.py
   1. Create the Automobile class described in the lecture
      1. You only need to implement the CalculateGallonsUsed member function, not the other functions
   2. 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")

1. automobile\_inheritance.py
   1. Use the Automobile class from the previous problem and create a child class called Truck
   2. Truck should have everything that Automobile has but it should add:
      1. cargoCapacity
      2. cargoWeight
   3. 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")