Analyzing Car Data Using Python

```
#math so I can use the square root function (sqrt())
import math
#matplotlib is a free use python module that is used to graph data structures visually
import matplotlib.pyplot as plot

class car():
    def __init__(self, CarFolder):
        #These are self variables that are used throughout the program
        self.Make = ''
        self.Model = ''
        self.Year = 0
        self.Price = 0.0
        self.Mass = 0
        self.Wheels = []
        self.TQ = []
        self.G = []
        self.I = []
        self.Gearing = []
        self.FinalGearing = []
        self.FinalSpeeds = []
        self.Wheels = []
        self.SpeedRating = ""
        self.Accelerationgraph = []


        #Reading from the text files, saving to self
        with open(CarFolder + '/CarInfo', 'r') as Info:
            #reading the file Info
            text = Info.read()
            #splitting up the text by ' ' characters (space), creating a list of values from Info
            self.I = text.split()
            #print out each value in self.I
            print("Make: " + self.I[0])
            print("Model: " + self.I[1])
            print("Year: " + self.I[2])
            print("Price: $" + self.I[3])
            print("Mass: " + self.I[4] + " kg")
            print("Wheel Diameter: " + self.I[5] + " cm")
            print("Max traction : " + self.I[6] + " g's" + '\n')


        with open(CarFolder + '/Torque', 'r') as Torque:
            print("     RPM : Torque")
            text = Torque.read().split()
            for value in text:
                #temp variable to store the different tq/rpm configurations
                temp = value.split(',')
                #adding list of [rpm, tq]
                self.TQ.append([int(temp[0]), int(temp[1])])
                #printing out these tq/rpm values from dyno chart in Torque file
                print(temp[0]+ " RPM : " + temp[1] + " Nm")
    print()

#Printing out the transmission gearing
with open(CarFolder + '/Gearing', 'r') as Gearing:
    self.G = Gearing.read().split()
    print("Transmission Gearing")
    count = 0
    #for each valueu in the gearing, print out the gear number, a ":", and the ratio
    while(count < len(self.G) - 1):
        print(count + 1, ':', self.G[count])
        count += 1
    #printing the differential gear ratio
    print("Differential Ratio: " + self.G[-1] + "\n")


#Setting final gearing
counter = 0
print("Final Gearing:")
while(counter < len(self.G)-1):
    #calculation is equal to gear ratio multiplied by the differential gear ratio
    calculation = round(float(self.G[counter]) * float(self.G[-1]), 3)
    #saving all of the calculations to self.FinalGearing
    self.FinalGearing.append(calculation)
    #print out each calculation the same way as the transmission gears
    print(counter + 1, ":", calculation)
    counter += 1
print()


#Reading self.I and saving values appropriately
#Converting values that are text into integer/float values to be used mathematically
self.Make = self.I[0]
self.Model = self.I[1]
self.Year = int(self.I[2])
self.Price = int(self.I[3])
self.Mass = int(self.I[4])
self.Wheels = [int(self.I[5]),float(self.I[6])]


#Calculating top speed in each gear
print("Final Speeds Per Gear")
itr = 1
#for each gear ratio in the final gearing, calculate the top speed given the max rpm, the gear r
for gearratio in self.FinalGearing:
    calculation = int(self.TQ[-1][0] / gearratio * self.Wheels[0] * 3.141 * 60/100000)
    self.FinalSpeeds.append(calculation)
    print(itr, ":", calculation, "kph")
    itr +=1

#Setting the SpeedRating for the car by analyzing the top speed of the vehicle
if(self.FinalSpeeds[-1] >= 350):
    self.SpeedRating = "Hypercar"
elif(self.FinalSpeeds[-1] >= 225 and self.FinalSpeeds[-1] < 350):
    self.SpeedRating = "Supercar"
elif(self.FinalSpeeds[-1] >= 160 and self.FinalSpeeds[-1] < 225):
    self.SpeedRating = "Sports car"
```

```
        else:
            self.SpeedRating = "Daily car"
        print(self.SpeedRating, '\n')


    #Calculating the acceleration curve with parameter self. self includes all of the data collected in 
    def calculateacceleration(self):
        counter = 1
        print("Acceleration Curve")
        #for each gear ratio, calculate the acceleration at a given rpm throughout the power band using 
        for gear in self.FinalGearing:
            print("Gear", counter)
            for tq in self.TQ:
                calculation = [counter, int(tq[0]), round(10 * (int(tq[1])*gear)/(self.Mass * self.Wheel
                self.Accelerationgraph.append(calculation)
                #if the acceleration exceeds the maximum acceleration of the tires, then there will be a
                if(calculation[-1] > float(self.I[6])):
                    print(calculation, "Traction Loss")
                else:
                    print(calculation)
            counter += 1
            print()

#Actual program:
#calling the car object with filename "Koenigsegg" <- data for this car
Car1 = car("Koenigsegg")
#Required for AP Exam <- Parameter, if statement, loop
Car1.calculateacceleration()

# Graphing each gear differently
listx = []
listy = []
#allocating space in these lists as a 2d data structure so that we can hold (rpm,acceleration) values
for ind in Car1.G:
    listx.append([])
    listy.append([])

gear = 0
#For each acceleration value in each gear, save values to listx and listy so that matplotlib can graph a
for i in Car1.Accelerationgraph:
    if(i[0] == gear + 1):
        gear += 1
        listx[gear].append(i[1])
        listy[gear].append(i[2])
        plot.plot(listx[gear],listy[gear], marker = '*', linestyle = '-')
    elif(i[0] == gear):
        listx[gear].append(i[1])
        listy[gear].append(i[2])
        plot.plot(listx[gear],listy[gear], marker = '*', linestyle = '-')



# Setting the xlabel and ylabel and title of in matplotlib graph
plot.xlabel('RPM')
plot.ylabel("Acceleration (g's)")
plot.title('Acceleration/RPM Per Gear')
```

```python

# Show grid in graph so that visual data has a reference
plot.grid(True)

# Render the plot/graph with matplotlib
plot.show()

```