**Home Assignment Hand-out**

| Course: | Python: Introduction to Coding for Artificial Intelligence and Mechatronics |
| --- | --- |
| Assignment: | Day 2 |
| Lecturer: | Dr. Jonathan Lesage [jonathan.lesage@gmail.com](mailto:jonathan.lesage@gmail.com) |
| Handed out: | Tuesday, January 24, 2023 |
| Submission deadline: | Monday, February 6, 2023, Midnight |
| To be submitted to: | SCFI at [scfi@magna.com](mailto:scfi@magna.com) |
| Format: | python module |
| File name should be: | Assignment\_Python\_D2\_studentlastname\_firstname.py.txt |
| Subject of e-mail should be: | SCFI |

**INSTRUCTIONS**:

Please answer all questions.

If an answer requires a calculation, show it. Don’t just quote an answer.

If doubt exists as to the interpretation of any question, contact the lecturer for clarification. See table above for contact information. If no help can be obtained, for any question requiring a written answer, then submit a clear statement of any assumptions you make to get to an answer.

This assignment is open book. For the purpose of study, you may consult outside papers, books, and electronic sources of information, but cannot use text or figures from outside sources as part of your answers.

You can never represent as your own the work of anyone else. Everything you write must be your individual work and must be in your words. Absolutely no collaboration is allowed among students in activities used for academic performance evaluation. The word “collaboration” includes activities such as copying parts of assignments from each other (or from a third-party source), comparing answers, transferring templates or answer formats from one student to another, and discussions on methods to arrive at answers.

Please only submit your PYTHON module Assignment\_Python\_D2\_studentlastname\_firstname.py.txt, no need to transcribe answers to a separate word file.

Follow the file naming protocol as given in the information table above.

If there are attachments, print your name on the top left corner of the first page of every attachment you submit and follow the file naming protocol.

Always follow the file format requirements given in the assignment hand out. If, for example, the instructor asks for PDF files only, don’t send Word documents, Excel files, or scanned JPG picture files as those will be rejected.

Submit separate files for separate assignments. Don’t combine the assignments of different modules of the course into one file of zip folder, even if those are to be forwarded to the same lecturer who taught various modules of the course. SCFI can only track assignment submission if those files are received as separate files and can be saved in separate folders dedicated to each course module.

Magna’s email system allows for attachments of 20MB or 25MB (check with your IT team) that should be enough for your file, so whenever possible, avoid submitting zipped files and folders. If any single file would exceed this limit, use only the zip format natively supported by Windows.

You have been tasked with estimating the approximate raw steel cost for various vehicle frames manufactured by Magna (2025 models). You have been supplied with worldwide 2025 pre-orders which includes the make and model for each vehicle to be produced. In order to accomplish this you will need to write a custom module with the following:

1. A function entitled ReadDelimited which takes a file name and optional arguments delimiter, tofloat, e.g.:

def ReadDelimited(filepath, delimiter = “,”, tofloat = True):

return DataDictionary

The delimiter is the character which separates the columns of the file. The function returns a dictionary called “DataDictionary” which contains lists which are the columns of the file, each indexed by their corresponding header (entry on the first row) of a given column. If the tofloat argument is set to True then the columns should be converted to floating point numbers using the float() command, otherwise these values are to be left as strings.

In contrast to the previous assignment, you must make use of the string manipulation methods .split() and .strip() (useful to get rid of those pesky line break characters!). Verify that your code works correctly by loading the RawMaterialDB.txt file (provided). Ensure that the code works for the case where only a single column exists, i.e., there are no delimiters. Hint: You may pass None to the .split() method.

1. A class called “Vehicle” with the attributes:

RawMaterialDB - A string specifying the path to the RawMaterialDB.txt file

VehicleID - A string variable identifying the vehicle in Make-Model format (e.g. “Honda-Civic”)

SteelContent - A float specifying the steel content in the vehicle as the weight in pounds

SteelPricePerKg - A float specifying the current price of steel per kilogram

You should assume that all of these attributes will be private and require appropriate public Set and Get methods. Your constructor should require the user to specify the VehicleID and SteelPricePerKg but leave the rawmaterialsdb as an optional argument, that is:

\_\_init\_\_(self, vehicleid, steelprice, rawmaterialdb = ‘.../RawMaterialDB.txt’):

The RawMaterialDB.txt file (provided) is tab delimited file with different vehicles in the format “Make-Model” on the first line (row) and the corresponding weight of steel on the second line (in pounds).

The private Set method for the SteelContent attribute should use the ReadDelimited function (located in the same module) to parse the RawMaterialDB.txt file and read the correct values for the VehicleID, e.g.:

def \_\_SetSteelContent(self):

db = ReadDelimited( arguments )

code to read steel content for given vehicle

self.\_\_SteelContent = Steel content read from file

You should ensure that if the VehicleID is not in the RawMaterialDB file, the user should be alerted and the SteelContent attribute is set to None. Also, you should ensure that in the Set method for VehicleID, the appropriate code gets called to update the SteelContent.

1. A function called ComputeSteelPrice which takes the weight of steel in pounds and the price of steel in USD per kilograms and returns the price of the given weight of steel in USD, e.g.:

def ComputeSteelPrice(SteelWeightPounds, SteelPricePerKg):

code to compute total price of the steel weighed in pounds

return PriceOfSteel

1. A function called EstimateRawMaterialCost which takes a string specifying the path to a preorders file, which has a different Vehicle in Make-Model format on each line. This function should use the ReadDelimited function to read the PreOrders.txt file provided, and make use of the Vehicle class together with the ComputeSteelPrice function to compute the total price of steel for all vehicles in the PreOrdersFile, e.g.:

def EstimateMaterialCost(PathToPreOrdersFile, SteelPricePerKg):

code to estimate total steel price

return TotalSteelPrice

Use the market price of steel to test the function.

Please only provide the module code***. All functions and the Vehicle class are to be in this module***. You should test/debug the code by importing the module into a Jupyter Notebook and running the commands necessary. The PreOrders.txt file will be provided with the assignment. **You do not need to provide the .ipynb file used for testing, only the module (.py) file.**

Note that if you are making changes to a module and want to see those update in the a jupyter notebook you will need to reload the module using the reload function from the imp module which you can load as follows:

from imp import reload

Now to reload a previously loaded module which you have made a change to:

reload(Assignment\_Python\_D2\_studentlastname\_firstname)