Lab Week 8 - File handling, Try except

Skills Needed to complete this Lab

|  |
| --- |
| * Understand how to open, close, read and write files * Be able to respond to common errors   + FileNotFound   + IOError   + ValueError   + etc |

## Fuel Economy

In this lab we’ll write a program to read through a file containing information about fuel economy and output the results to a file above a threshold that the user gives. If the user wants to see all vehicles with a combined mpg greater than 50, then your program will output that information to the file of their choosing. The information is tab-delimited. When you read a line from the file, the values are separated by the tab character **\t**. There are many ways to split this string data.

Vehicle file organization

|  |  |
| --- | --- |
| Column Name | Description |
| year | The year of the vehicle |
| make | The make of the vehicle |
| model | The vehicle model |
| cylinders | The number of cylinders in the vehicle |
| trany | The type of transmission ( Manual, 5-spd, automatic, etc ) |
| Vclass | Vehicle class ( Two Seaters, subcompact, etc ) |
| displ | Engine displacement of the vehicle in liters |
| combinedmpg | The combined city and hwy fuel economy |
| citympg | The fuel economy in the city |
| highwaympg | The highway fuel economy |

Your program will need to ask the user for a minimum fuel economy, and be able to handle non float data being entered. It should continually ask for a correct value. It should also make sure they don’t enter a value less than or equal to zero or greater than 100. The program will ask for an input file and should loop until the user gives a valid file that can be opened. It should also ask for an output file. While a FileNotFoundError won’t be thrown trying to open a file in write mode, it can generate an IOError.

When writing out to the output file, you should output the Year, Make and Model left justified by 40 spaces and the mpg right justified by 10 spaces.

Your program should also be able to handle an incorrect combinedmpg column as in vehicles2.txt. When a bad value that can’t be converted is encountered, you should warn the user and give the year, make and model that had the error.

### Example Program

>>> ================================ RESTART ================================

>>>

Enter the minimum mpg ==> bad input

You must enter a number for the fuel economy

Enter the minimum mpg ==> -10

Fuel economy given must be greater than 0

Enter the minimum mpg ==> 300

Fuel economy must be less than 100

Enter the minimum mpg ==> 80

Enter the name of the input vehicle file ==> invalidfile.txt

Could not open file invalidfile.txt

Enter the name of the input vehicle file ==> vehicles2.txt

Enter the name of the file to output to ==> out\*.txt

There is an IO Error out\*.txt

Enter the name of the file to output to ==> output.txt

Could not convert value invalid for vehicle 1993 Chevrolet Caprice

Could not convert value nine for vehicle 1993 Chrysler Concorde

>>>

### Example output.txt file

2000 Nissan Altra EV 85.000

1999 GMC EV1 85.000

2008 MINI MiniE 98.000

2011 Nissan Leaf 99.000

Etc….

**Grading and Turning In**

Turn in your program before the end of the lab. Only upload the Solution .py file, as other files will be ignored. Grading will be performed with a different input file.