# CSC121 Lab 08: Files and Exceptions

## Goals

In this lab assignment, students will demonstrate the ability to:

* Write code to read data from a text file
* Write code to write output to a text file
* Use loops to process files and records
* Handle runtime errors and validate data using exception handling

## Instructions

In this lab, you will demonstrate your mastery of file input/output and exception handling.

Follow the instructions in each problem and submit the specified files.

Problem 1 consists of a program that you create from scratch that meets the problem specification. For Problem 2, you'll be given a starter file which will have functions you need to implement.

## Problems

### Problem 1

Trish at Bargain Used Books wants computer programs to help her keep inventory of the store. This first program will read in an inventory file.

This program will open inventory.csv for reading. Then it will read the item data in the file and display the entries. Here's an example inventory.csv file:

Science Book,10,12.9,Science textbook for students

Math Book,15,13.95,Math textbook for students

Fiction Book,32,7.0,Novel for casual readers

Each line of the file shows the name of an inventory item, the number of the item that is available, the unit cost of the item, and a description. Each value is separated by commas. Each line represents a different item.

Create a file named **Lab08P1.py**. Write a program that does the following:

* The program attempts to open a file called inventory.csv. If the file is not present, the program prints "File inventory.csv not found" and terminates.
* If the file is present, read each line of the file into a string.
  + The string that has been read in contains the item name, item count, unit price, and description separated by commas.
  + Split the string into the four parts using the split() function you learned about in Lesson 7. That code should create a list with four elements.
  + Assign each list element to a variable representing the item name, item count, unit price, and description.
    - The item count variable should be of type int.
    - The unit price variable should be of type float.
  + Output to the screen the item data as shown in the Sample Output below.
    - The second and third lines of the item data are shifted to the right using the '\t' character.
    - As we've done for multiple labs, remember to display monetary values with 2 digits after the decimal point.
* NOTE: You may assume that all lines in inventory.csv are valid, and it is not required to do error checking on the file input.

Sample Output:

Science Book

10 in stock, $12.90

Science textbook for students

Math Book

15 in stock, $13.95

Math textbook for students

Fiction Book

32 in stock, $7.00

Novel for casual readers

Submit the program file **Lab08P1.py** to Blackboard for credit.

### Problem 2

This is the second program that Trish at Bargain Used Books needs for her inventory tracking. This program will write to an inventory file.

A starter file called **Lab08P2-starter.py** has been provided on Blackboard. Download this file and rename it to **Lab08P2.py**. Fill in the implementation to five functions:

* Data validation functions:
  + get\_item\_name() - Ask the user to enter an item name. Item names will not be allowed to contain commas, so the program should test for that. If an item name contains a comma, give the user an error, and ask for the item name again. The function should return the validated item name.
  + get\_item\_count() - Ask the user to enter an item count. The count should be validated to be an integer using exception handling. Additionally, the number entered cannot be less than 0. If the value entered is invalid, give the user an error and ask for the item count again. The function should return the validated item count.
  + get\_unit\_cost() - Ask the user to enter an item cost. The cost should be validated to be a float using exception handling. Additionally, the number entered cannot be less than 0. If the value entered is invalid, give the user an error and ask for the item cost again. The function should return the validated unit cost.
  + get\_description() - Ask the user to enter an item description. Descriptions will not be allowed to contain commas, so the program should test for that. If a description contains a comma, give the user an error, and ask for the description again. The function should return the validated item description.
* File writing function:
  + write\_inventory\_file() – The inventory data was stored as a list of strings with each element being a record that should be written to inventory.csv:
    - Open a file called inventory.csv. If the file exists, it will be overwritten.
    - Loop through inventory\_list and get each string element. Write each string to a separate line in the file.
    - After finishing the loop, close the file.
    - The file inventory.csv should now contain the inventory items entered by the user. Print the message "The file inventory.csv was created."

Here's one possible sample output run:

Welcome to Trish's Inventory Input System

Enter the item name: Science, Math, and Fiction Books

Item names cannot contain commas.

Enter the item name: Science Book

Enter the item count: ten

Item count must be an integer.

Enter the item count: -10

Item count must be 0 or greater.

Enter the item count: 10

Enter the unit cost: Twelve Dollars

Unit cost can only contain digits and a single decimal point.

Enter the unit cost: 12.00

Enter the description: Detailed, but consise, guidebook for students

Descriptions cannot contain commas.

Enter the description: Detailed and concise guidebook for students

Enter another item? (y/n) y

Enter the item name: Math Book

Enter the item count: 15

Enter the unit cost: 13.95

Enter the description: Math textbook for 1st year college

Enter another item? (y/n) y

Enter the item name: Fiction Book

Enter the item count: 32

Enter the unit cost: 7.90

Enter the description: Dramatic tales of heroic deeds

Enter another item? (y/n) n

The file inventory.csv was created.

After running this program with the above input, here's what inventory.csv would look like:

Science Book,10,12.0,Detailed and concise guidebook for students

Math Book,15,13.95,Math textbook for 1st year college

Fiction Book,32,7.9,Dramatic tales of heroic deeds

Note that if you have a working solution for Problem 1, you can read the inventory.csv file created by this solution.

Submit the program file **Lab08P2.py** to Blackboard for credit.

## Grading Rubric

### Grading rubric for Problem 1 (30 points)

* Program has a well-formatted and correct header [5 points]
* Program does execute correctly and produces correct results [25 points]

### Grading rubric for Problem 2 (70 points)

* Program has a well-formatted and correct header [10 points]
* Program does execute correctly and produces correct results [60 points]