# 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

NOTE: This lab document is a copyrighted work of Wake Tech Community College and the course instructor. Any posting of this document outside of Wake Tech is considered a copyright violation. Students who post these documents outside of Wake Tech are subject to academic and possible legal actions.

## 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.

Both problems consist of a program that you create from scratch that meets the problem specifications.

## Problems

### Problem 1

As an IT technician, one of your tasks involves analyzing the log files of servers and workstations to identify any issues. These log files contain messages that are categorized into one of the four categories: CRITICAL, ERROR, WARNING, INFO.

In this assignment, you will write a program that counts the number of log messages in each of these categories from a file named program.log.

#### Log File Format

Each line in the log file is structured as follows:

<Date> <Time> - <Error Category> <Log Message>

Where:

* Date: The date in the format YYYY-MM-DD
* Time: The time in the format HH:MM:SS.ssssss
* Error Category: One of CRITICAL, ERROR, WARNING, INFO
* Log Message: The specific message corresponding to the error category (e.g. ThresholdAlert, SysCheckOK, etc.)

The log file you will be analyzing is called program.log. Here is a sample of what that log file could look like:

2023-07-16 17:29:09.053831 - WARNING ThresholdAlert

2023-07-16 17:29:09.089397 - ERROR LightFault

2023-07-16 17:29:09.122305 - CRITICAL FileSysFull

2023-07-16 17:29:09.147270 - CRITICAL CPUPLUS150

2023-07-16 17:29:09.177157 - INFO SysCheckOK

2023-07-16 17:29:09.212638 - CRITICAL UnauthorizedDetected

2023-07-16 17:29:09.222653 - WARNING CPUOVER80

2023-07-16 17:29:09.245627 - CRITICAL CPUPLUS150

2023-07-16 17:29:09.272556 - WARNING WatchdogExpired

A longer sample is available on Blackboard.

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

* Initialize five counters to zero. These counters will be used to count the number of log messages in each category. There should be one counter for each category (CRITICAL, ERROR, WARNING, INFO) and an additional counter for messages that do not fit into these categories.
* Attempt to open a file called program.log. If the file is not present, the program prints "File program.log not found" and terminates.
* If the file is present, your program should read each line of the file using a for loop.
  + For each line, split the line into its five components (date, time, dash, error category, log message) using the split() function. This will create a list with five elements.
  + Analyze the error category for each line using an if-elif-else construct and increment the corresponding counter. For example, if the error category is "CRITICAL", increment the critical error counter by one.
  + After all lines have been processed, display the count of log messages for each category, including the count of unknown error types.

Note: You should use a try-except block to handle the error when the file does not exist.

Sample Output (using what was posted on Blackboard):

CRITICAL: 46

ERROR: 68

WARNING: 56

INFO: 55

UNKNOWN: 1

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

### Problem 2

Trish runs a small business called Trish's Swap Shop, where she sells a variety of items including books, DVDs, and games. To help manage her business more efficiently, she wants to generate a daily sales report that categorizes and totals her sales for each category of items sold.

You will write a program that creates this daily sales report.

The source of data for your program will be a text file called "sales.txt". This file will contain the details of each sale made during a day. Each line in the file represents one sale and contains the following fields separated by a comma:

* Date and time of the sale (in the format YYYY-MM-DD HH:MM:SS)
* Category of the item sold (Book, DVD, or Game)
* Price of the item sold (in dollars)

Here's an example of what a line in "sales.txt" could look like:

2023-07-16 10:15:30, Book, 15.99

A sample data file is available on Blackboard.

Create a file named **Lab08P2.py** and write a program that does the following:

* Tries to open "sales.txt". If the file is not present, the program should print "File 'sales.txt' not found" and terminate.
* If the file is present, read each line of the file using a loop. Split each line into its components (date/time, category, and price) using the split() function. Note that you will need to split across commas to process the data in this file.
* The program should maintain a running total of the sales for each category.
* After all the data from sales.txt has been read, the program should write the totals for each category to a new file called "daily\_report.txt". This file should have one line for each category, like this:

Sample Content of daily\_report.txt:

Books: $107.80  
DVDs: $27.76  
Games: $78.67

Please make sure that your program can handle cases where the sales.txt file contains extra spaces before or after the commas, or at the beginning or end of lines.

The program should also handle situations where the price on a line could not be converted to a float. In that case, the program should print an error message to the console to inform the user there is invalid data, and tell the user where the error occurred:

Error on line 12: Could not convert "sixty-seven cents" to price format

Error on line 22: Could not convert "4.04s" to price format

Error on line 36: Could not convert "4 dollars" to price format

Note that only errors are printed to the console. The totals are written to the daily\_report.txt file and are NOT output to the console.

You should be using try-except blocks for handling two different potential situations:

* The sales.txt file may not exist.
* The text in the pricing field may not be valid for conversion to a float.

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

## Grading Rubric

### Grading rubric for Problem 1 (45 points)

* Program has a well-formatted and correct header [5 points]
* Program does execute correctly and produces correct results [40 points]

### Grading rubric for Problem 2 (55 points)

* Program has a well-formatted and correct header [5 points]
* Program does execute correctly and produces correct results [50 points]