Name: Nicholas Smith
Date: 5/13/2024
Course: IT FDN 110 B

Assignment 5 – Advanced collections and error handling

Introduction

The purpose of this article is to cover the steps that were taken in developing Assignment05.py, which demonstrates the ability to collect data from files, use dictionaries and lists, save data to files, and implement advanced error handling techniques.

Creating the Program

There were four main components to developing the program for this assignment.

1) First I started by defining constants and variables. The constants and variables were largely similar to assignment 04, with the exception of student_data being a dictionary rather than a list.
2) Next I defined a series of cases as part of the main body of the code, which called on functions that I define in later steps.
3) I defined the four functions that I initiated in step two, one to add students to the registration, one to display current registrations, one to save all records to a csv file, and one to quit the program. I imported the csv library at the top of the code in order use the special DictReader and DictWriter functions for reading and writing data to the csv file. This also enabled references to elements in the dictionary by name as opposed to by index as we did in previous exercises with lists.
4) I implemented some error handling code to handle error cases for user inputs, and for loading and reading data from the csv files.

I got particularly hung up on one part of the code related to the csv.DictWriter function. I could not figure out why when I was loading data back into the csv file it was including white space in between my records. After some investigating I determined that there was a default command '\r' which was adding an extra return after each record.

Summary

After running the program in the command line, I verified that the program was running as intended. It successfully provided a menu of options for the user and performed conditional operations based on the user's input. Additionally, the program generated a discoverable .csv file with the desired text. The program was also able to tell the user what data was previously stored in that .csv file upon running the program and add additional records if the user wished to do so.

The inclusion of csv specific commands makes the code easier to read, and the code handles a lot of the data translation for us with less syntax. The error handling helps with directing the user to make right decisions in the event of an unsuccessful command.

In summary, we started by documenting our purpose, demonstrated how to read and load data to and from a file, and how to use lists and dictionaries for the purposes of storing and collecting data.