Deric Fetters

5/23/2024

IT FDN 110 A

Assignment 04

Collections of data

# Introduction

Module 4 most notably introduces data collections, including tuples, lists, sets and dictionaries, and methods for reading data from and writing data to files. It also presents the benefits of using loops with data collections to manage data.

# Declared and initialized Lists

After revising the header of the starter file, the first action was declaring and initializing to empty set lists “student\_data” and “students”. This task was straightforward, only requiring the use of square brackets. See Figure 1.

A black screen with white text

Description automatically generated

***Figure 1: Assignment04.py list declaration and initialization***

# Reading data from file

One of the first requirements of assignment 4 was to read data from “Enrollments.csv”. To accomplish this, the file was opened. A For loop, utilizing the split function to create elements from the characters between commas and the strip function to remove the carriage return at the end of the line, was used to set values of the elements of “student\_data” to the values of the elements from the file. The readlines() function was used with the loop to read all the lines of the file. The append() function was used to add each three element “student\_data” list to the “students” list. Once each line was read and entered into the “students” list, the “Enrolments.csv” was closed. One issue I had with my initial code was extra spaces in some of the elements when printed to the screen. I looked for extra spaces in the “Enrollments.csv” file but did not find any. I utilized the strip() function on each “student\_data” element call to remove the spaces. See Figure 2.

A computer screen with text

Description automatically generated

***Figure 2: Assignment04.py Reading data from file***

# Storing data collected into lists

Providing for the input of new student names into lists was another requirement for assignment 4. The stater file utilized input statements and the “csv\_data” text string to store data from the user. The code was updated by removing the “csv\_data” string and setting elements of “student\_data” equal to “student\_first\_name”, “student\_last\_name”and “course\_name” while also including a “\n” escape character. “Student\_data” was then appended to the list “Students”. See figure 3.

A computer screen shot of a program code

Description automatically generated

***Figure 3: Assignment04.py Input data to list***

# Printing data from lists in proper format

Option 2 from the program menu, “Show current data.”, requires that the data be shown on the screen. The starter file utilized the print() function and “csv\_data”. Since in the starter file “csv\_data” was a text string and formatted as data was read from the file or entered by the user, the print statement in the starter file was very short and straightforward. To print the lists, “csv\_data” was set to the elements of a row of “Students”, formatted with proper spacing and commas and then printed to the screen. A For loop was used to repeat this for each row of the “Students” list. One issue I had with this section was at first I tried to simply print each row of “students” by passing “Students” in the print function. The rows printed, but with extra single parenthesis. I correct this by utilizing the formatted “csv\_data” string previously mention. See Figure 4.

A computer screen shot of text

Description automatically generated

***Figure 4: Assignment04.py Print data to screen***

# Writing data to file

Option 3 from the menu indicated to write the student data to the “Enrollments.csv” file. Since “csv\_data” was formatted as the data was read from the file or entered by the user, the code was very straight forward. The write() function was used to write “csv\_data” to the .csv file. In order to write to the .csv file from “students”, “csv\_data” was set equal to the elements of “students” as had been described in the “Printing data from lists in proper format” section above. The write() function was then used to write “csv\_data” to the next line in the .csv file. A For loop was used to repeat this for each row “Students”. This section of code was very similar to code used for Option 2 of the menu making it very straightforward. See figure 5.

# Prints data saved to file to screen

Option 3 also required that the data written to the file be written to the screen. To accomplish this, I copied and pasted the code from option 2 and updated the message to “You have registered:”. After looking further, I believe I could have just added a print statement for the message to the user and a print statement to the code for writing the data to the file and could have accomplished both in the same loop.

# Summary

In Module 4, I learned about data collections, including tuples, lists, sets and dictionaries, and methods for reading data from and writing data to files. I also learned the benefits of using loops with data collections to manage data.