Andrea Martinez

November 13, 2024

Python 100

Assignment 05

Assignment 05 – Programming with Python Dictionaries

Introduction

In module 04 of this course, one of the main objectives was to test out the PyCharm debugger tool to help the developer debug code. In this document, I'll briefly describe my initial thoughts about debugging software. In addition, I'll describe the steps taken to generate a program with conditional logic using while-loops and if-statements to create a programming menu from user-defined inputs. I'll also demonstrate the differences between module 3 and 4, which include the applications of for-loops and reading files. Finally, I will review the steps needed to append data and store multi-dimensional data under a csv file.

Using the Debugging Tool

Coming with Matlab experience, I was disappointed that PyCharm did not have a way to show the values of the distinct variables in a code after the code was ran. There was no way for me to see what was stored in the variables or constants created. However, after learning this debugger tool, I am able to put a breakpoint, which is a type of special markers that suspend program execution at a specific points in the code (https://www.jetbrains.com/help/pycharm/using-breakpoints.html, 2024). This allows me to stop the code at an area of interest to evaluate what my code is doing in the background and observe how the data is being stored in the variables that I initialized at the start of the code like shown in Figure 1. Using a single marker or multiple markers can be very powerful to bring special awareness to the developer

Figure 1: PyCharm Debugger Interface

Reading a File

Starting this assignment was very straightforward because it used the code that I wrote during assignment 03 as the foundation. This was true because there were only some minor differences between these two assignments such as the addition of for-loops and use of collections.

After the initialization of the variables and constants, I proceeded to open the csv file defined under constant FILE_NAME. I opened it in read mode because the file already existed after I saved it in the same folder structure. If the csv file did not exist, the code would have thrown the following error since it couldn't find the file in the specified location: "FileNotFoundError: [Errno 2] No such file or directory: 'Enrollments.csv'." The file did not have to have any specific data saved in there, but I still added 3 initial student names in the file so that I could see the "students" list get created in the for-loop. Writing the for-

loop was very interesting because it taught me how to parse through a set of data, or in this case, rows of data inside of a file. I liked how Python has many methods to perform different functions such as the readlines() method or the strip() and split(). I learned that readlines() returns all the lines in a file saved as a list of strings. I also learned that the strip() method removes any special characters and whitespaces, and the split() method splits a string into a list by using a separator, like any special character, or in my case, a comma ",". Finally, I use a new list variable with the append() method to collect all the strings together in a list. At this stage of the program, the contents inside of the existing csv file should be saved inside of a list of strings.

Writing Conditional Logic Statements

The while loop I wrote did not have a counter variable that incremented in value during each iteration, instead, I added the "True" constant, which represents the Boolean value for true. Since the while loop was set to always True, this meant that the loop would continue to run endlessly until the user would tell it to stop with an exit condition. Therefore, I added a break statement to finish the loop when the user was done using the program by selecting option 4 in the programming menu I will mention in the next section.

Saving Data in List

The first thing I did inside of the while loop was displaying the Menu with the print() function, prompting the user to make a selection. I stored the user selection under the variable 'menu_choice' as a string to be later used in the if-statement. Now that the user made a selection, I nested an if-statement inside of the while loop with a series of choices. If choice #1 was selected from the menu, I asked the user to register a student for a class with three input functions learned in module 01. I then made a list with the information entered by the user, and subsequently appended the values to a list called 'student'. I chose to append that data with the append() function in case the next user selection was also a '1', such that I had all the registered students saved under the same list. I could have also chosen to use the "+=" operator to collect the data in the same way as append(). Then, if choice #2 was selected, I displayed the current students registered for the class. So, for example, if I selected choice #1 three different times to register three different students, their names and classes would get printed with choice #2. For this purpose, I had to create a for loop such that the code would individually parse through each row in the list and print all the rows of students. If choice #3 was selected, I opened a csv file in write mode and stored all the current students registered for the class into the Enrrollment.csv file. To do this step, I implemented a for loop that would individually add all the rows available in the 'students' list until I closed the file. A small printed statement would also get output with choice #3, informing the user that the data is being saved to a csv file. If selection #4 was made, no other transaction would be made, and I would just end the program with the break command. Now, even though, this was not a part of the script requirement, I added an "else" statement for any other answer that was not either one of the choices in the menu (1-4). This else statement printed an error message, indicating the selection was not valid and a valid answer must be entered again.

Error Handling

Testing the Program

As a developer, I must test my code to make sure that it is doing what it is supposed to by meeting all the requirements set by the instructor. So, while I was testing the code, I made sure to use the debugger tool, by incrementally stepping over lines of code to see what values were assigned to the specific variables I initialized at the start of the code. I liked adding breakpoints inside of for-loops in particular because it allowed me to evaluate each iteration as the 'student' counter incremented in value. In Figure 2 below, the left-hand yellow square shows the button I used to step into the next iteration of the for-loop, and the yellow square on the right-hand side shows the value that is assigned to csv_data at each of the steps of the loop. After making sure that the code met all the requirements from the instructor, I tested the code in the terminal to ensure that the program would also run without the IDE.

Figure 2: Additional aids from the PyCharm Debugger

Source Control in Github

Summary

In conclusion, the planning, writing, and execution of this code taught me how to properly use lists, for-loops, and the PyCharm debugger. I improved my skills using while-loops, if-statements, and writing data to a csv file. I enjoyed learning the multi-dimensional aspect of saving data to a file and using iterations to parse through sets of data. I can now say that I feel pretty comfortable writing a program that interacts with the user with the input() function. I continued to apply good bookkeeping practices in coding such as using Booleans, and initializing variables, adding comments before a line of code, creating multi-line commands, and formatting strings.