Erickson Smith
November 17, 2020
IT FDN 110 A Au 20: Foundations of Programming: Python
Assignment 05

To Do List Script

Introduction

In this week's Assignment, Assignment05. I was asked to take an existing code and modify it so that it would be fully functional. The script manages the data from a text file, adds the data to a list, and returns the data to the file to be saved. This data is saved into a "Task" and "Priority" pair using the new function - Dictionaries.

Starting the code

To start the script I needed to download and create a new project in Pycharm. After that the first step was to adjust the change log of the code to show (Who, What, and When) the code was modified. (Figure 01.)

```
# ------------------------------------------------------------------------ #
# Title: Assignment 05
# Description: Working with Dictionaries and Files
#              When the program starts, load each "row" of data
#              in "ToDoToDoList.txt" into a python Dictionary.
#              Add the each dictionary "row" to a python list "table"
# ChangeLog (Who,When,What):
# RRoot,1.1.2030,Created started script
# <YOUR NAME HERE>,<DATE>,Added code to complete assignment 5
# ------------------------------------------------------------------------ #
```

Figure 01: Change Log.

The next step was to look at the set of declared variables so that I could build the code. (Figure 02.)

```
# -- Data -- #
# declare variables and constants
objFile = "ToDoList.txt"   # An object that represents a file
strData = ""  # A row of text data from the file
dicRow = {}    # A row of data separated into elements of a dictionary {Task,Priority}
lstTable = []  # A list that acts as a 'table' of rows
strMenu = ""   # A menu of user options
strChoice = "" # A Capture the user option selection
```

Figure 02: Declared variables.

With the change log set and the variables declared I was ready to start building the code.

Building the code

In order for this week's code to begin to come together I had to begin with step one, load the file and allow the code to read from memory whenever prompted. (Figure 03.)

```
# -- Processing -- #
# Step 1 - When the program starts, load the any data you have
# in a text file called ToDoList.txt into a python list of dictionaries rows (like Lab 5-2)
# TODO: Add Code Here

objFile = open(objFileName, 'r')
for row in lstTable:
    lstRow = row.split(',')
    dicRow = {'Task': lstRow[0], 'Priority': lstRow[1].strip()}
    lstTable.append(dicRow)
objFile.close()
```

Figure 03: Opening the file.

After that was step two, providing the user with a menu of options to choose from. (Figure 04.)

```
# -- Input/Output -- #
# Step 2 - Display a menu of choices to the user
while (True):
    print("""
    Menu of Options
    1) Show current data
    2) Add a new item.
    3) Remove an existing item.
    4) Save Data to File
    5) Exit Program
    """)
    strChoice = str(input("Which option would you like to perform? [1 to 5] - "))
```

Figure 04: Menu of Options for the user.

Step 3, Show the current data. This is the first if statement. If the user input (1) from the menu of options one of two things would happen. First if there was no data in the file it would print (“No data found.”) Second if there was data in the file it would print it out for the user to read. (Figure 05.)

```
# Step 3 - Show the current items in the table
if (strChoice.strip() == '1'):
    # TODO: Add Code Here
    if not lstTable:
        print('No data found.')
    else:
        for row in lstTable:
            eleOne = row['Task']
            eleTwo = row['Priority']
            print('Task: ', eleOne, ' | Priority: ', eleTwo, sep='')
    continue
```

Figure 05: Step 3 Displaying current Data.

Step 4, Add a new item. If the user input (2) into the menu of options. They would be prompted to input a “Task” and that task's “Priority”. The code would then put it into a dictionary and append it to a list of dictionaries.(Figure 06.)

```
# Step 4 - Add a new item to the list/Table
elif (strChoice.strip() == '2'):
    # TODO: Add Code Here
    new_task = input('Enter new Task: ')
    new_priority = input('What Priority level? (High, Med, Low): ')
    dicRow = {'Task': new_task, 'Priority': new_priority}
    lstTable.append(dicRow)
    continue
```

Figure 06: Appending the input data to the file.

The next step was step 5, removing data from the file. As the user finishes tasks on the "To Do" list. They needed to be able to remove them. This step would ask the user to input the "Task" they wanted to remove then it would loop through the file to see if the task was in the file. If it was, it would be removed and the code would print out "Task removed". (Figure 07.)

```
# Step 5 - Remove a new item from the list/Table
elif (strChoice.strip() == '3'):
    # TODO: Add Code Here
    strTask = input('Which Task would you like to remove? ')
    for row in lstTable:
        if row['Task'] == strTask:
            lstTable.remove(row)
            print('Task removed')
            print(lstTable)
    continue
```

Figure 07: Removing data from the file.

Step 6 was to save the input data to the file. If the user input (4) from the menu the data collected up to that point would be written into the file and saved to the harddrive. (Figure 08.)

```
# Step 6 - Save tasks to the ToDoList.txt file
elif (strChoice.strip() == '4'):
    # TODO: Add Code Here
    objFile = open(objFileName, "w")
    for row in lstTable:
        objFile.write(row["Task"] + "," + row["Priority"] + "\n")
    objFile.close()
    print('Saved')
    continue
```

Figure 08: Saving data to the file.

The final step was to exit out of the program. (Figure 09.)

```
# Step 7 - Exit program
elif (strChoice.strip() == '5'):
    # TODO: Add Code Here
    input('Press the Enter key to exit')
    break  # and Exit the program
```

Figure 09: Exiting the program.

Summary

In this week's assignment I learned to work with dictionaries and how they can clean up your information for .txt management. This was also my first look at taking someone else's code and manipulating it so that it works for what I need. I feel that this is a real look into the functions a dev deals with on a regular basis.