Matt Dunn
August 7, 2023
IT FDN 110 A
Assignment 05
https://github.com/madunn5/IntroToProg-Python

Dictionaries – Creating a To-Do List

Introduction

In this paper I will be go over the steps I took to finish the starter code that was provided by Professor Root in order to create a program that functions list a To-Do List. The program asks the user to enter a task and a priority (high, medium, or low), and then save the inputs to a text file. In addition, the program can also take in more tasks, remove tasks, display the current list, and finally save and exit the program so that a new text file is saved based on the most recent interaction. This was another great exercise in loops, if statements, and taking inputs from the user and saving them somewhere else for use later. In addition, I was able to test out dictionaries for the first time in this class, as well as improve upon my formatting abilities to make the code presentable and easy read, as well as the output that the user would see.

Creating & Testing the Program

The first step I took when creating the code was to look over the starter code provided by Professor Root. It was very clearly outlined with the expectations for this program, and I decided to work on it one section at time, in order to make sure that everything was working as expected.

The first section of the code acts a log of information about what the code should accomplish, as well as any changes made to the code so that anyone can come in and get an idea of what is going on. Professor Root started the log, so I added my name, the date, and a brief summary of what I did. In a professional setting it would probably be better to be more detailed about the steps, but since the backbone of this code was already created for me, I decided to just be more concise with my log description.

```
1   # ------------------------------------------------------------------------ #
2   # Title: Assignment 05 - Creating a To Do List
3   # Description: Working with Dictionaries and Files
4   #              When the program starts, load each "row" of data
5   #              in "ToDoToDoList.txt" into a python Dictionary.
6   #              Add each dictionary "row" to a python list "table"
7   # ChangeLog (Who,When,What):
8   # RRoot,1.1.2030,Created started script
9   # MDunn,8.7.2023,Updated code in order to finish Assignment 05
10  # ------------------------------------------------------------------------ #
```

Figure 1. Top of the code where detail can be added about what the code should accomplish and any changes that have been made to it.

The next part of the code is the declaration of variables. Professor Root had already outlined a couple of variables that he expected I might need, many of which I kept. I did create a new variable called “FileName” to represent the name of the of text file, as well as move the “strMenu” input to this section

since that is not going to change during the actual coding and I thought that it looked better in this section so as not to distract in the actual coding. Now I can just reference “strMenu” instead of having that all written out (see Figure 4).

```
# -- Data -- #
# declare variables and constants
FileName = 'ToDoList.txt'  # Text file name
objFile = ''  # An object that represents a file
RemoveTask = ''  # 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 = """
    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
    """  # A menu of user options. Placing at top of code so I can easily update w/o having to change my processing code
strChoice = ""  # A Capture the user option selection
```

Figure 2. Declaring variables in their own section to have an overview of what to expect down below in the actual code.

Next, I need to load up my text file in order to get my data into my lstTable variable. I do this by using the open() to open the text file, and then use a for loop to split out the rows of the text file to create my lstTable variable. Here I used a try and except function, which basically says to try the first part of function, and if it fails then it will run whatever is in the try function section. My thought here was that it’s possible that the text file could be empty depending on when the user is running the program, so in that case it would fail to load if the text file doesn’t exist. In that case it will print a message to the use saying that the text file currently has no items in it.

```
# -- Processing -- #
# Step 1 - When the program starts, load the data you have
# in a text file called ToDoList.txt into a python list of dictionaries rows (like Lab 5-2)
# the try function will try to open the file if it exists. If it doesn't exist it will inform the user
# that it currently doesn't exist
try:
    objFile = open(FileName, 'r')
    for row in objFile:
        lstRow = row.split(',')  # Returns a list!
        dicRow = {'task': lstRow[0], 'priority': lstRow[1].strip()}
        lstTable += [dicRow]
    objFile.close()
except:
    print('There are currently no items on your To Do List. Please add some tasks!')
```

Figure 3. Populating the lstTable variable by trying the first set of code. If it fails than the except function will print a statement to the user.

Now that I’ve finished the data and processing pieces of the code that Professor Root provided me with, it’s time to move on to the input and output section of the code, which is the meat and potatoes of the code. This first part of the code sets while loop to True, so basically this will run at the beginning of each loop of the code until the code is told to exit. The first part prints the menu that the user will see before

choosing their input, and this is where my strMenu variable comes in handy and makes the code look a lot cleaner and less busy.

```
# -- Input/Output -- #
# Step 2 - Display a menu of choices to the user
while True:
    print(strMenu)
    strChoice = str(input('Which option would you like to perform? [1 to 5] - '))
    print()  # adding a new line for looks
```

Figure 4. Creating a menu for the user to choose their option.

The remaining parts of the code are what to do for each option that is possible, numbers 1-5. The first option should return whatever is currently in the lstTable. This is relatively easy to do and requires an if() statement along with a for loop that loops through lstTable and prints the current tasks and priorities. The continue option at the end just prompts the code to go back to the menu option once the code has executed. The "row['task']" and "row['priority']" tells the code to look to that part of the dictionary for each row of lstTable, which will be a consistent theme throughout the code.

```
    # Step 3 - Show the current items in the table
    if strChoice.strip() == '1':
        print('Your To Do List is as follows:')
        for row in lstTable:
            print('Task:', row['task'].strip(), '| Priority:', row['priority'].strip())
        continue
```

Figure 5. Code for option 1 which displays the current data in the list. This should include both previously saved data and anything that has been entered into the list in the current session.

Option 2 takes two inputs from the user, task and priority. These two tasks should be appended to lstTable before returning a statement to the user saying that the task has been added, but not saved yet (more on this later). Since we are appending dictionaries to a list (lstTable is set up as a list at the top of the code in the Data section), I will use {} to outline that the inputs are "task" and "priority", respectively. This will allow for easy data manipulation throughout the code (as already seen in Figure 5).

```
# Step 4 - Add a new item to the list/Table
elif strChoice.strip() == '2':
    task = input('Please enter a task: ')
    priority = input('Please choose from Low, Medium or High Priority: ')
    lstTable.append({'task': task, 'priority': priority})
    print(task, 'has been added to the list, but not saved yet!')
    continue
```

Figure 6. Code for option 2 that asks for two inputs from the user before appending them to lstTable. Note, the data has been added but not saved yet!

Option 3 allows the user to remove a task from their list. This proved to be the most challenging part of the code for me, but I was able to work it out. For this option, the code will loop through lstTable and see if the row['task'] matches what the user has input for what they want removed. I set this variable as RemoveTask so that it is clear what is going on in the code. Once the task is found, the code will print that the task has been removed but has not been saved still. Using the .lower() function allows for the code to not have to be key sensitive to whatever the user inputs (though it's still vulnerable to typos!)

ow, the tricky part of this was how to tell the user if the task isn't found at all in their list. I could put the print statement within the for loop, but I noticed that this causes the printed statement to print as many times as there are rows in lstTable, and I only want the statement to print once. After some pondering, I decided that the best way to do this would be to use a Boolean flag set to false outside of the loop, and then if the for loop finds the task it will set the Boolean flag to be true. Then, in a separate if() statement if the flag is true it will pass (since a statement has already been printed), otherwise if it's false then it will print that the task cannot be found.

```
# Step 5 - Remove a new item from the list/Table
elif strChoice.strip() == '3':
    RemoveTask = input('What task would you like to remove?: ')
    found = False
    for task in lstTable:
        if task['task'].lower() == RemoveTask.lower():
            lstTable.remove(task)
            print(RemoveTask, 'has been removed from list, but has not been saved yet!')
            found = True
            break
    # if the task is not found in the lstTable then it will remain false, and the code will print a single statement
    # instead of multiple statements based on the len(lstTable)
    if found:
        pass
    else:
        print('Task not found in the To Do List.')
    continue
```

Figure 7. Option 3 removes a task based on the user's input. If the task is not found in the code, then it will print that the task cannot be found.

Option 4 writes all the data in lstTable to a text file, thus saving the data to the text file. Like a lot of the code above, this is also taken care of by using a for() loop and writing each task and priority from each in lstTable to the text file. I include \n at the end of each line so that the text file saves each record on a new line. Opening the objFile with "w" allows for the file overwrite each time, thus saving the current data in lstTable each time.

```
# Step 6 - Save tasks to the ToDoToDoList.txt file
elif strChoice.strip() == '4':
    objFile = open('ToDoList.txt', 'w')
    for row in lstTable:
        objFile.write(row['task'] + ',' + row['priority'] + '\n')
    print('Data Saved!')
    continue
```

Figure 8. Option 4 saves the data to the text file by writing the data from lstTable.

Finally, Option 5 will close the text file and exit the code. I also included a printed statement to the user know that the file has now closed.

```
# Step 7 - Exit program
elif strChoice.strip() == '5':
    objFile.close()
    print('File is now closed. Goodbye!')
    break  # and Exit the program
```

Figure 9. Option 5 closes the text file and exits the code.

Now that all the code has been written, it's time to test it! First, I ran it in PyCharm and wanted to add two tasks, delete one, and then save the file. Below are some screenshots from that test run.

```
/Users/Matt/Assignment05/bin/python /Users/Matt/Documents/_PythonClass/Assignment05/Assignment05_Starter.py
There are currently no items on your To Do List. Please add some tasks!

    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

Which option would you like to perform? [1 to 5] - 2

Please enter a task: Do Homework
Please choose from Low, Medium or High Priority: High
Do Homework has been added to the list, but not saved yet!

    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

Which option would you like to perform? [1 to 5] - 2

Please enter a task: Get a car wash
Please choose from Low, Medium or High Priority: Low
Get a car wash has been added to the list, but not saved yet!

    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
```

Figure 10. A text file did not currently exist at the time of the test, so the code prompts be to add tasks. I add in two different tasks with different priorities by choosing option 2 for each.

```
Which option would you like to perform? [1 to 5] - 1

Your To Do List is as follows:
Task: Do Homework | Priority: High
Task: Get a car wash | Priority: Low

    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

Which option would you like to perform? [1 to 5] - 3

What task would you like to remove?: Get a car wash
Get a car wash has been removed from list, but has not been saved yet!

    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

Which option would you like to perform? [1 to 5] - 1

Your To Do List is as follows:
Task: Do Homework | Priority: High
```

Figure 11. I choose option 1 in order to view my current list. After some thought, I decide to delete Get a Car Wash and then return my current list.

```
    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

Which option would you like to perform? [1 to 5] - 4

Data Saved!

    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

Which option would you like to perform? [1 to 5] - 5

File is now closed. Goodbye!

Process finished with exit code 0
```

Figure 12. That's all I want to add for now, so I choose option 4 to save my data before choosing option 5 to close the file.

Figure 13. Text file with my current to-do list after the test run.

Running the Program in the Terminal

The last thing to test for this assignment was to the run the program in the Mac Terminal.

```
    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

Which option would you like to perform? [1 to 5] - 1

Your To Do List is as follows:
Task: Do Homework | Priority: High

    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

Which option would you like to perform? [1 to 5] - 2

Please enter a task: Buy more dog food
Please choose from Low, Medium or High Priority: Medium
Buy more dog food has been added to the list, but not saved yet!

    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

Which option would you like to perform? [1 to 5] - 2

Please enter a task: Go clothes shopping
Please choose from Low, Medium or High Priority: Low
Go clothes shopping has been added to the list, but not saved yet!
```

Figure 14. Running the code in the Mac Terminal by navigating to the folder first and then running the program. I display current data and then add in two more tasks for my to-do list.

```
    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

Which option would you like to perform? [1 to 5] - 3

What task would you like to remove?: Do Homework
Do Homework has been removed from list, but has not been saved yet!

    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

Which option would you like to perform? [1 to 5] - 1

Your To Do List is as follows:
Task: Buy more dog food | Priority: Medium
Task: Go clothes shopping | Priority: Low

    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
```

Figure 15. Now that I'm almost done with my homework, I decide to remove it from the list and display the current data.

```
Which option would you like to perform? [1 to 5] - 4

Data Saved!

    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

Which option would you like to perform? [1 to 5] - 5

File is now closed. Goodbye!
```

Figure 16. I save my data and exit the file.

Figure 17. Updated text file with the inputs from the Terminal run.

Summary

This assignment was another great opportunity to demonstrate what I've learned so far in this class. Dictionaries are the most difficult thing I've learned so far, but I see their versatility in this week's assignment. Being able to create keys and values is going to be important in my future endeavors, and I'm glad to see that I'm already laying the foundation for that! I'm excited to continue improving with next week's assignment.