Barbara Spadavecchia
February 15, 2022
IT FTN 110 A
Assignment05

Using Lists and Dictionaries to Create Scripts

Introduction

This paper discusses the process of creating a "ToDo" program that used Dictionaries and Lists. As part of this process, this paper also goes over some ways to improve scripts by using techniques that include: the separation of concerns, structured error handling, script templates, and functions. It also lists benefits of GitHub.

Lists

Lists hold data in computer memory which will be lost when an application is closed unless it is stored in a file. The data is usually stored in a text file. Lists are a simple way to hold a collection of objects and have built-in functions to help you work with those objects. Lists are sequences that are mutable and can be modified. Lists use brackets[]

Dictionaries

Dictionaries work with pairs of data. They use a key instead of an index to look up values. Dictionaries use {} braces to show that you want a variable to be a dictionary. They have the advantage of having keys that are very descriptive.

```
# ------------------------------------------------ #
# Title: Lab 5-1
# Description: Writing and Reading Data from a file
# ChangeLog (Who,When,What):
# RRoot,1.1.2030,Created started script
# BSpadavecchia,02-11-2022, Added read/write to file code
# ------------------------------------------------ #

# Declare my variables
strChoice = '' # User input
lstRow = []  # list of data
strFile = 'HomeInventory.txt'  # data storage file
objFile = None  # file handle

# Get user Input
while(True):
    print("Write or Read file data, then type 'Exit' to quit!")
    strChoice = input("Choose to [W]rite or [R]ead data: ")

    # Process the data
    if (strChoice.lower() == 'exit'): break
```

```
elif (strChoice.lower() == 'w'):
    # List to File
    objFile = open(strFile, "w")
    lstRow = ["Item", "Value"]
    objFile.write(lstRow[0] + ',' + lstRow[1] + "\n")
    lstRow = ["Lamp", "$30"]
    objFile.write(lstRow[0] + ',' + lstRow[1] + "\n")
    lstRow = ["End Table","$60"]
    objFile.write(lstRow[0] + ',' + lstRow[1] + "\n")
    objFile.close()
elif (strChoice.lower() == "r"):
    #File to list
    objFile = open(strFile,"r")
    for row in objFile:
        lstRow = row.split(",")
        print(lstRow[0]+ '|' + lstRow[1].strip())
    objFile.close()

else:
    print('Please choose either W or R!')
```

Figure 1 Lab 5-1 Working with Files and Lists

Lab 5-1 gets user input on whether to write or read file data or exit. If the choice selected is "w" the data will write into "HomeInventory.txt" file. The script is divided into sections that declare the variables, get user input, process the data which either writes from a list (lstRow) or reads from the file to the list. This lab gave practice in writing to a file from a list and in reading from a file and loading it into a list. This lab also used "strip" to remove spaces.

```
C:\Users\bspad>python.exe "C:\_PythonClass\Mod5\Module05 - Lists and Dictionaries\Lab5-1.py"
Write or Read file data, then type 'Exit' to quit!
Choose to [W]rite or [R]ead data: r
Lamp|$30
End Table|$60
Write or Read file data, then type 'Exit' to quit!
Choose to [W]rite or [R]ead data:
```

Figure 2 Lab 5-1 Output running from command prompt

https://www.youtube.com/watch?v=m0o0CkYsDzI This video talks about reading and writing to text files in Python. This video explained how in Python a variable is assigned to the text file. It used the example of newfile = open("newfile.txt", "w"). This example had a clear way of explaining the way that reading and writing is done to text files by only changing one part of the syntax at a time. It also went over another explanation of the carriage returns \n" and the need to close files and not leave them open. The command readline showed how just one line at a time could be read and how to choose which line.

Lab 5-2 Working with a Table of Dictionaries

In this lab, lists were converted to dictionaries. To do the conversion, the [] had to be changed to {} and a key for "Item" and a key for "Value" needed to be added. The code works by first declaring the variables. The list of hardcoded values is written into the file from the dictionary. When the script is run it either writes data paired to the key or reads data that is paired with a key. The script when it ran, displayed both the key "item" along with the

item name. It also displayed the key “value” and the amount. It would be nice to display the results without the dictionary braces.

```
# ------------------------------------------------- #
# Title: Listing 9
# Description: Writing and Reading Data from a file
# ChangeLog (Who,When,What):
# RRoot,1.1.2030,Created started script
# BSpadavecchia,02-12-2022,Changed rows from lists to dictionaries
# ------------------------------------------------- #

# Declare my variables
strChoice = '' # User input
dicRow = {}  # list of data
strFile = 'HomeInventory.txt'  # data storage file
objFile = None  # file handle

# Get user Input
while(True):
    print("Write or Read file data, then type 'Exit' to quit!")
    strChoice = input("Choose to [W]rite or [R]ead data: ")

    # Process the data
    if (strChoice.lower() == 'exit'): break
    elif (strChoice.lower() == 'w'):
        # List to File
        objFile = open(strFile, "w")
        dicRow = {"item": "Lamp","value": "$30"}
        objFile.write(dicRow["item"] + ',' + dicRow["value"] + '\n')
        dicRow = {"item": "End Table","value": "$60"}
        objFile.write(dicRow["item"] + ',' + dicRow["value"] + '\n')
        objFile.close()
    elif (strChoice.lower() == 'r'):
        # File to List
        objFile = open(strFile, "r")
        for row in objFile:
            lstRow = row.split(",") # Returns a list!
            dicRow = {"item": lstRow[0], "value": lstRow[1].strip}
            print(dicRow)
        objFile.close()
    else:
        print('Please choose either W or R!')
```

Figure 3 Lab 5-2 Working with a Table of Dictionaries (using a dictionary instead of a row)

```
C:\Users\bspad>python.exe "C:\_PythonClass\Mod5\Module05 - Lists and Dictionaries\Lab5-2.py"
Write or Read file data, then type 'Exit' to quit!
Choose to [W]rite or [R]ead data: w
Write or Read file data, then type 'Exit' to quit!
Choose to [W]rite or [R]ead data: r
{'item': 'Lamp', 'value': '$30\n'}
{'item': 'End Table', 'value': '$60\n'}
Write or Read file data, then type 'Exit' to quit!
Choose to [W]rite or [R]ead data: exit
```

Figure 4 Lab 5-2 Running script from the command prompt

Improving Scripts

The separation of concerns is a design principle that helps organize code into distinct sections. Each section addresses a particular concern. In the assignment, the general separations that were done after the script description were: Data, Processing, and Presentation. These sections then can be broken down into subsections.

Functions can help with sectioning of the code. In the "ToDo" list, the list was displayed multiple times. This would have been easier to have the same header appear as a predefined function.

Script Templates help with consistency for a script. The same layout makes it easier for everyone who works on a program to know the way the script is organized. The header is especially important in providing information about what the script is about and when changes have been made. The template makes it easier for everyone involved to update in a consistent manner.

Error Handling (Try-Except)

Error handling can help manage errors. The try-except construct can help trap errors. It can help limit errors to a group of statements. The except part expects a specific case.

Assignment 05 "ToDoList"

This assignment began with a starter script. At first, I thought that variable list was a limitation since there was no "TODO" where code was to be inserted. There was also no output example to try to match so there were more choices in how each section was processed. The script was already organized by separation of concerns. The template for the Title, Description, and ChangeLog were already in place. The sections for Data, Processing were provided. I approached the assignment by section after reading it over. The first part of loading the ToDoList file needed some sort of check to see if anything was in the file. This was an opportunity to use the try-except. It was also a section that created multiple errors. The next section was the processing portion that displayed the menu and showed the user their To Do List. I deliberately left the "TODO" parts in place as there is still more formatting and error handling that can take place. Tasks were displayed at each menu choice. The "remove an item" caused me to try out a variety of options. The items require that the whole task name be written out in the version I ended up using. It is more cumbersome but it also gives more control to the user. The last part that I did was try out the program and add comments.

```
# ------------------------------------------------------------------------ #
# Title: Assignment 05
# Description: Working with Dictionaries and Files
#              When the program starts, load each "row" of data
#              in "ToDoList.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
# BSpadavecchia,02-13-2022,Added code to complete assignment 5
# BSpadavecchia,02-14-2022,Revised code on assignment 5
# BSpadavecchia,02-15-2022,Added try except on assignment 5
# BSpadavecchia,02-16-2022,Revised code on assignment 5
# BSpadavecchia,02-17-2022,Revised code on "Add a new item"
# BSpadavecchia,02-18-2022,Revised code on "Remove an existing item"
# BSpadavecchia,02-19-2022,Revised code on "Save Data to File
# ------------------------------------------------------------------------ #
```

```
# -- Data -- #
# Step 1 - declare variables and constants
objFile = []
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 select
strFileName = "ToDoList.txt"

# -- 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)
# The try-except is testing to see if the ToDoList.txt file has anything in it to load
try:
    print("What task do you need to do?")
    print("Here are your current tasks:")
    objFile = open(strFileName, "r")
    for row in objFile:
        strData = row.split(",")
        dicRow = {"task": strData[0].strip(), "priority": strData[1].strip()}
        print(dicRow)
        lstTable.append(dicRow)
    objFile.close()
    input("Press enter to continue")
except:
    if len(strFileName) < 0:
        print("Your ToDo List is empty")

# TODO: Add Code Here (possible revision on print format)
# -- 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] - "))
    print()  # adding a new line for looks
    if (strChoice.strip() == '1'):
        print("**********Here is your current ToDo list:***********\n")
        for row in lstTable:
            print("Task:", row['task'], "|", "Priority:", row['priority'])
        else:
            if len(lstTable) == 0:
                print("You have nothing in your ToDo list")
        continue

        # TODO: Add Code Here
        # Step 3  - Add a new item to the list/Table
    elif (strChoice.strip() == '2'):
        strTask = input("Please enter a new task:  ").strip()
        strPriority = input("Please enter a priority:  ").strip()
        print("************************************************") #  adding a new line for looks
        print("**********Here is your current ToDo list:***********\n")
        dicRow = {"task": strTask, "priority": strPriority}
        lstTable.append(dicRow)
        for row in lstTable:
            print("Task:", row['task'], "|", "Priority:", row['priority'])

        continue
        # Step 4 - Remove a new item from the list/Table
    elif (strChoice.strip() == '3'):
        print("Here is your current ToDo List:\n")
        for row in lstTable:
            print("Task:", row['task'], "|", "Priority:", row['priority'])
        strKeytoRemove = input("\nWhat task do you want to delete?  ")
```

```
        for row in lstTable:
            if row['task'].lower() == strKeytoRemove.lower():
                lstTable.remove(row)
                print("Task: " + strKeytoRemove + " was removed from list\n")
                print("Here is your current list")
                for row in lstTable:
                    print("Task:", row['task'], "|", "Priority:", row['priority'])
            else:
                print("That task does not exit")
        print("*****" * 6)

     # TODO: Add Code Here (Would like more Error Handling)
        continue
    # Step 5 - Save tasks to the ToDoList.txt file

    elif (strChoice == '4'):
        # Step 5a - Show the current items in the table
        print("*****" *6)
        print("Here is your current list")
        for row in lstTable:
            print("Task:", row['task'], "|", "Priority:", row['priority'])
        print("*****" * 6)

        # Step 5b - Ask if they want save that data
        if ("y" == str(input("Save this data to file? (y/n) - ")).strip().lower()):
            objFile = open(strFileName, "w")
            for dicRow in lstTable:
                objFile.write(dicRow["task"] + "," + dicRow["priority"] + "\n")
            objFile.close()
            input("Data saved to file! Press the [Enter] key to return to menu.")
        else:
            input("New data was NOT Saved.  Please press [Enter] to return to Menu.")
            continue

        # TODO: Add Code Here
        continue
    # Step 6 - Exit program
    elif (strChoice.strip() == '5'):
        # TODO: Add Code Here
        break  # and Exit the program

    else:
        print("\n Please choose a menu option [1-5]\n")
```

Figure 5 Script for ToDo List

```
C:\Users\bspad>python.exe "C:\_PythonClass\Assignment05\Assignment05.py"
What task do you need to do?
Here are your current tasks:
{'task': 'clean bedroom', 'priority': 'low'}
Press enter to continue

    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 new task:  clean bathroom
Please enter a priority:  low
**********************************************
**********Here is your current ToDo list:************

Task: clean bedroom | Priority: low
Task: clean bathroom | 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

Here is your current ToDo List:

Task: clean bedroom | Priority: low
Task: clean bathroom | Priority: low

What task do you want to delete?  clean bedroom
Task: clean bedroom was removed from list

Here is your current list
Task: clean bathroom | Priority: low
******************************
```

Figure 6 Assignment5 script running in command prompt

GitHub

GitHub is a way to make backups of code files and make them available for others. It can store multiple versions of the code files and you have a commit history and who made changes. You can also access it from any location. You can have several people working on the same project at the same time. There is built in issue tracking.

Conclusion

The ToDo List project gave the feel of a real chance to code but it also gave me so many options. I now have a better understanding of the difference between dictionaries and lists. I also appreciate the benefit of naming conventions for variables. While it was interesting to try to do some error-handling, I did find that it was still very easy to produce errors. GitHub was easy to sign up for and I will look forward to its benefits.