Deja Monet
Randal Root
Foundations of Python
10 August 2022
GitHub

Assignment 05: Lists & Dictionaries

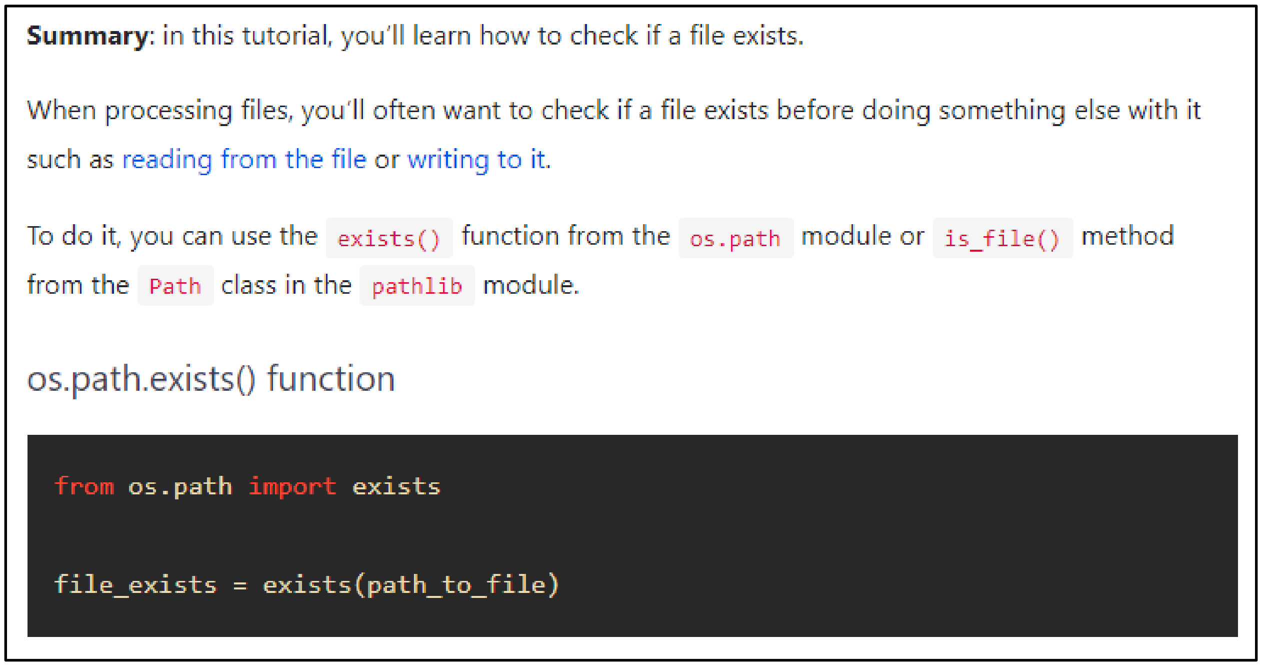
Introduction

In this assignment, I will review how to create, view, and edit data in Python lists and dictionaries. Additionally, I will review importing a Python module or library and calling a function from a module.

Completing the Assignment: Writing the Code

In this assignment, students were told not to use functions, use pre-written code and already defined variables – which had already been organized accordingly – so I decided to use Spyder instead of Jupyter Lab as my IDE. The first thing I did was review the steps the script had to complete as well as the already defined variables. I then added two variables, **str_task** and **str_priority,** to capture the user input data in Step 4.

The first step in the assignment is to load any data that already exists in a text file ToDoList.txt into a list of dictionary rows. I had to research how to check if a file exists, and found that I could do so using the **exists()** function from the **os.path** module (**Figure 1**).



Summary: in this tutorial, you'll learn how to check if a file exists.

When processing files, you'll often want to check if a file exists before doing something else with it such as reading from the file or writing to it.

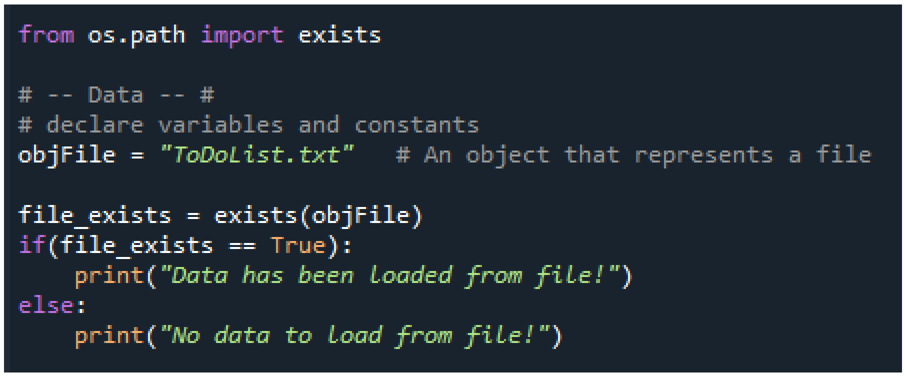
To do it, you can use the `exists()` function from the `os.path` module or `is_file()` method from the `Path` class in the `pathlib` module.

os.path.exists() function

```
from os.path import exists

file_exists = exists(path_to_file)
```

Figure 1. A description from pythontutorial.net on how to check if a file exists.

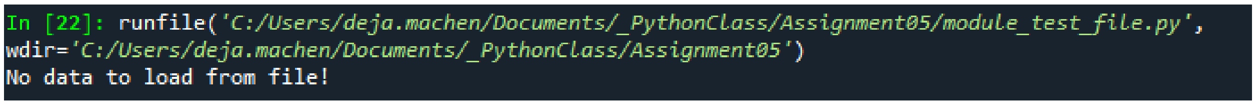
I set my working directory to my Assignment05 folder and created a new script **module_test_file.py** to test importing this module and the **exists()** function (**Figure 2**).



```python
from os.path import exists

# -- Data -- #
# declare variables and constants
objFile = "ToDoList.txt"   # An object that represents a file

file_exists = exists(objFile)
if(file_exists == True):
    print("Data has been loaded from file!")
else:
    print("No data to load from file!")
```

Figure 2. The code in module_test_file.py; an example of importing a library and using a function from this library.

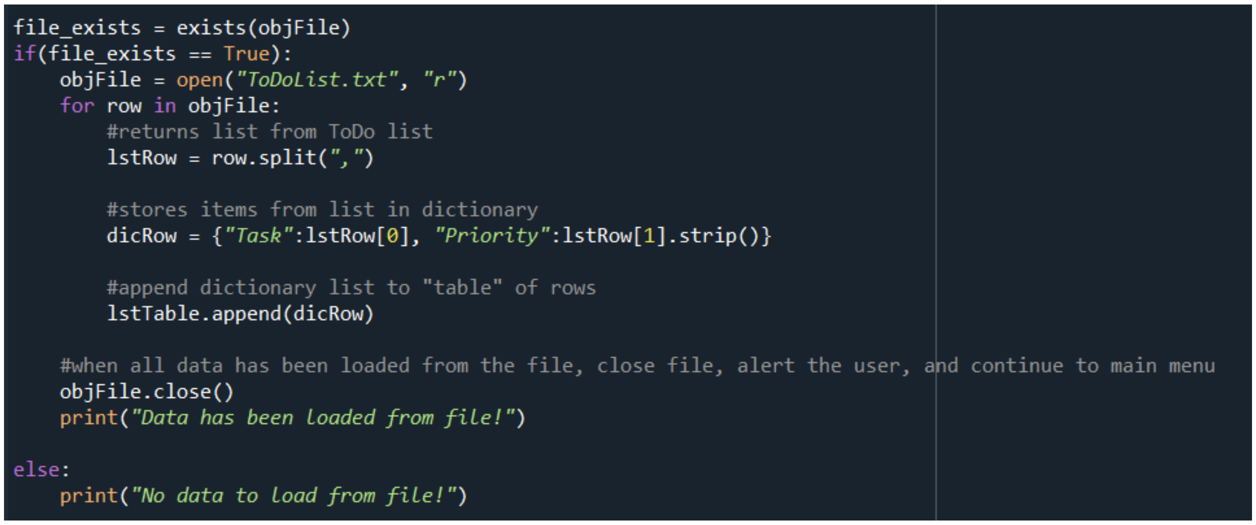
I ran the script when there was not a ToDoList.txt file in the Assignment05 directory, and the script correctly identified that the file did not exist; I also ran the script when a test file did exist in the directory (**Figure 3**).



```
In [22]: runfile('C:/Users/deja.machen/Documents/_PythonClass/Assignment05/module_test_file.py',
wdir='C:/Users/deja.machen/Documents/_PythonClass/Assignment05')
No data to load from file!
```

Figure 3. The output of the module_test_file.py script when there is no ToDoList.txt in the working directory; output when this .txt file does exist in the directory is not shown.

I then filled in the above if/else statements to load data from the ToDoList.txt file if this file did exist, or to tell the user there was no data to load if the file did not exist (**Figure 4**).

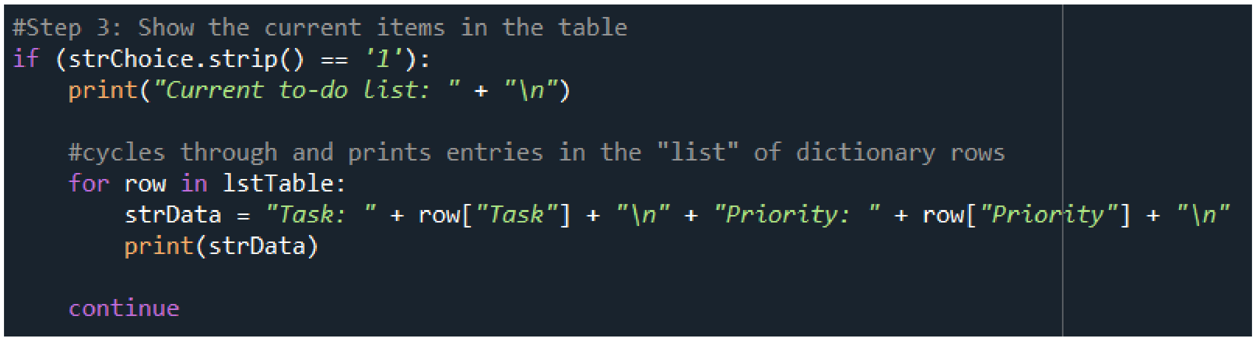


```python
file_exists = exists(objFile)
if(file_exists == True):
    objFile = open("ToDoList.txt", "r")
    for row in objFile:
        #returns list from ToDo list
        lstRow = row.split(",")

        #stores items from list in dictionary
        dicRow = {"Task":lstRow[0], "Priority":lstRow[1].strip()}

        #append dictionary list to "table" of rows
        lstTable.append(dicRow)

    #when all data has been loaded from the file, close file, alert the user, and continue to main menu
    objFile.close()
    print("Data has been loaded from file!")

else:
    print("No data to load from file!")
```

Figure 4. The script will load any data from the ToDoList.txt file into dictionary rows and append these rows to a table if this .txt file exists.

I used the pre-defined variables **lstRow** to return individual elements of the ToDoList.txt file, **dicRow** to store these items in a dictionary, and **lstTable** to store the dictionary rows as a list. The next step, displaying the menu of choices to the user (step 2) had already been completed, so I moved onto step 3, which is displaying the current items in the to-do list to the user. I used the pre-defined variable

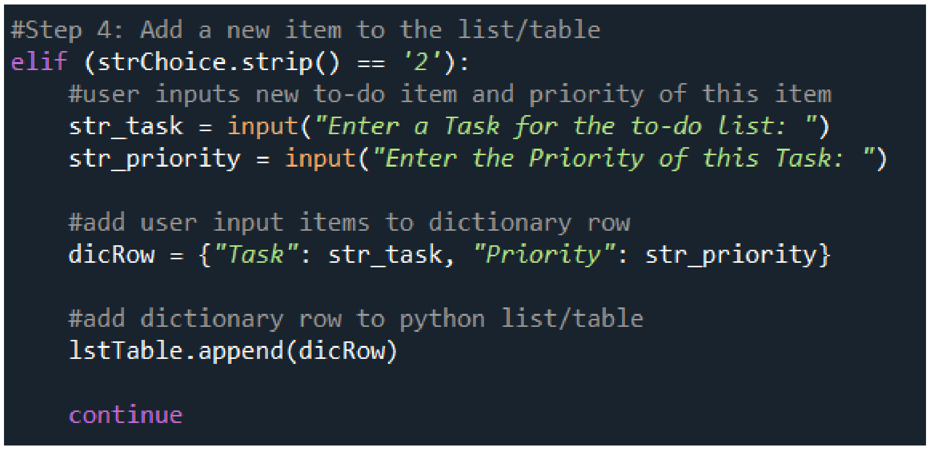
strData inside of a for loop in order to cycle through and print the items in the to-do list to the user (**Figure 5**).



```
#Step 3: Show the current items in the table
if (strChoice.strip() == '1'):
    print("Current to-do list: " + "\n")

    #cycles through and prints entries in the "list" of dictionary rows
    for row in lstTable:
        strData = "Task: " + row["Task"] + "\n" + "Priority: " + row["Priority"] + "\n"
        print(strData)

    continue
```

Figure 5. strData is used to cycle through all entries in lstTable and print these items to the user.

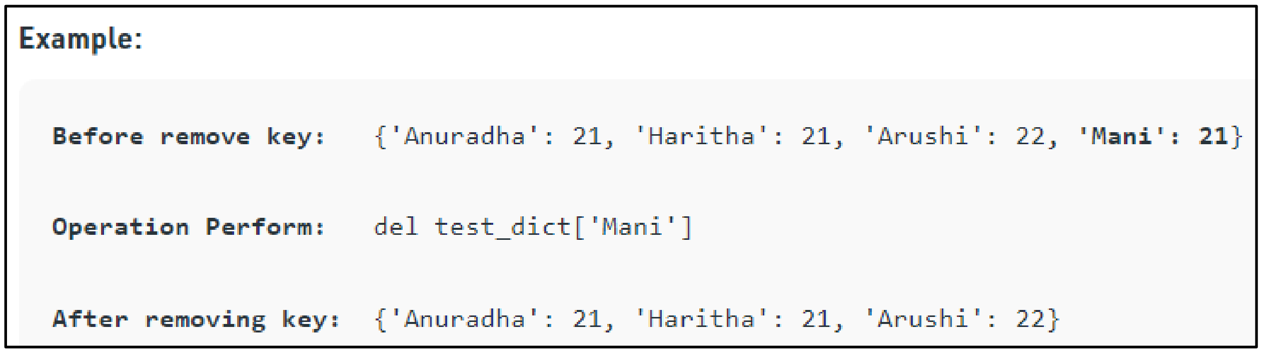
The next step was to allow the user to add new items to the to-do list. To do this, I allowed the user to input a new item and the priority of this item, stored as the variables **str_task** and **str_priority**, respectively. I then stored these items in the dictionary row, and then appended this dictionary row to the list/table (**Figure 6**).



```
#Step 4: Add a new item to the list/table
elif (strChoice.strip() == '2'):
    #user inputs new to-do item and priority of this item
    str_task = input("Enter a Task for the to-do list: ")
    str_priority = input("Enter the Priority of this Task: ")

    #add user input items to dictionary row
    dicRow = {"Task": str_task, "Priority": str_priority}

    #add dictionary row to python list/table
    lstTable.append(dicRow)

    continue
```

Figure 6. dicRow contains the user input to-do item and its priority, and is then appended to the list of to-do items.

The next step is to allow the user to remove items from the to-do list. I researched how to remove items from a dictionary, and found that the **del keyword** can be used to do this (**Figure 7**).

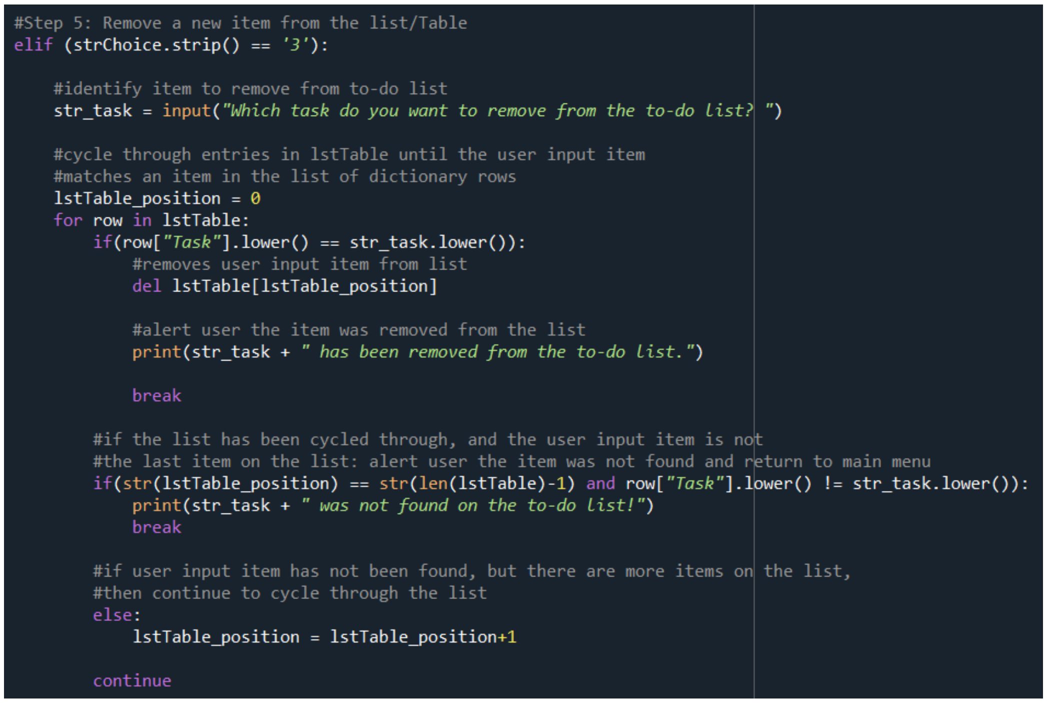


Example:

```
Before remove key:   {'Anuradha': 21, 'Haritha': 21, 'Arushi': 22, 'Mani': 21}

Operation Perform:   del test_dict['Mani']

After removing key:  {'Anuradha': 21, 'Haritha': 21, 'Arushi': 22}
```

Figure 7. An example from GeeksforGeeks.org of using the del keyword in order to remove an item from a dictionary in Python.

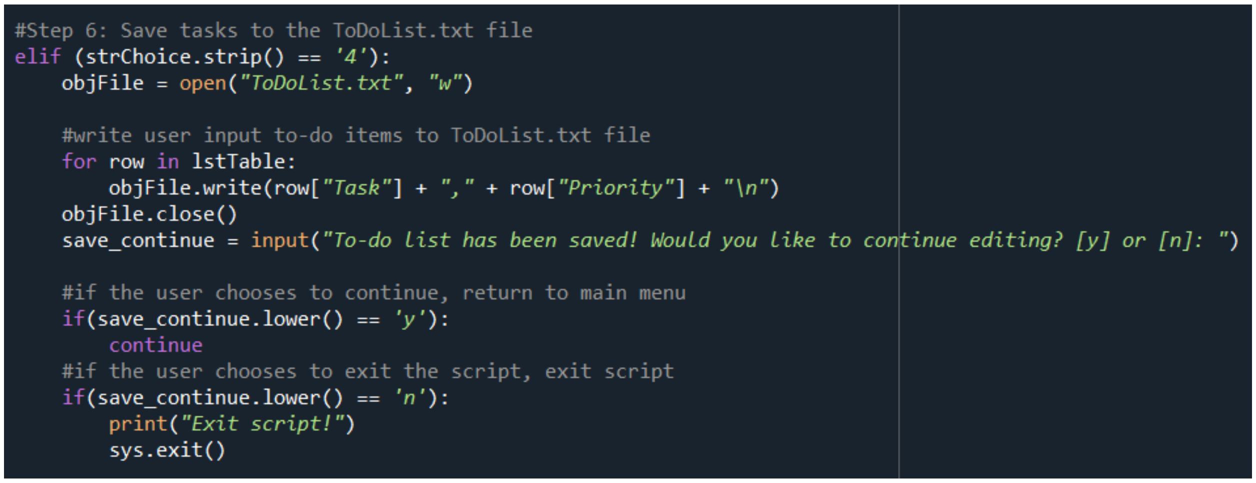
Since **lstTable** is a collection of dictionary rows, I could use the **del** keyword with **lstTable** in order to remove a to-do item from the to-do list. I decided to cycle through the entries in **lstTable** using a for loop until the user input item matched an item in the to-do list, and then remove the item. I used the variable **lstTable_position** in order to cycle through **lstTable**, and set this value to zero, the first position in the list. I retrieved the item that the user wanted to remove from the list using the **input()** function and stored this as **str_task**. In order to cycle through the **lstTable** array, I identified the variable **lstTable_position**, which increases until the user input item is found or until the end of the list is reached. If the end of the list is reached and the user input item has not been found, the user is alerted and returned to the main menu (**Figure 8**).



```
#Step 5: Remove a new item from the list/Table
elif (strChoice.strip() == '3'):

    #identify item to remove from to-do list
    str_task = input("Which task do you want to remove from the to-do list? ")

    #cycle through entries in lstTable until the user input item
    #matches an item in the list of dictionary rows
    lstTable_position = 0
    for row in lstTable:
        if(row["Task"].lower() == str_task.lower()):
            #removes user input item from list
            del lstTable[lstTable_position]

            #alert user the item was removed from the list
            print(str_task + " has been removed from the to-do list.")

            break

        #if the list has been cycled through, and the user input item is not
        #the last item on the list: alert user the item was not found and return to main menu
        if(str(lstTable_position) == str(len(lstTable)-1) and row["Task"].lower() != str_task.lower()):
            print(str_task + " was not found on the to-do list!")
            break

        #if user input item has not been found, but there are more items on the list,
        #then continue to cycle through the list
        else:
            lstTable_position = lstTable_position+1

        continue
```

Figure 8. The user input item will be removed from the to-do list; if the user-input item is not on the to-do list, the user will be returned to the main menu.

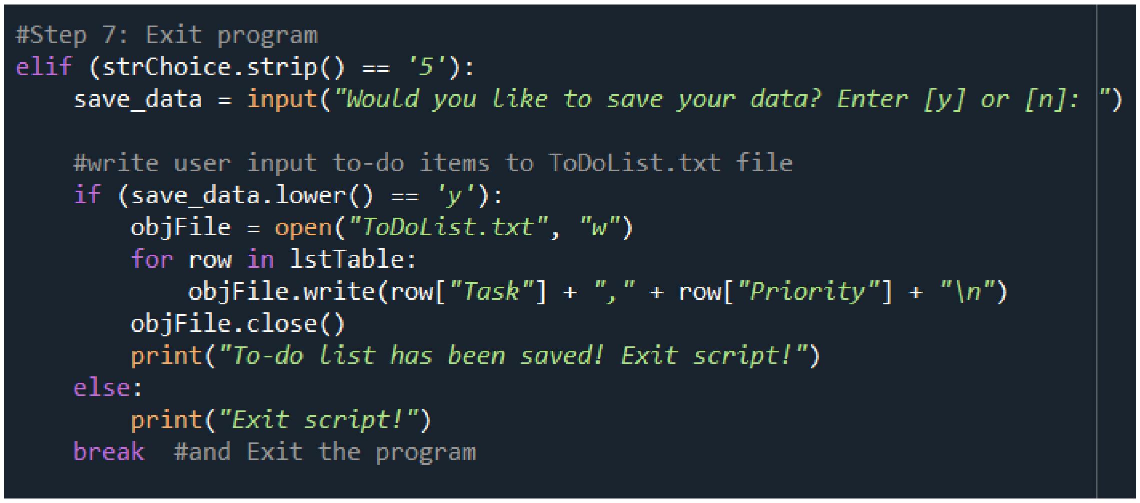
The next step was to save all of the tasks on the to-do list to a .txt file. I saved the data for the user by using a for loop, then gave the user the option to continue the script or exit. To exit, I imported the **sys** module in order to use the function **sys.exit()** (**Figure 9**).



```
#Step 6: Save tasks to the ToDoList.txt file
elif (strChoice.strip() == '4'):
    objFile = open("ToDoList.txt", "w")

    #write user input to-do items to ToDoList.txt file
    for row in lstTable:
        objFile.write(row["Task"] + "," + row["Priority"] + "\n")
    objFile.close()
    save_continue = input("To-do list has been saved! Would you like to continue editing? [y] or [n]: ")

    #if the user chooses to continue, return to main menu
    if(save_continue.lower() == 'y'):
        continue
    #if the user chooses to exit the script, exit script
    if(save_continue.lower() == 'n'):
        print("Exit script!")
        sys.exit()
```

Figure 9. After the data is saved, the user can choose to return to the main menu or to exit the script.

The last step was to allow the user to choose to exit the script. In this step, I also allowed the user to choose to save their data. Instead of exiting the script by using **sys.exit()**, this loop exits by using **break**. In this step, the user is not given the choice to return to the main menu (**Figure 10**).



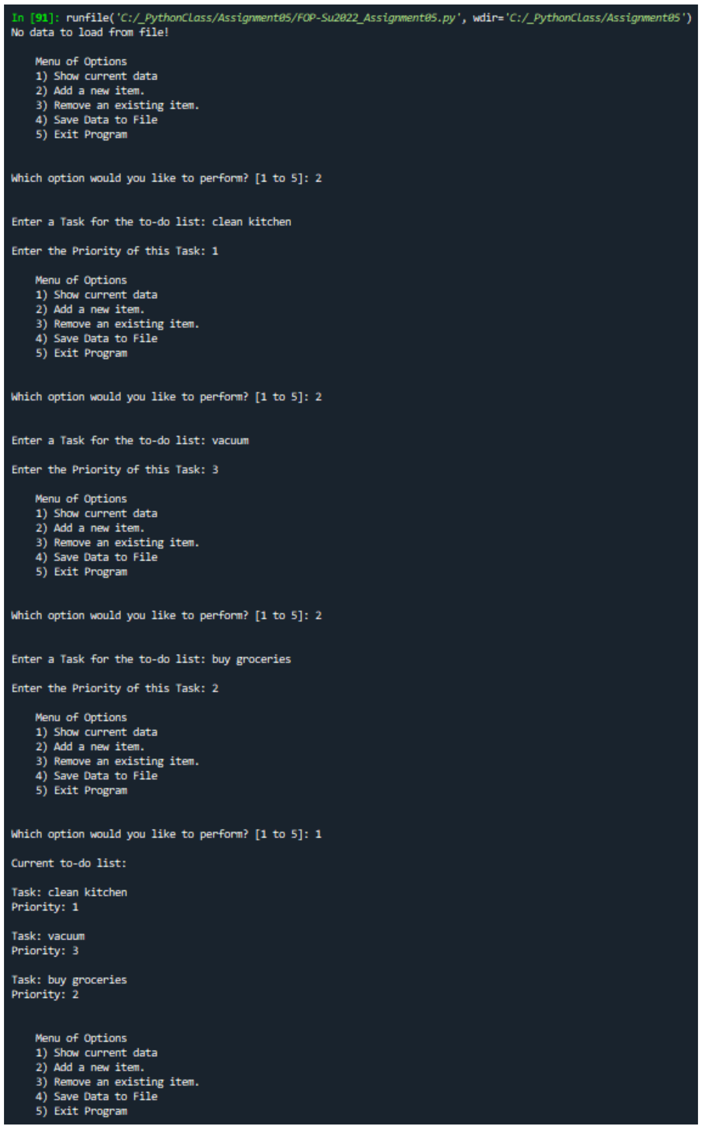
```
#Step 7: Exit program
elif (strChoice.strip() == '5'):
    save_data = input("Would you like to save your data? Enter [y] or [n]: ")

    #write user input to-do items to ToDoList.txt file
    if (save_data.lower() == 'y'):
        objFile = open("ToDoList.txt", "w")
        for row in lstTable:
            objFile.write(row["Task"] + "," + row["Priority"] + "\n")
        objFile.close()
        print("To-do list has been saved! Exit script!")
    else:
        print("Exit script!")
    break  #and Exit the program
```

Figure 10. The user is given the choice to save their data before exiting the script.

The script completed, I was able to move on to testing and troubleshooting in Spyder and Anaconda Prompt.

<u>Completing the Assignment</u>: Testing the Script

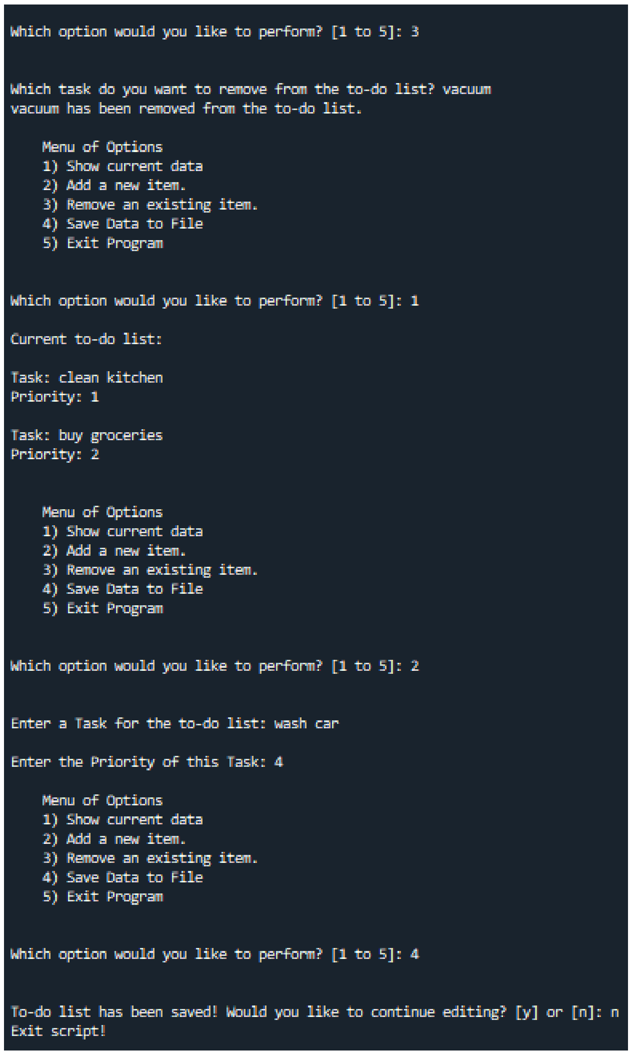
I began by running the script in Spyder when a ToDoList.txt file did not exist. I entered the to-do tasks “clean kitchen,” “vacuum,” and “buy groceries” as priority 1, 3, and 2, respectively. I then viewed the to-do list (**Figure** 11).



```
In [91]: runfile('C:/_PythonClass/Assignment05/FOP-Su2022_Assignment05.py', wdir='C:/_PythonClass/Assignment05')
No data to load from file!

    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


Enter a Task for the to-do list: clean kitchen

Enter the Priority of this Task: 1

    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


Enter a Task for the to-do list: vacuum

Enter the Priority of this Task: 3

    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


Enter a Task for the to-do list: buy groceries

Enter the Priority of this Task: 2

    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

Current to-do list:

Task: clean kitchen
Priority: 1

Task: vacuum
Priority: 3

Task: buy groceries
Priority: 2


    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 11. Part one of testing the script in Spyder.

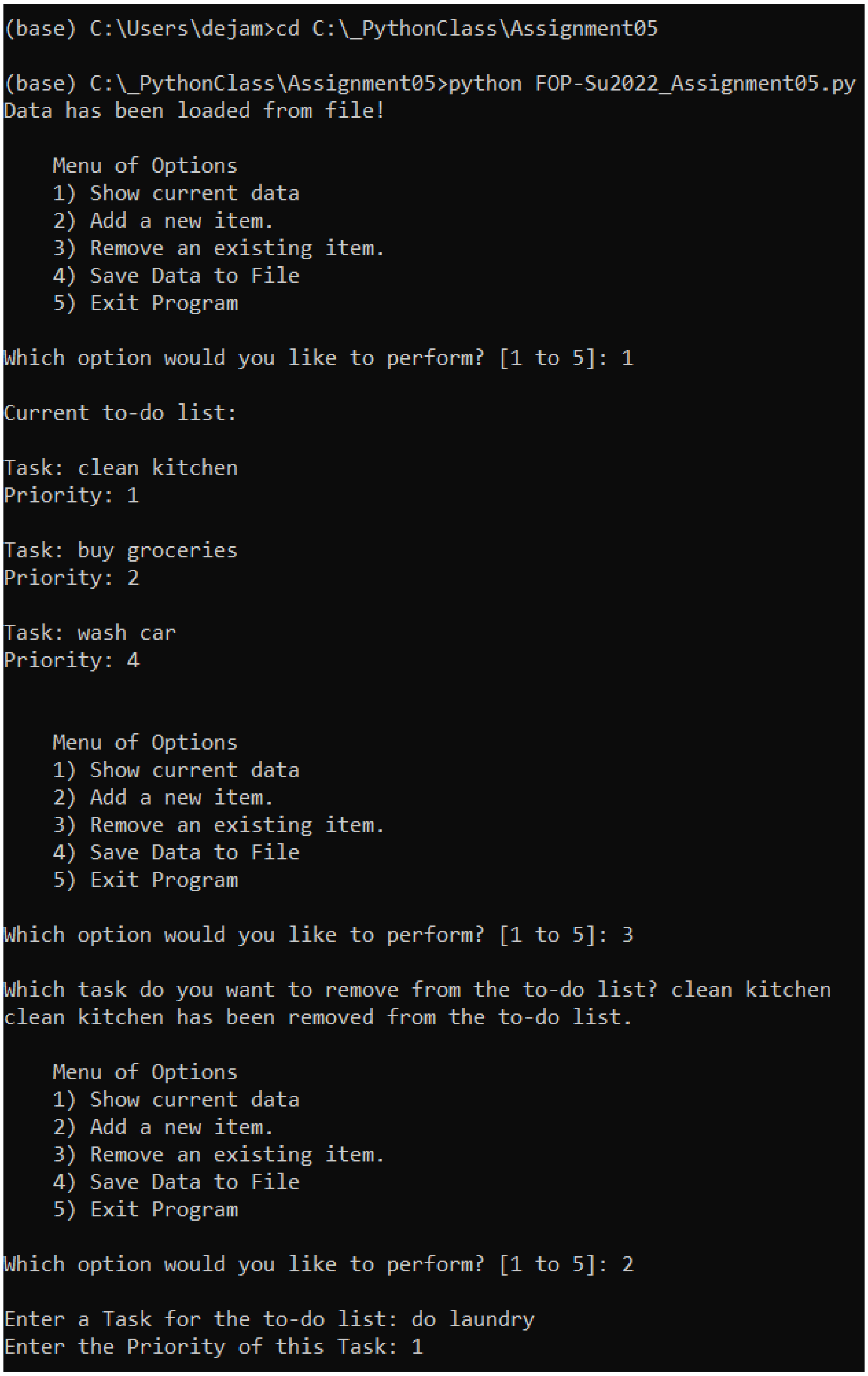
I then removed “vacuum,” and viewed the to-do list again. Finally, I added “wash car” as priority 4, then saved the to-do list, and exited the script (**Figure 12**).



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


Which task do you want to remove from the to-do list? vacuum
vacuum has been removed from the 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]: 1

Current to-do list:

Task: clean kitchen
Priority: 1

Task: buy groceries
Priority: 2


    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


Enter a Task for the to-do list: wash car

Enter the Priority of this Task: 4

    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


To-do list has been saved! Would you like to continue editing? [y] or [n]: n
Exit script!
```

Figure 12. Part two of testing the script in Spyder.

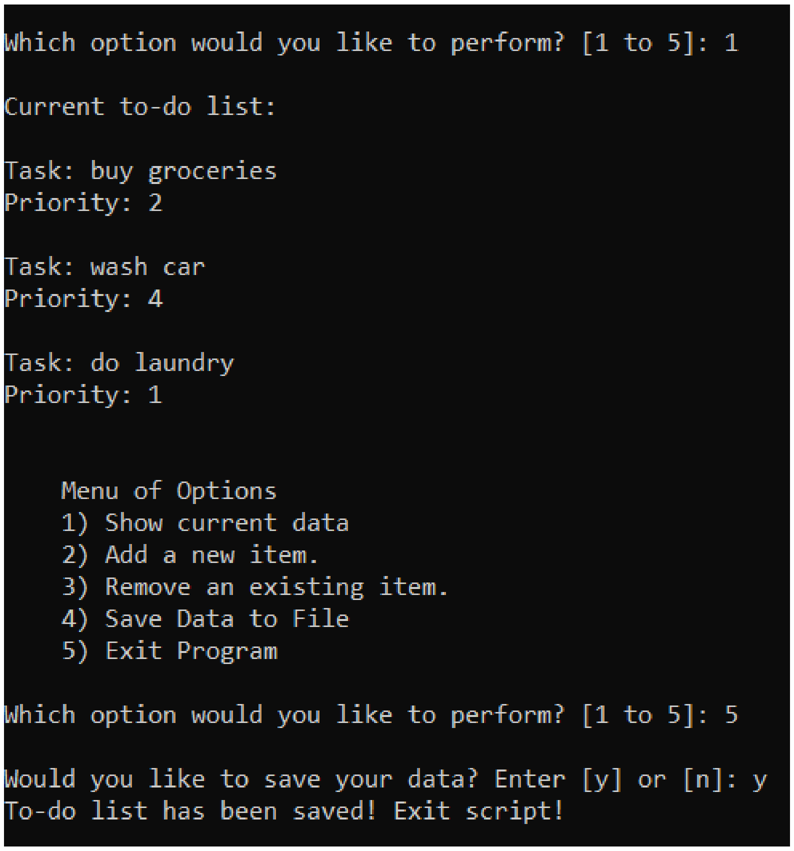
After testing the script in Spyder, I called the script in Anaconda Prompt when a ToDoList.txt file did exist, and viewed the data. Next, I removed “clean kitchen” from the to-do list, added “do laundry” as priority 1 (**Figure 13**).



```
(base) C:\Users\dejam>cd C:\_PythonClass\Assignment05

(base) C:\_PythonClass\Assignment05>python FOP-Su2022_Assignment05.py
Data has been loaded from file!

    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

Current to-do list:

Task: clean kitchen
Priority: 1

Task: buy groceries
Priority: 2

Task: wash car
Priority: 4


    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

Which task do you want to remove from the to-do list? clean kitchen
clean kitchen has been removed from the 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]: 2

Enter a Task for the to-do list: do laundry
Enter the Priority of this Task: 1
```

Figure 13. Part one of testing the script in Anaconda Prompt.

I then viewed the data again, chose to exit the program via option 5, chose to save the data from option 5, and exited the script (**Figure 14**).



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

Current to-do list:

Task: buy groceries
Priority: 2

Task: wash car
Priority: 4

Task: do laundry
Priority: 1


    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

Would you like to save your data? Enter [y] or [n]: y
To-do list has been saved! Exit script!
```

Figure 14. Part two of testing the script in Anaconda Prompt.

With the script completed and tested in both Spyder and Anaconda Prompt, the assignment was complete.

Summary

In this assignment, I reviewed editing and viewing data stored as Python lists and dictionaries. I also used Python modules in order to check if a file already existed in a working directory and to exit a script.