Name: Jane Shin
Date: November 16, 2022
Course: IT FDN 110 A Au 22: Foundations of Programming: Python

https://github.com/ks180337/IntroToProg-Python

Assignment 05 – Lists & Dictionaries

Introduction

This document describes the steps taken in performing Assignment 05, where a Python script file is created, using an existing starter script, that manages a "To-do List". The script uses a printed "menu" to guide the user through various options for the "To-do List", including displaying current list, adding a new item, removing an existing item, saving data to the original file, and exiting the program.

The "To-do List" file contains two columns of data, "Task" and "Priority." The columns are loaded into a Python dictionary object. Each dictionary object represents one row of data, and these rows are added to a Python list object to create a table of data.

This assignment requires updating existing template and code. New tools and concepts introduced in Module 5, such as list, dictionary, reading and writing data from a file into a list/dictionary will be used in accomplishing this task.

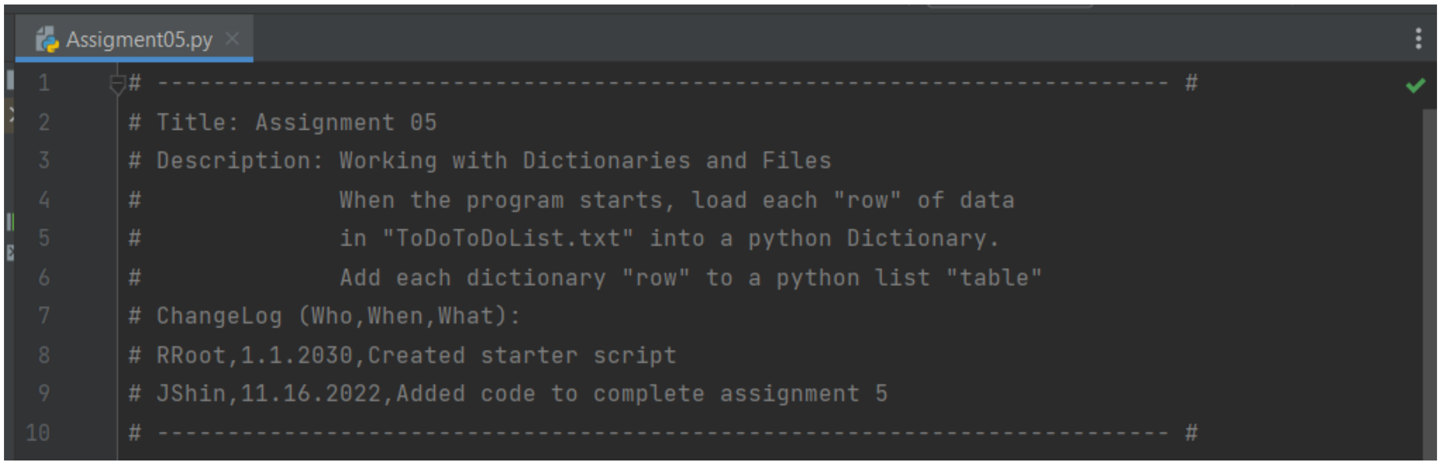
Instructions

Module 5 covered a detailed application of the list/dictionary concepts, delivered through course videos, a book chapter, and web pages. The course material delivered detailed explanation and examples of the following topics:

- Using List and Dictionary to store and load data
- Using index/key to manipulate data
- Reading data from a file into a list
- Reading data from a file into a dictionary
- Programming using a pattern called "Separations of Concerns"
- Using a function to organize code
- Using a script template and how it is used
- Error handling using Try-Except method
- Using GitHub for source control

Creating the Script

Header and Initial Comments



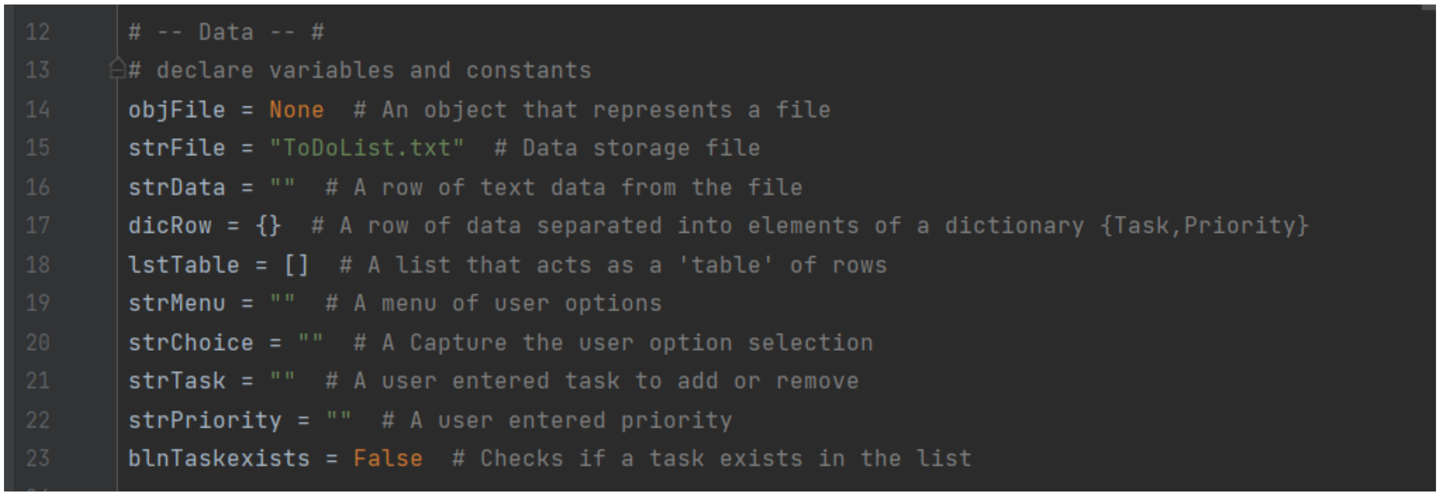
Assigment05.py

```
# ---------------------------------------------------------------------- #
# 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 each dictionary "row" to a python list "table"
# ChangeLog (Who,When,What):
# RRoot,1.1.2030,Created starter script
# JShin,11.16.2022,Added code to complete assignment 5
# ---------------------------------------------------------------------- #
```

Figure 1. Header and initial comments

- PyCharm was used to create and run the script. The assignment was saved as a .py file named "Assignment05.py" at the location according to a direction given in the instruction.
- Script header and comments – A script header was created per the examples given in the instruction to communicate the function and document changes of the script.
- The Change Log was updated to reflect the codes added to the original starter script to complete the assignment

Declaration of Variables

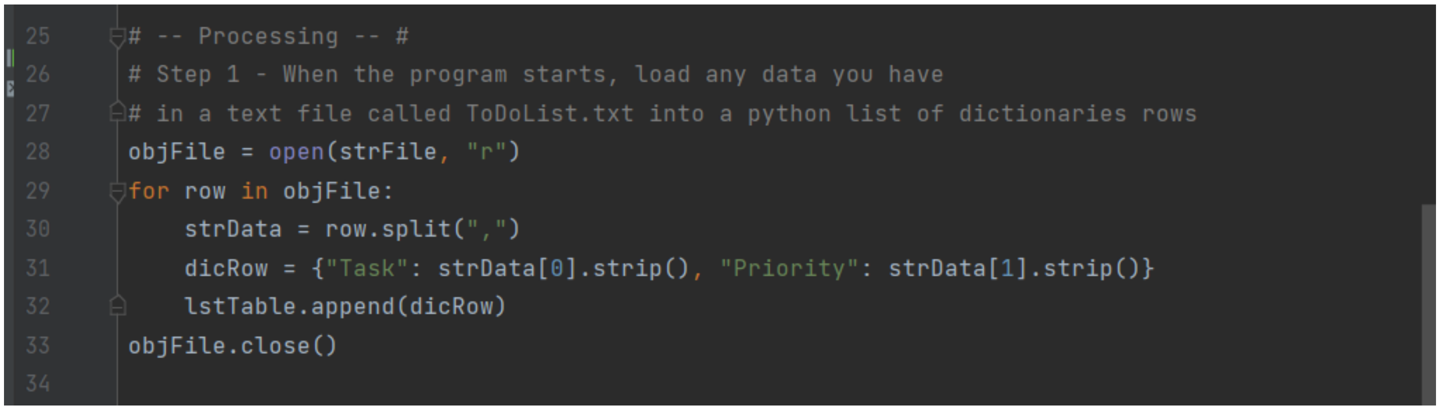


```
# -- Data -- #
# declare variables and constants
objFile = None  # An object that represents a file
strFile = "ToDoList.txt"  # Data storage file
strData = ""  # A row of text data from the file
dicRow = {}  # A row of data separated into elements of a dictionary {Task,Priority}
lstTable = []  # A list that acts as a 'table' of rows
strMenu = ""  # A menu of user options
strChoice = ""  # A Capture the user option selection
strTask = ""  # A user entered task to add or remove
strPriority = ""  # A user entered priority
blnTaskexists = False  # Checks if a task exists in the list
```

Figure 2. Declaration of variables

- Using the "Separation of Concerns" concept, variables and constants are declared within the "Data" section along with comments explaining the purposes of each variable.
- Variables are organized in order of appearance.
- Variable from the original script were retained, and several new variable, constants, and a Boolean variable were added.

Step 1 – Loading the data from text file

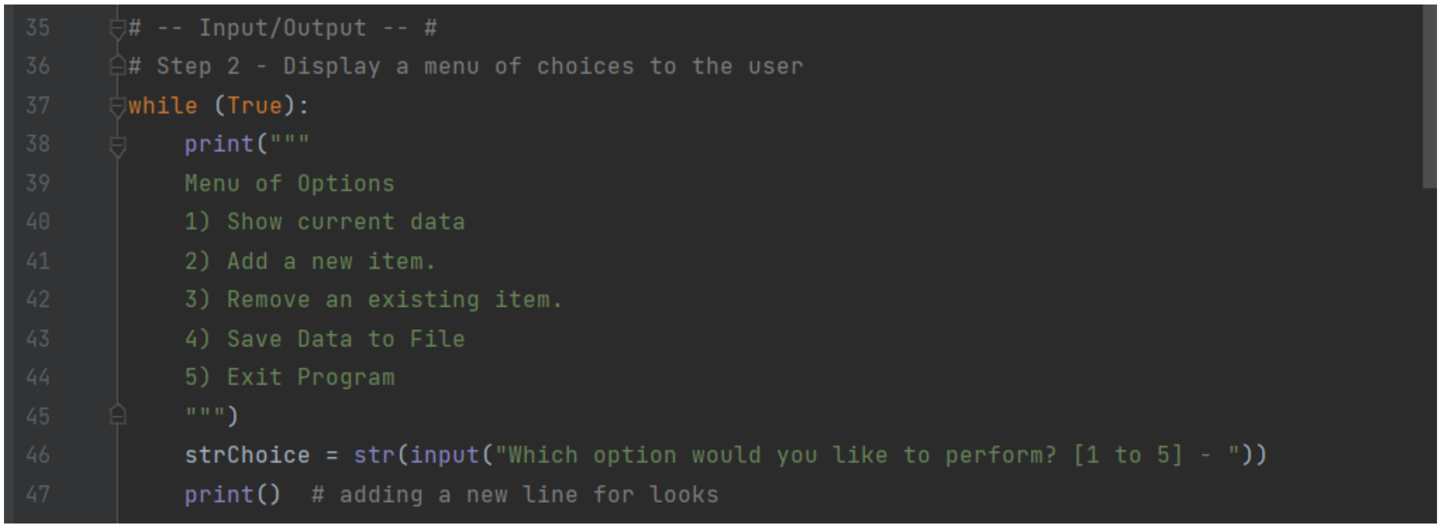


```
# -- Processing -- #
# Step 1 - When the program starts, load any data you have
# in a text file called ToDoList.txt into a python list of dictionaries rows
objFile = open(strFile, "r")
for row in objFile:
    strData = row.split(",")
    dicRow = {"Task": strData[0].strip(), "Priority": strData[1].strip()}
    lstTable.append(dicRow)
objFile.close()
```

Figure 3. Step 1 – Loading the data from text file

- The **open()** function was used to open the file that contains initial data, “ToDoList.txt”.
- “r” mode allows to read from a text file. If the file does not exist, Python will generate an error.
- The file is then assigned to the variable *objFile*.
- A for loop is chosen to loop through the data file, while reading each “rows” of text data from the file into *strData*.
- The values of *strData* then were assigned to a dictionary object representing one row of data, named *dicRow*, along with the keys “Task” and “Priority”. Note the curly brackets indicating a dictionary object.
- *dicRow* is then appended to a list object named *lstTable*.
- The **close()** method was used to close the file.
- **strip()** was used to remove any unnecessary space or carriage return.

Step 2 – Displaying a menu of choices

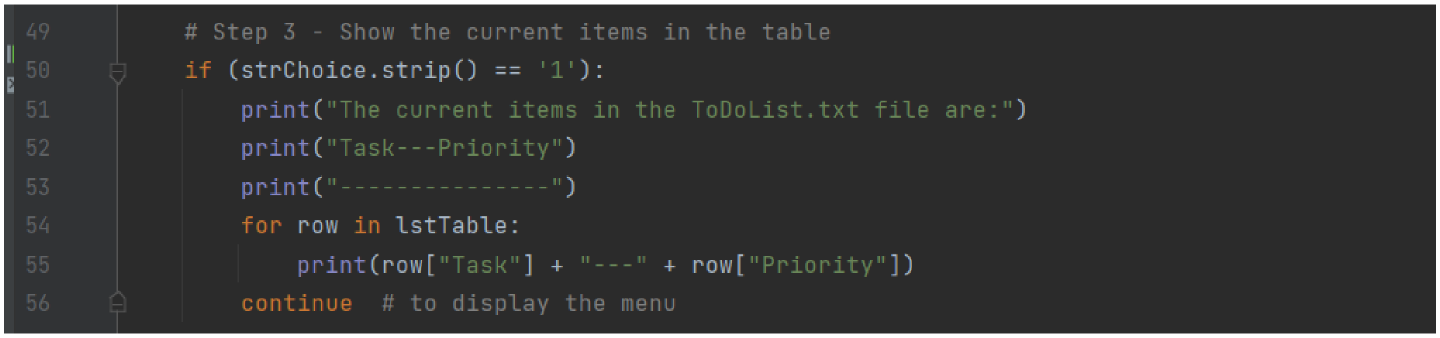


```
# -- Input/Output -- #
# Step 2 - Display a menu of choices to the user
while (True):
    print("""
    Menu of Options
    1) Show current data
    2) Add a new item.
    3) Remove an existing item.
    4) Save Data to File
    5) Exit Program
    """)
    strChoice = str(input("Which option would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

Figure 4. Step 2 – Displaying a menu of choices

- A **while** loop was selected to construct a loop that continuously displays a menu of choices, then solicits a user for a selection of options available.
- All lines after the **while** loop is indented to create a block that groups all logical statement together.
- Menu of options are displayed using **print()** function.
- Data types – the numbers obtained from user through **input()** function are initially of string data type, which will be used for the remainder of the script.
- Variables – generated data was assigned to variable *strChoice* to be used in subsequent operations.

Step 3 – Showing the current items

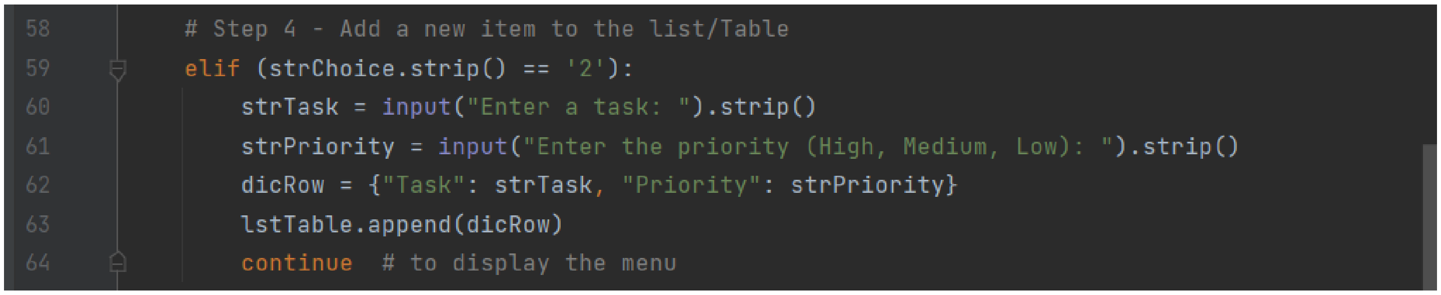


```
    # Step 3 - Show the current items in the table
    if (strChoice.strip() == '1'):
        print("The current items in the ToDoList.txt file are:")
        print("Task---Priority")
        print("---------------")
        for row in lstTable:
            print(row["Task"] + "---" + row["Priority"])
        continue  # to display the menu
```

Figure 5. Step 3 – Showing the current items

- *strChoice*, the variable that contains user option choice is evaluated using the **if…elif** sequence.
- If *strOption* equals 1, then Option 1 is chosen to display the items read from the "ToDoList.txt" file.
- A **for** loop is chosen to loop through the nested list *lstTable,* while printing each "rows". The keys were used to retrieve the corresponding values from the dictionary objects.
- The **elif** statement is bookended with **continue** to force the loop to jump back to the beginning and display the menu and asks the user to make the next choice.

Step 4 – Adding a new item

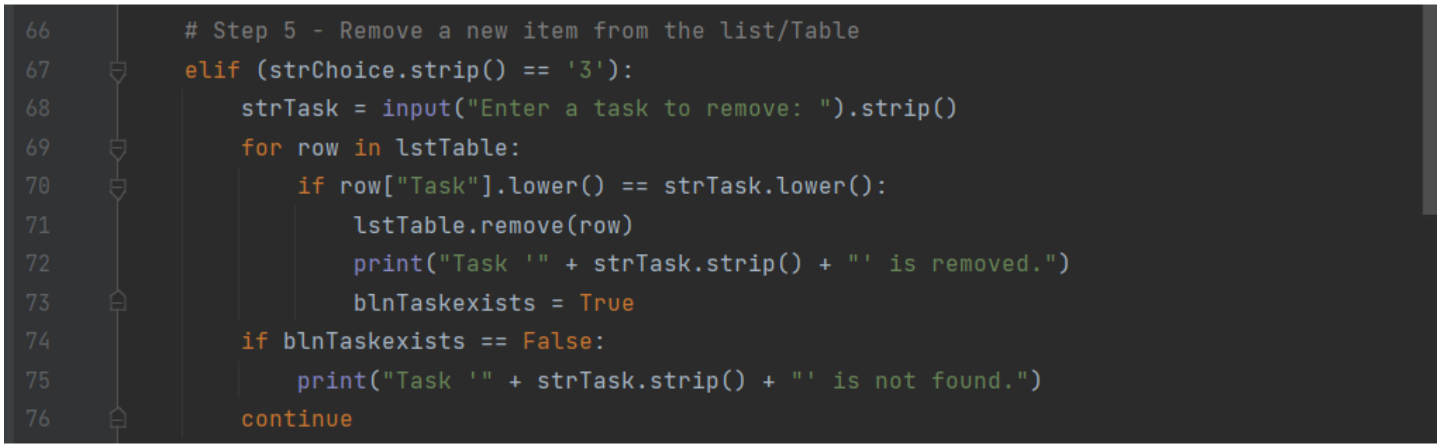


```
    # Step 4 - Add a new item to the list/Table
    elif (strChoice.strip() == '2'):
        strTask = input("Enter a task: ").strip()
        strPriority = input("Enter the priority (High, Medium, Low): ").strip()
        dicRow = {"Task": strTask, "Priority": strPriority}
        lstTable.append(dicRow)
        continue  # to display the menu
```

Figure 6. Step 4 – Adding a new item

- *strOption*, the variable that contains user option choice is evaluated using an **if** statement.
- If *strOption* equals 2, then Option 2 is chosen.
- **input()** function was used to prompt user to input the data required for program.
- Data types – the data obtained from user through **input()** functions are initially of string data type, which will be used for the remainder of the script.
- Variables – obtained data was assigned to variables (*strTask*, *strPriority*) to be used in subsequent operations.
- *strItem* and *strValue* are then assigned to a dictionary object *dicRow*, using curly brackets to indicate that it is a dictionary.
- *lstTable* is a nested list comprised of sublists *dicRow.*
- *dicRow* is appended to a list object named *lstTable*.
- The **if** statement is bookended with **continue** to force the loop to jump back to the beginning and display the menu and asks the user to make the next choice.

Step 5 – Removing an existing item

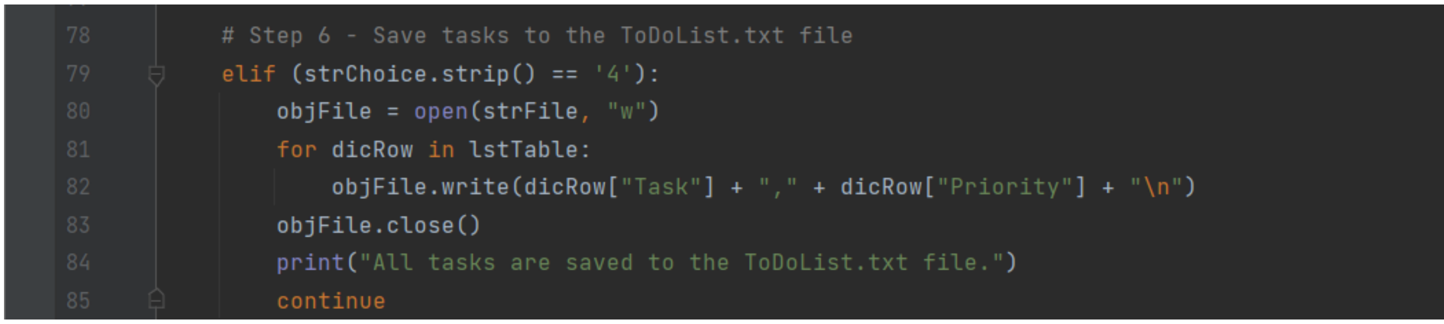


```
    # Step 5 - Remove a new item from the list/Table
    elif (strChoice.strip() == '3'):
        strTask = input("Enter a task to remove: ").strip()
        for row in lstTable:
            if row["Task"].lower() == strTask.lower():
                lstTable.remove(row)
                print("Task '" + strTask.strip() + "' is removed.")
                blnTaskexists = True
        if blnTaskexists == False:
            print("Task '" + strTask.strip() + "' is not found.")
        continue
```

Figure 7. Step 5 – Removing an existing item

- *strOption*, the variable that contains user option choice is evaluated using an **if** statement.
- If *strOption* equals 3, then Option 3 is chosen.
- **input()** function was used to prompt user to input the data required for program.
- Data types – the data obtained from user through **input()** functions are initially of string data type, which will be used for the remainder of the script.
- Variables – obtained data was assigned to variables (*strTask*) to be used in subsequent operations.
- A Boolean variable *blnTaskexists* is declared as False at the start of the script.
- A **for** loop is chosen to loop through the nested list *lstTable,* while comparing the stored value with key named "Task". If the *strTask* matches the value:
    - The corresponding row of the *lstTable* is removed
    - A message indicating that the task is removed from the database is displayed.
    - *blnTaskexists* value is set as True
- If *blnTaskexists* is still False at the end of the loop, that means the input task does not exist in the database. A message indicating that the task does not exist is displayed.
- The **if** statement is bookended with **continue** to force the loop to jump back to the beginning and display the menu and asks the user to make the next choice.

Step 6 – Saving tasks to the file



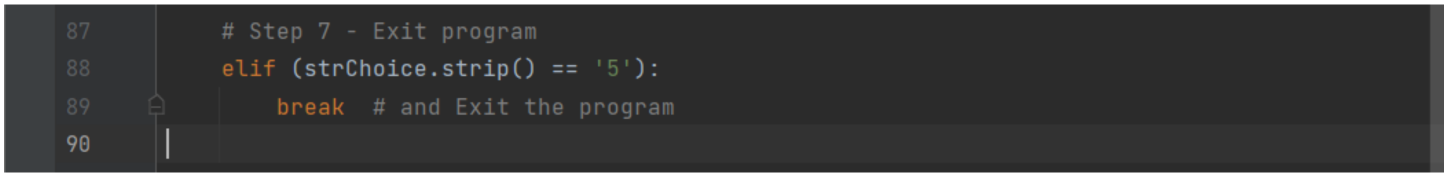
```
    # Step 6 - Save tasks to the ToDoList.txt file
    elif (strChoice.strip() == '4'):
        objFile = open(strFile, "w")
        for dicRow in lstTable:
            objFile.write(dicRow["Task"] + "," + dicRow["Priority"] + "\n")
        objFile.close()
        print("All tasks are saved to the ToDoList.txt file.")
        continue
```

Figure 8. Step 6 – Saving tasks to the file

- *strOption*, the variable that contains user option choice is evaluated using an **if** statement.
- If *strOption* equals 4, then Option 4 is chosen.
- The open() function was used to open a file.
- "w" mode allows to write to a text file. If the file already exists, its content is overwritten. If the file does not exist, it is newly created.

- The file is then assigned to the variable objFile.
- A for loop is chosen to loop through the nested list lstTable, while writing each “rows” or sublist into file.
- The write() method was used to write data stored in variables into the created file.
- After each row, a new line was created with “\n”
- The close() method was used to close the file.
- print() function was used to display a confirmation that the data was saved.
- The **if** statement is bookended with **continue** to force the loop to jump back to the beginning and display the menu and asks the user to make the next choice.

Step 7 – Exiting the program



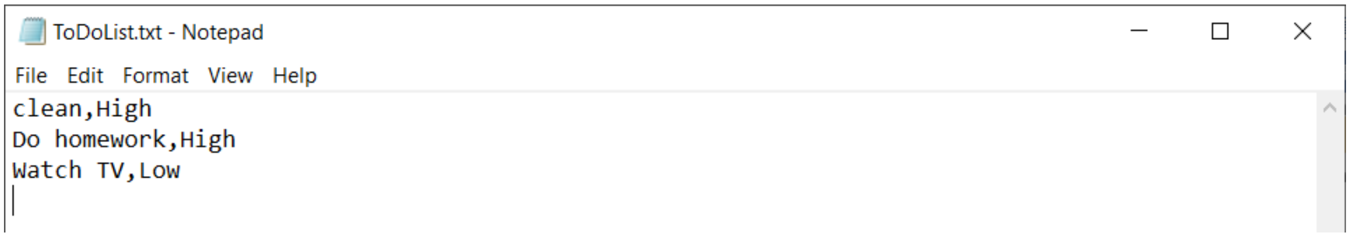
```
    # Step 7 - Exit program
    elif (strChoice.strip() == '5'):
        break  # and Exit the program

```

Figure 9. Step 7 – Exiting the program

- *strOption*, the variable that contains user option choice is evaluated using an **if** statement.
- If *strOption* equals 5, then Option 5 is chosen.
- The **elif** statement is bookended with **break** to break the while loop and end the program.

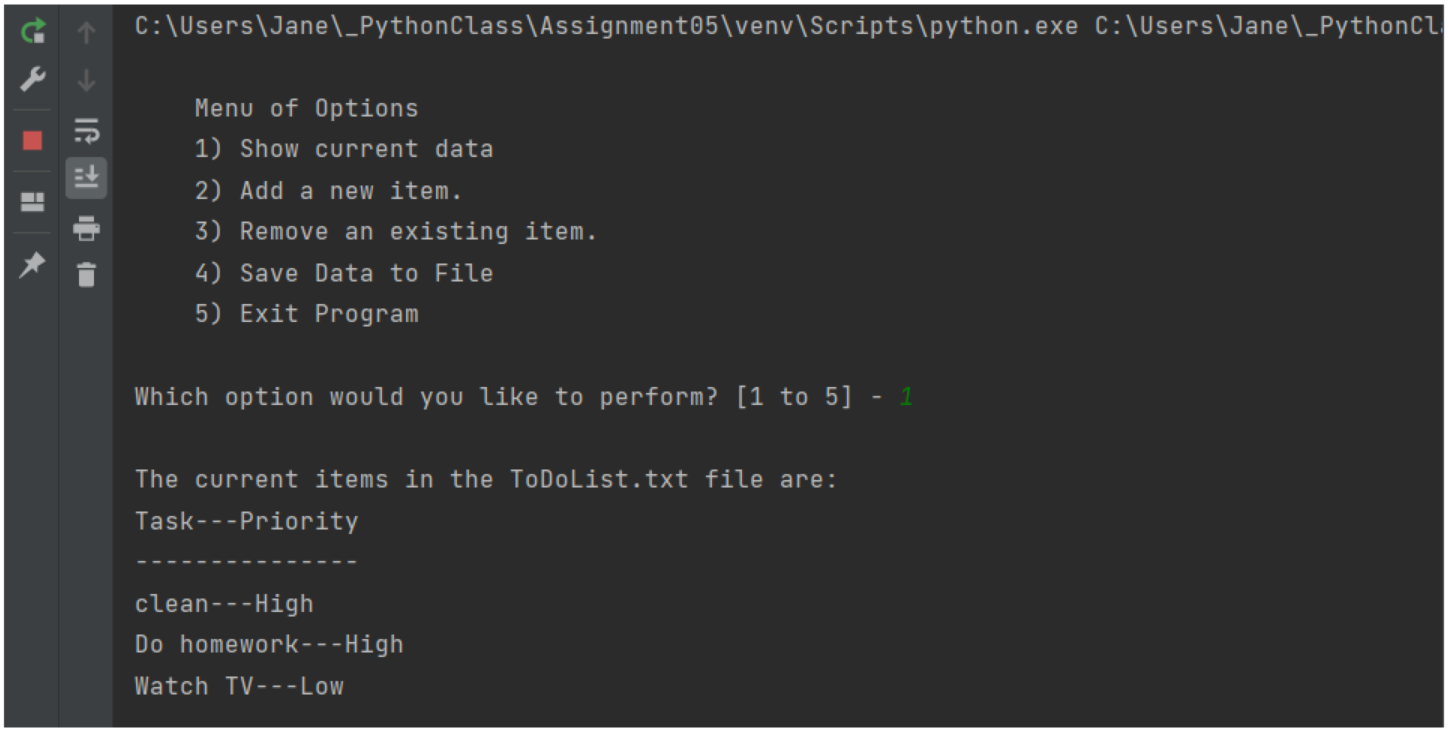
Running the Script

In PyCharm



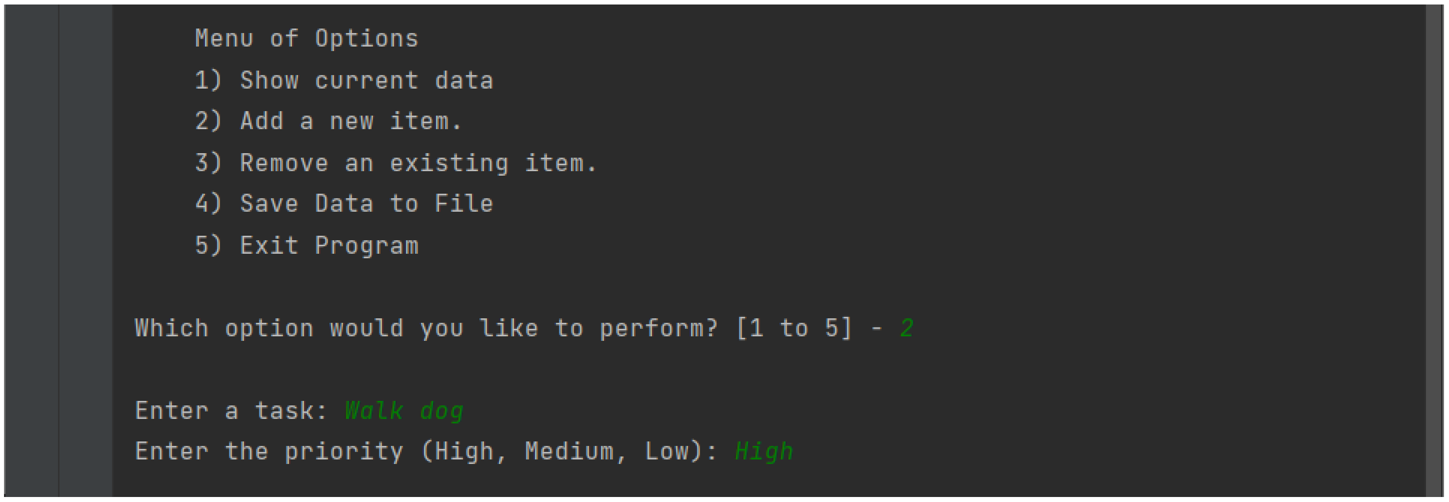
```
clean,High
Do homework,High
Watch TV,Low
```

Figure 10. The original ToDoList.txt



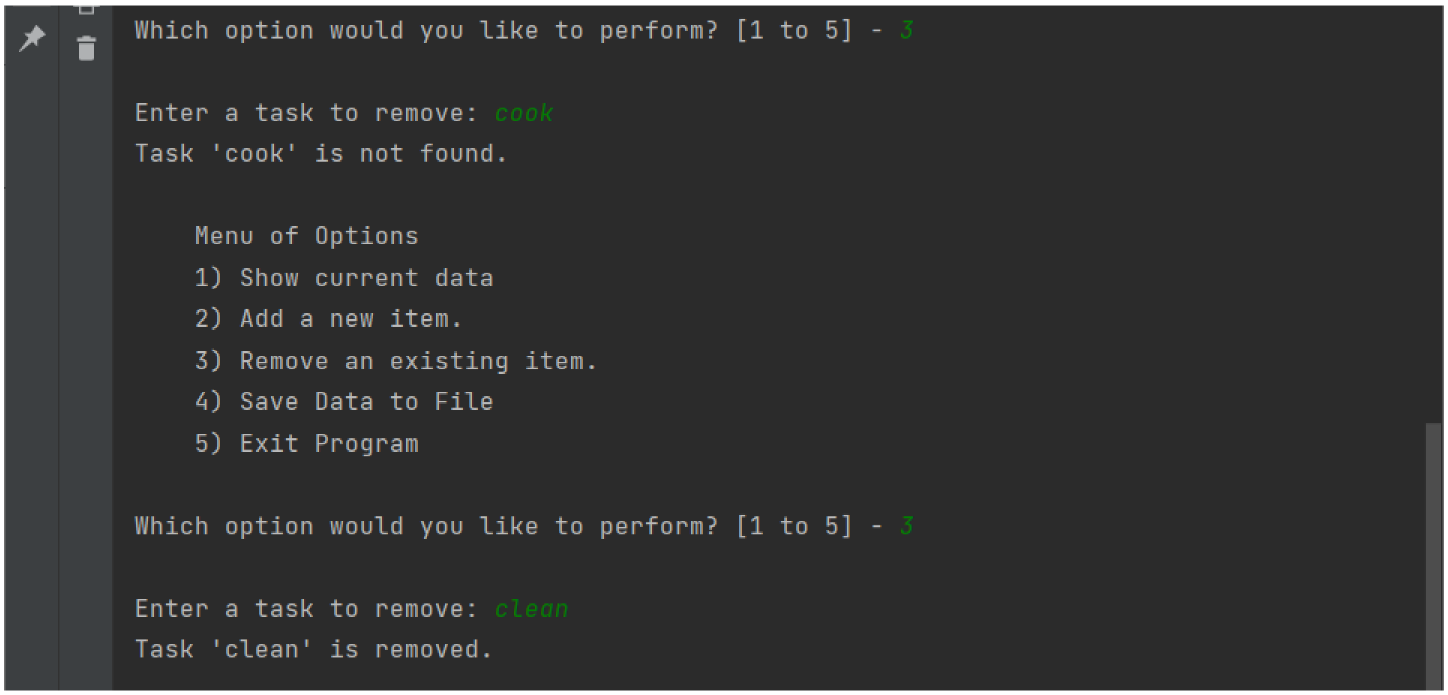
```
C:\Users\Jane\_PythonClass\Assignment05\venv\Scripts\python.exe C:\Users\Jane\_PythonCl

    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

The current items in the ToDoList.txt file are:
Task---Priority
---------------
clean---High
Do homework---High
Watch TV---Low
```

Figure 11. Option 1



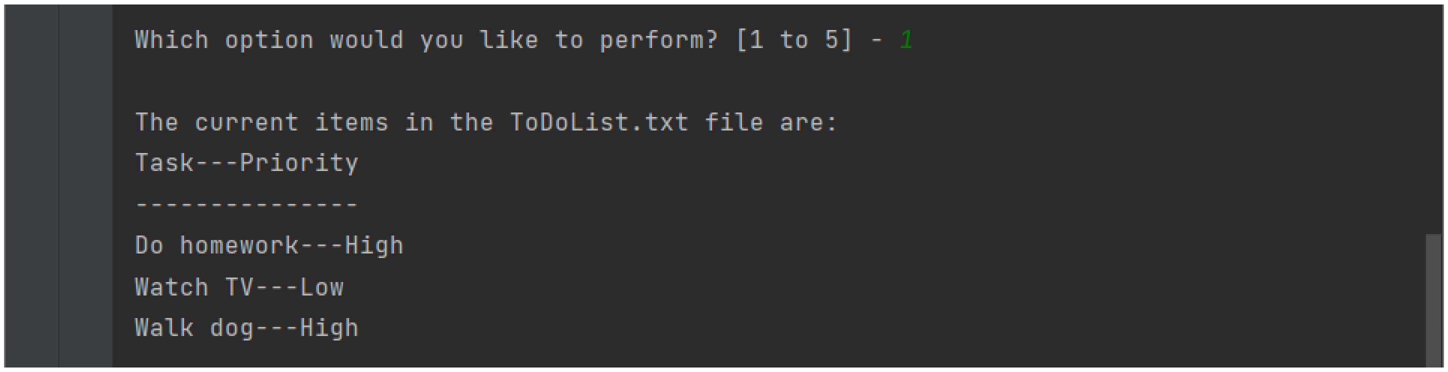
```
    Menu of Options
    1) Show current data
    2) Add a new item.
    3) Remove an existing item.
    4) Save Data to File
    5) Exit Program

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

Enter a task: Walk dog
Enter the priority (High, Medium, Low): High
```

Figure 11. Option 2



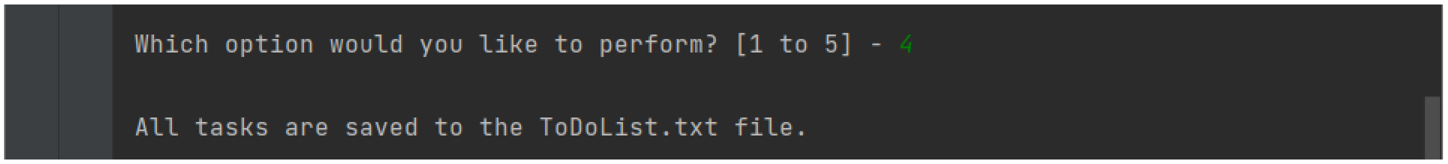
```
Which option would you like to perform? [1 to 5] - 3

Enter a task to remove: cook
Task 'cook' is not found.

    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

Enter a task to remove: clean
Task 'clean' is removed.
```

Figure 12. Option 3



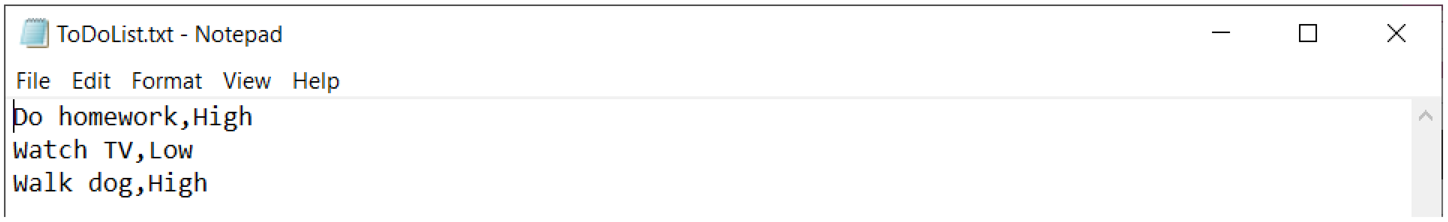
```
Which option would you like to perform? [1 to 5] - 1

The current items in the ToDoList.txt file are:
Task---Priority
---------------
Do homework---High
Watch TV---Low
Walk dog---High
```

Figure 13. Option 1 to show the updated list



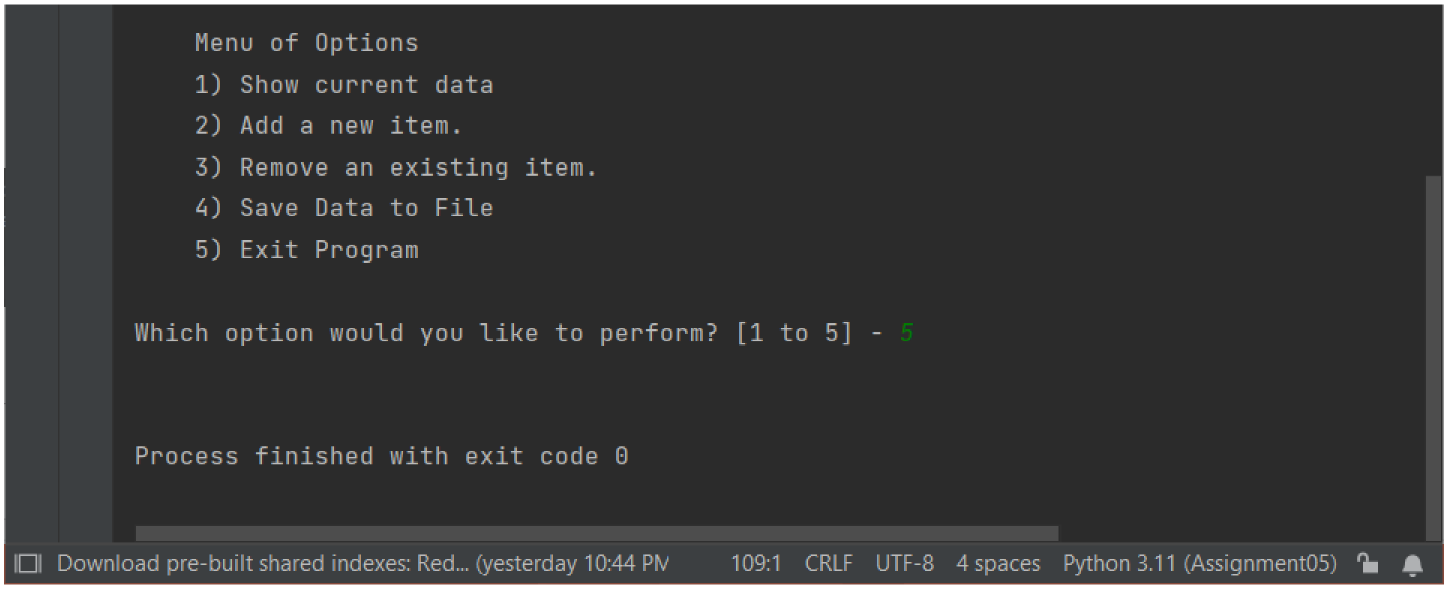
```
Which option would you like to perform? [1 to 5] - 4

All tasks are saved to the ToDoList.txt file.
```

Figure 14. Option 4



```
Do homework,High
Watch TV,Low
Walk dog,High
```

Figure 15. Updated ToDoList.txt



```
    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


Process finished with exit code 0
```

Download pre-built shared indexes: Red... (yesterday 10:44 PM
109:1 CRLF UTF-8 4 spaces Python 3.11 (Assignment05)

Figure 16. Option 5

- The script has been run successfully in PyCharm.

Summary

The script described above demonstrates several topics and guidelines covered in Module 5. Assignment05 requirements outlined in Steps 5.1- 5.5 were successfully carried out and documented in this report. A script was created using PyCharm, to manage a “To-do List per user input. The script uses a printed "menu" to guide the user through various options for the “To-do List”, including displaying current list, adding a new item, removing an existing item, saving data to the original file, and exiting the program. The "To-do List" file contains two columns of data, "Task" and "Priority." The columns are loaded into a Python dictionary object. Each dictionary object represents one row of data, and these rows are added to a Python list object to create a table of data. New tools and concepts introduced in Module 5, such as collections of data, for loops, while loops, open, write, and close methods for using files were used in accomplishing this task. Throughout the script, comments were used to describe the functions and intents of the applicable set of codes.