Name: Megan (Megs) Cambra
Date: May 17th, 2023
Class: IT FDN 110

Assignment 05
Dive into Productivity: The To-Do List Program

Introduction

In the vast ocean of tasks and responsibilities, staying organized can feel like ***deep-sea exploration***. But fear not, fellow coder! The To-Do List program is here to help you navigate through the depths of your tasks and priorities with ease. This innovative program allows you to create, manage, and conquer your to-do list like a coding diver, popping task bubbles as you go. Get ready to dive into efficient task management and experience a wave of productivity!

Simplify Your Task Management!

The To-Do List program revolves around a Python script that interacts with a file called "`ToDoList.txt`." The script loads each task and its corresponding priority from the file, storing them in a list of *dictionaries* called "`lstTable`." A *dictionary* provides a way to organize and represent structured data, making it easier to access and manipulate information based on specific identifiers rather than their positions in a data structure. Each *dictionary* represents a task, with the keys "Task" and "Priority" indicating the task name and its priority level, respectively.

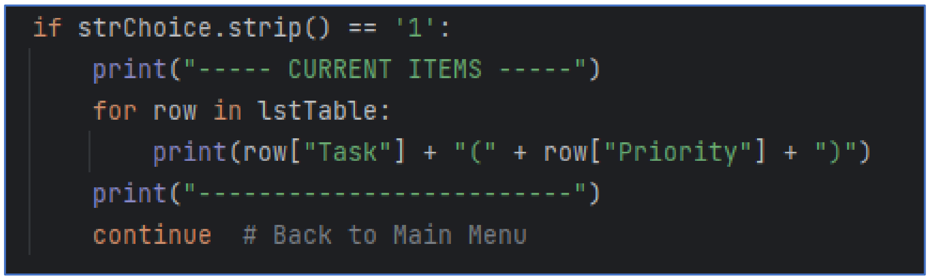
To enhance user experience, the program presents a fun and dynamic menu with five options:

1. Display current data
2. Add a new item
3. Remove an existing item
4. Save Data to File
5. Exit Program

We'll deep dive into each option in the following paragraphs.

I. Task Depths: Discover and Display

The To-Do List program provides a convenient option to display current tasks and their priorities. By using an `if` *statement* and a `for` *loop*, such as the code snippet in the following Figure, you can easily retrieve and present your tasks in a user-friendly format.



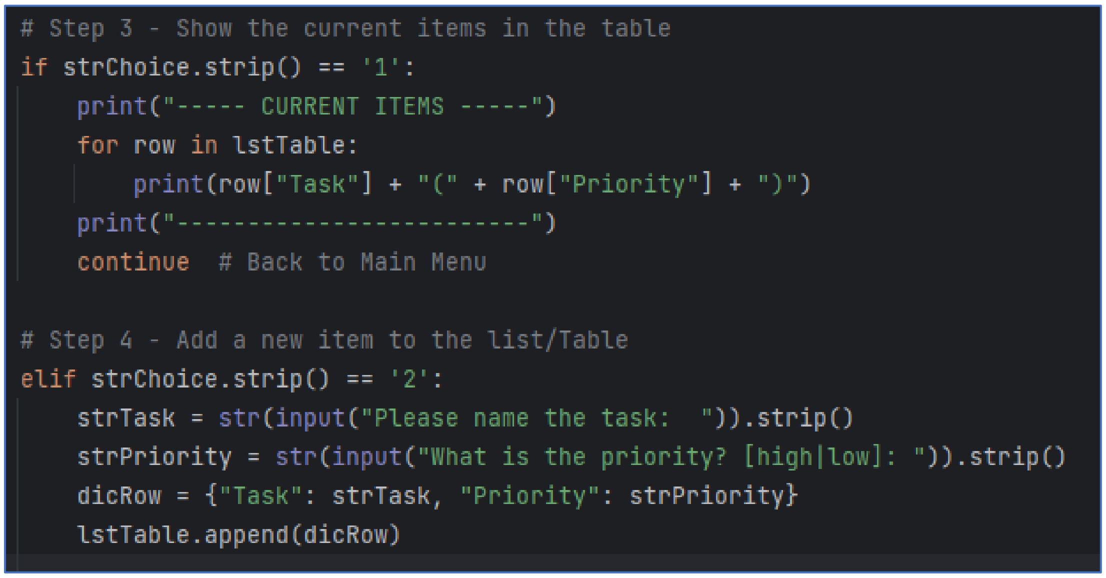
```
if strChoice.strip() == '1':
    print("----- CURRENT ITEMS -----")
    for row in lstTable:
        print(row["Task"] + "(" + row["Priority"] + ")")
    print("------------------------")
    continue  # Back to Main Menu
```

Figure 1. *Displaying the current tasks and priorities in the to-do list. The code iterates through the* `lstTable` *and prints each task along with its corresponding priority.*

With this functionality, you can have a comprehensive overview of your tasks, helping you plan and prioritize effectively.

II. Coding Splash: Adding New "Tasks"

Adding new tasks to your list is effortless with the To-Do List program. Through a prompt-based input system, you can easily provide the task name and its priority level. The inputs are then appended as a new *dictionary* row. Figure 2, below, shows an example from the code that demonstrates how to execute this.



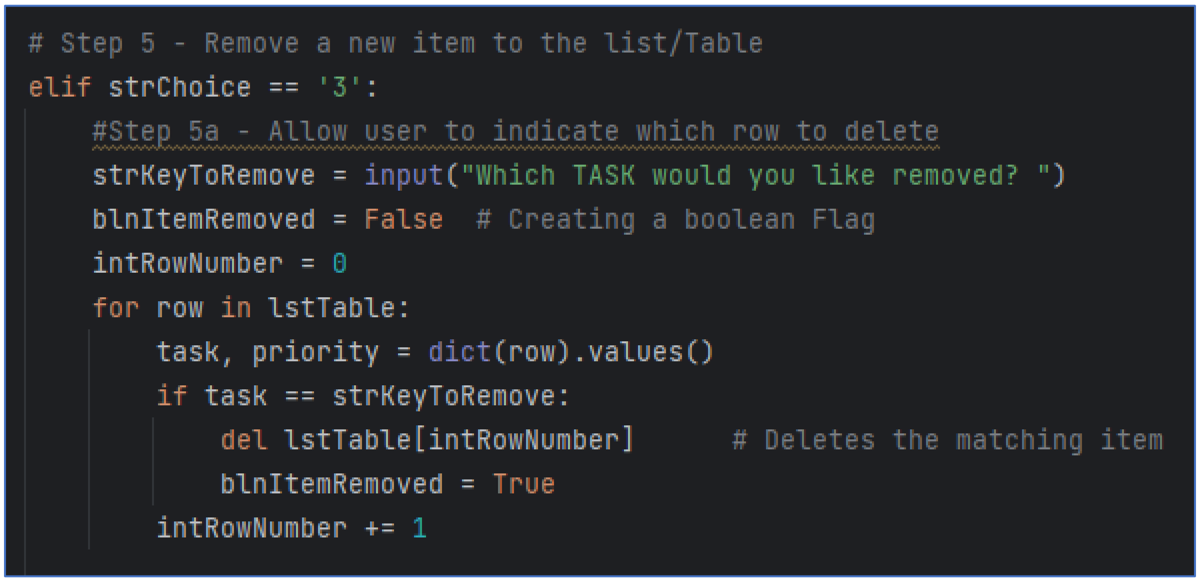
```
# Step 3 - Show the current items in the table
if strChoice.strip() == '1':
    print("----- CURRENT ITEMS -----")
    for row in lstTable:
        print(row["Task"] + "(" + row["Priority"] + ")")
    print("------------------------")
    continue  # Back to Main Menu

# Step 4 - Add a new item to the list/Table
elif strChoice.strip() == '2':
    strTask = str(input("Please name the task:  ")).strip()
    strPriority = str(input("What is the priority? [high|low]: ")).strip()
    dicRow = {"Task": strTask, "Priority": strPriority}
    lstTable.append(dicRow)
```

Figure 2. *Adding an* `elif` *to the* `if` *statement to coordinate Option '2', receiving input.*

By incorporating this feature into the program, you can seamlessly integrate new tasks and ensure that nothing important slips through the cracks.

III. Deep Dive Deletion: Wave Goodbye to Completed Tasks!

Mistakes happen, and completed tasks deserve recognition. A new skill learned in this lesson assignment is incorporating the ability to delete/remove. The To-Do List program offers a hassle-free way to remove tasks from your list. By entering the name of the task, you want to remove, the program efficiently locates and deletes it from the list. Let's dive deeper into how this code snippet, Figure 3, works.



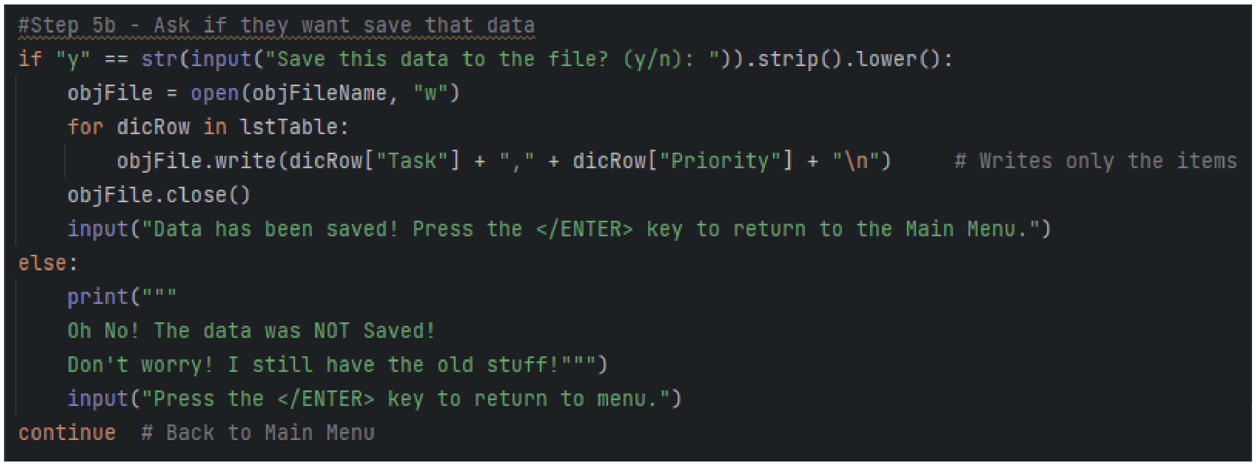
```
# Step 5 - Remove a new item to the list/Table
elif strChoice == '3':
    #Step 5a - Allow user to indicate which row to delete
    strKeyToRemove = input("Which TASK would you like removed? ")
    blnItemRemoved = False  # Creating a boolean Flag
    intRowNumber = 0
    for row in lstTable:
        task, priority = dict(row).values()
        if task == strKeyToRemove:
            del lstTable[intRowNumber]      # Deletes the matching item
            blnItemRemoved = True
        intRowNumber += 1
```

Figure 3. Prompts the user to enter the name of the task they want to remove, searches for a matching task in the `lstTable`, and removes it if found.

In this part of the code, the program prompts you to enter the name of the task you want to remove. It then iterates through each row in the `lstTable` to find a match. If a matching task is found, it uses the `del` statement to remove that specific row from the list. The `blnItemRemoved` variable keeps track of whether an item was successfully removed and informs the user if the item was deleted or not. By employing this code, you can effortlessly eliminate completed or erroneous tasks from your list, keeping it clean and up-to-date.

IV. Save Data Submersion: Treasure Your Tasks

Using the skills learned from the previous lesson assignment, preserving your valuable task data is essential, even when you close the program. The To-Do List program provides a convenient "`Save Data to File`" option.



```
#Step 5b - Ask if they want save that data
if "y" == str(input("Save this data to the file? (y/n): ")).strip().lower():
    objFile = open(objFileName, "w")
    for dicRow in lstTable:
        objFile.write(dicRow["Task"] + "," + dicRow["Priority"] + "\n")     # Writes only the items
    objFile.close()
    input("Data has been saved! Press the </ENTER> key to return to the Main Menu.")
else:
    print("""
    Oh No! The data was NOT Saved!
    Don't worry! I still have the old stuff!""")
    input("Press the </ENTER> key to return to menu.")
continue  # Back to Main Menu
```

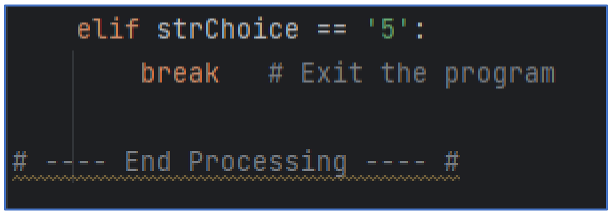
Figure 4. Prompts the user if they want to save the data, and if confirmed, it opens the file in write mode and writes each task and priority from the lstTable into the file.

By leveraging file handling techniques, the program allows you to store all your tasks and priorities in a file, such as "`ToDoList.txt`." This feature ensures that your important task information is securely saved for future reference.

V. Smooth Sailing: Exit the Program

When you're done with your to-do list for now, you can choose the "Exit Program" option. Option 5, to gracefully exit the To-Do List program. The program executes the `break` statement, which terminates the *loop* and exits the program. This ensures a smooth and seamless transition as you conclude your task management journey.

Below is an example using a code snippet that demonstrates how the program handles the exit functionality:

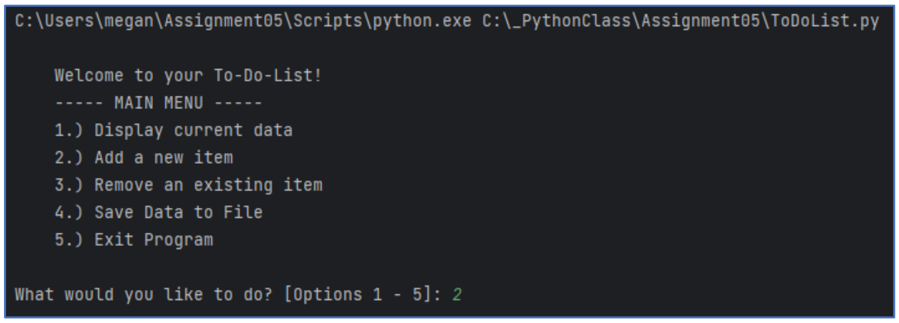


```
    elif strChoice == '5':
        break   # Exit the program

# ---- End Processing ---- #
```

Figure 5. *When the user chooses the exit option, the program executes the* `break` *statement, which terminates the loop and ends the program execution.*

Diving into the Code and Results

When you take the plunge and run the code, a world of to-do list management unfolds before you. Get ready to dive into the various options that await! Whether you want to display your current tasks and priorities, add a new task with its priority, remove an existing task, save your data, or gracefully exit the program, this code has you covered. With each interaction, the program provides insightful results, showing you the current state of your tasks, confirming successful additions or removals, and even displaying the saved data if you choose to store it. So put on your coding gear, immerse yourself in the program, and experience the thrill of efficient to-do list management.



```
C:\Users\megan\Assignment05\Scripts\python.exe C:\_PythonClass\Assignment05\ToDoList.py

    Welcome to your To-Do-List!
    ----- MAIN MENU -----
    1.) Display current data
    2.) Add a new item
    3.) Remove an existing item
    4.) Save Data to File
    5.) Exit Program

What would you like to do? [Options 1 - 5]: 2
```

Figure 6a. *The To-Do-List program runs. It displays the Main Menu and waits for the user's input on which command/option they would like to execute.*

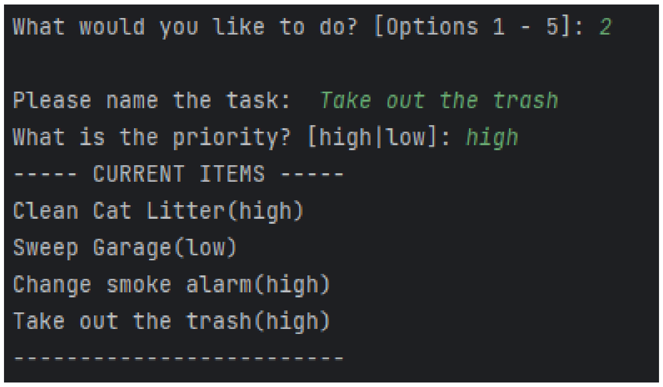

```
What would you like to do? [Options 1 - 5]: 2

Please name the task:  Take out the trash
What is the priority? [high|low]: high
----- CURRENT ITEMS -----
Clean Cat Litter(high)
Sweep Garage(low)
Change smoke alarm(high)
Take out the trash(high)
-------------------------
```

Figure 6b. *Running option 2 from the Main Menu to add a task to the file. The updated list is then displayed for the user.*

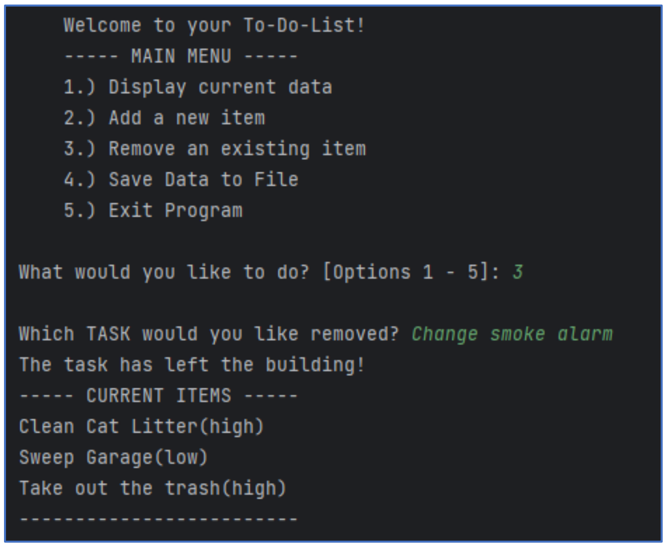

```
    Welcome to your To-Do-List!
    ----- MAIN MENU -----
    1.) Display current data
    2.) Add a new item
    3.) Remove an existing item
    4.) Save Data to File
    5.) Exit Program

What would you like to do? [Options 1 - 5]: 3

Which TASK would you like removed? Change smoke alarm
The task has left the building!
----- CURRENT ITEMS -----
Clean Cat Litter(high)
Sweep Garage(low)
Take out the trash(high)
-------------------------
```

Figure 6c. *Main Menu appears again. Now the user executes the command to remove one of the tasks from the file. The file is updated and displayed.*

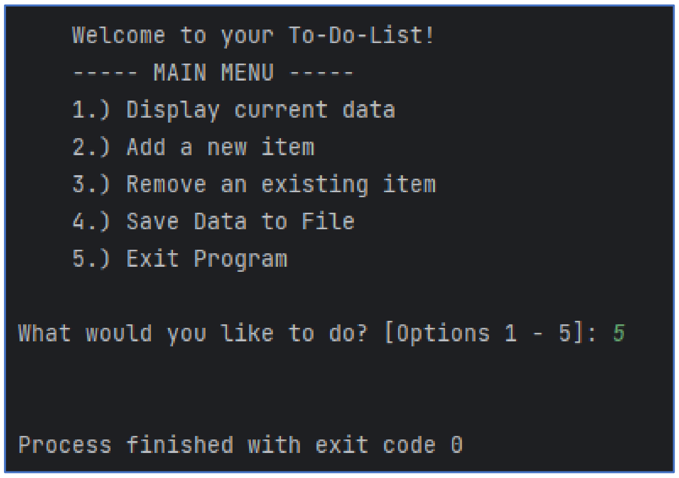
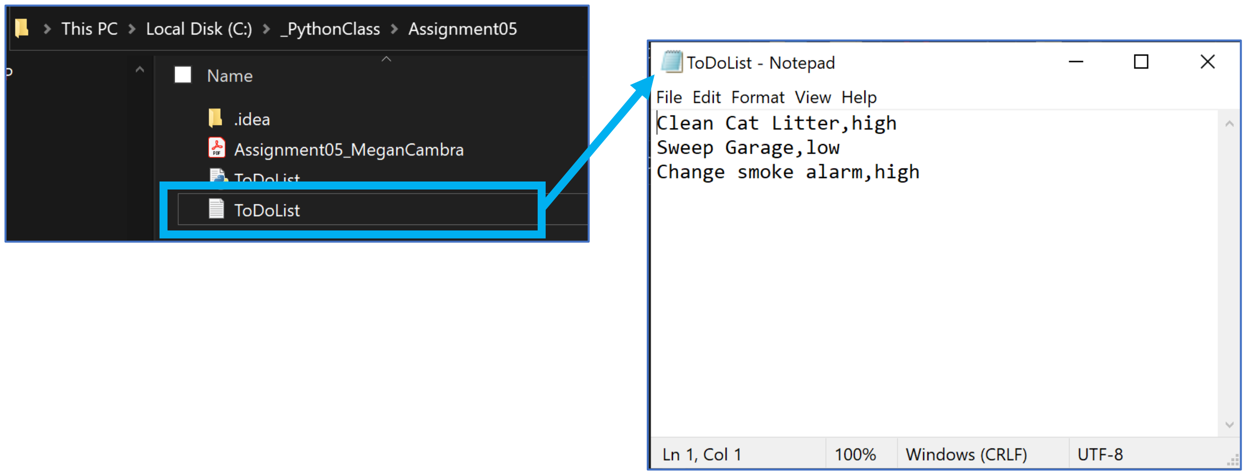

```
    Welcome to your To-Do-List!
    ----- MAIN MENU -----
    1.) Display current data
    2.) Add a new item
    3.) Remove an existing item
    4.) Save Data to File
    5.) Exit Program

What would you like to do? [Options 1 - 5]: 5


Process finished with exit code 0
```

Figure 6d. *Again, prompted by the Main Menu. The user is done and chooses option 5 to exit the To-Do-List program.*



```
Clean Cat Litter,high
Sweep Garage,low
Change smoke alarm,high
```

Figure 6e. *Results of running the code and the data that ended up in the ToDoList.txt file.*

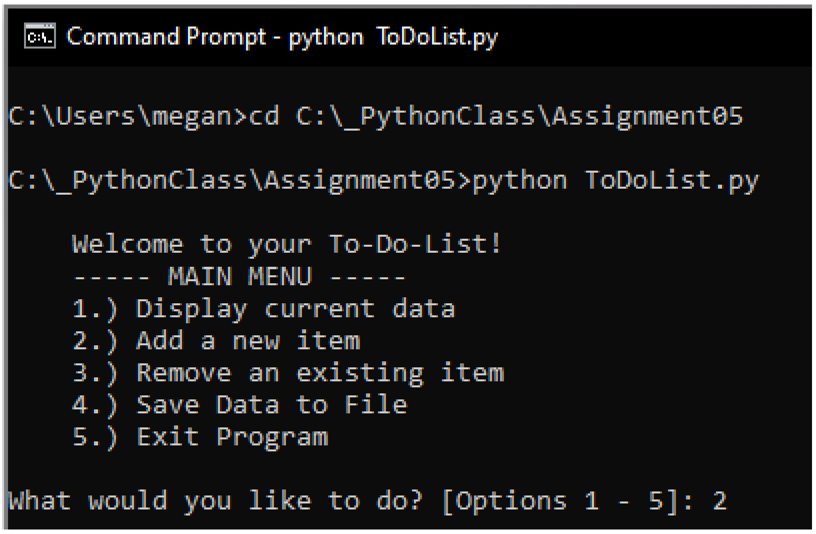
Summary

Surfacing from the Coding Abyss:

In the vast depths of coding, *dictionaries* emerge as indispensable tools that enable efficient data organization and access. Like expert divers exploring the ocean depths, we harness the power of *dictionaries* to navigate through information with grace and precision. These dynamic structures act as our diving gear, offering a flexible and effective means to organize data through key-value pairs.

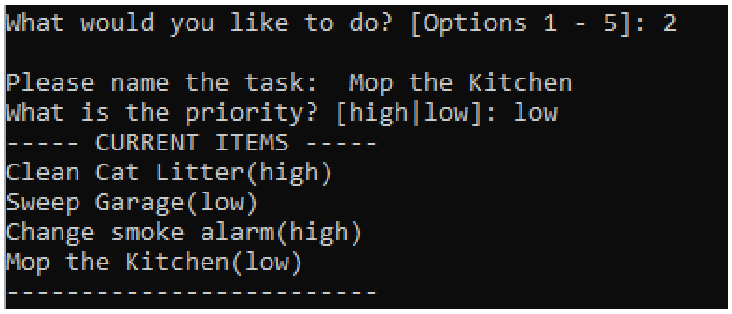
Just as skilled divers, pair meaningful identifiers (or "keys") with their corresponding values, *dictionaries* allow us to swiftly retrieve data and resurface with the information we seek. Whether managing to-do lists or delving into other coding adventures, leveraging *dictionaries* equips us with the ability to dive deep, streamline data management, and explore the boundless depths of coding possibilities. So, gear up, take the plunge, and witness the wonders that *dictionaries* unveil in our coding expeditions.

APPENDIX I. Command Prompt



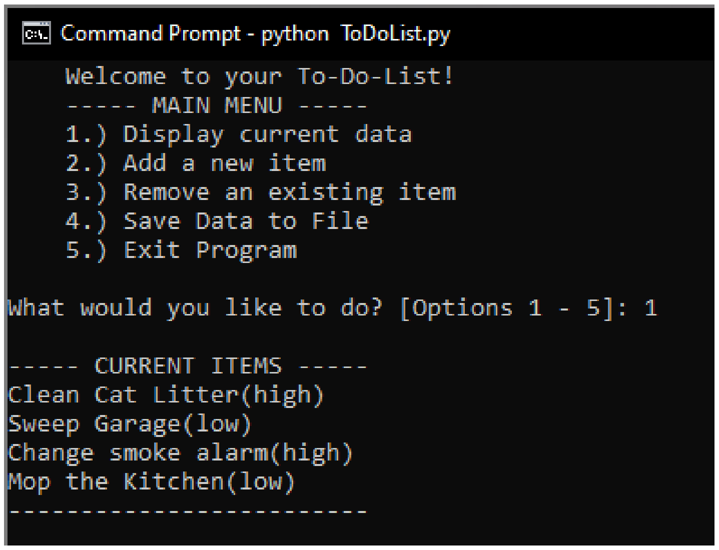
```
Command Prompt - python  ToDoList.py

C:\Users\megan>cd C:\_PythonClass\Assignment05

C:\_PythonClass\Assignment05>python ToDoList.py

    Welcome to your To-Do-List!
    ----- MAIN MENU -----
    1.) Display current data
    2.) Add a new item
    3.) Remove an existing item
    4.) Save Data to File
    5.) Exit Program

What would you like to do? [Options 1 - 5]: 2
```

Figure 7a. Changing directory and running the To-Do-List program using the Command Prompt window instead of PyCharm.



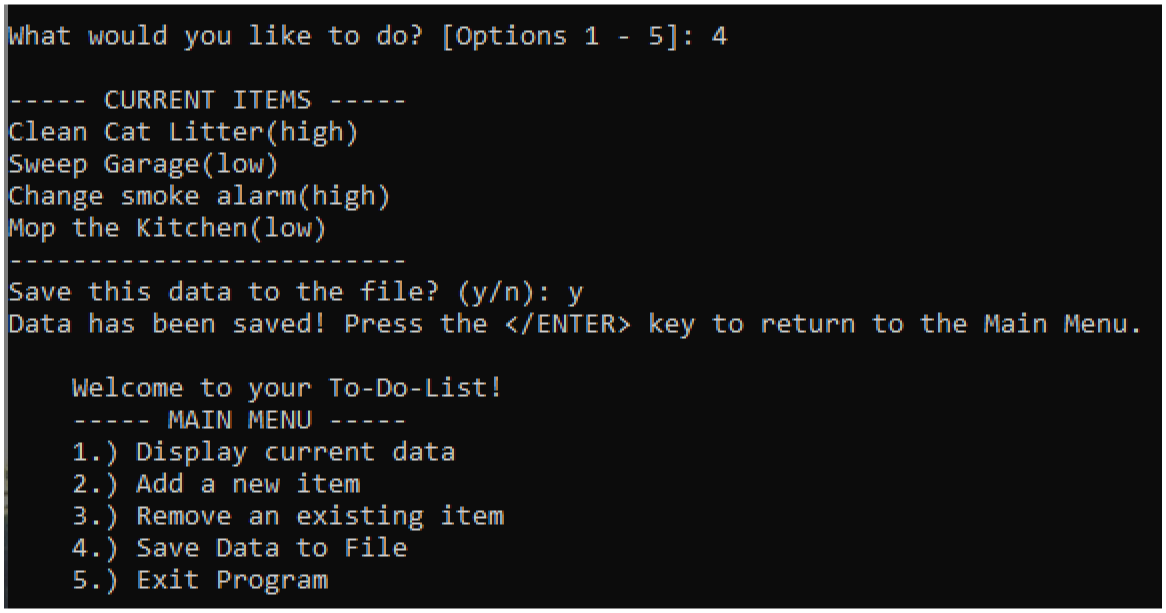
```
What would you like to do? [Options 1 - 5]: 2

Please name the task:  Mop the Kitchen
What is the priority? [high|low]: low
----- CURRENT ITEMS -----
Clean Cat Litter(high)
Sweep Garage(low)
Change smoke alarm(high)
Mop the Kitchen(low)
-------------------------
```

Figure 7b. Testing out option 2 using the Command Prompt.



```
Command Prompt - python  ToDoList.py
    Welcome to your To-Do-List!
    ----- MAIN MENU -----
    1.) Display current data
    2.) Add a new item
    3.) Remove an existing item
    4.) Save Data to File
    5.) Exit Program

What would you like to do? [Options 1 - 5]: 1

----- CURRENT ITEMS -----
Clean Cat Litter(high)
Sweep Garage(low)
Change smoke alarm(high)
Mop the Kitchen(low)
-------------------------
```

Figure 7c. Testing out option 1 using the Command Prompt.



```
What would you like to do? [Options 1 - 5]: 4

----- CURRENT ITEMS -----
Clean Cat Litter(high)
Sweep Garage(low)
Change smoke alarm(high)
Mop the Kitchen(low)
-------------------------
Save this data to the file? (y/n): y
Data has been saved! Press the </ENTER> key to return to the Main Menu.

    Welcome to your To-Do-List!
    ----- MAIN MENU -----
    1.) Display current data
    2.) Add a new item
    3.) Remove an existing item
    4.) Save Data to File
    5.) Exit Program
```

Figure 7d. *Testing out option 4 using Command Prompt.*

APPENDIX II. Python Script of "ToDoList.py"

```python
# ------------------------------------------------------------------------ #
# 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 each dictionary "row" to a python list "table"
# ChangeLog (Who,When,What):
# RRoot,1.1.2030,Created started script
# MCambra,5.17.2023,Added code per each section prompt
# ------------------------------------------------------------------------ #

# ---- Data / Main Variables ---- #
# Declare variables and constants
objFileName = "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 dictionary that acts as a 'table' of rows
strChoice = ""  # Capture the user option selection

# ---- Begin Processing ---- #
# Step 1 - When the program starts, Load from ToDoFile.txt into a python
Dictionary.
objFile = open(objFileName, "r")
for line in objFile:
    strData = line.split(",")
    dicRow = {"Task": strData[0].strip(), "Priority": strData[1].strip()}
    lstTable.append(dicRow)
objFile.close()

# Step 2 - Display a menu of choices to the user
```

```
while True:
    print("""
    Welcome to your To-Do-List!
    ----- MAIN MENU -----
    1.) Display current data
    2.) Add a new item
    3.) Remove an existing item
    4.) Save Data to File
    5.) Exit Program
    """)
    strChoice = str(input("What would you like to do? [Options 1 - 5]: "))

    print()  # Blank line

    # Step 3 - Show the current items in the table
    if strChoice.strip() == '1':
        print("----- CURRENT ITEMS -----")
        for row in lstTable:
            print(row["Task"] + "(" + row["Priority"] + ")")
        print("-------------------------")
        continue  # Back to Main Menu

    # Step 4 - Add a new item to the list/Table
    elif strChoice.strip() == '2':
        strTask = str(input("Please name the task:  ")).strip()
        strPriority = str(input("What is the priority? [high|low]:
")).strip()
        dicRow = {"Task": strTask, "Priority": strPriority}
        lstTable.append(dicRow)

        #Step 4a - Show the current items in the table
        print("----- CURRENT ITEMS -----")
        for row in lstTable:
            print(row["Task"] + "(" + row["Priority"] + ")")
        print("-------------------------")
        continue  # Back to Main Menu

    # Step 5 - Remove a new item to the list/Table
    elif strChoice == '3':
        #Step 5a - Allow user to indicate which row to delete
        strKeyToRemove = input("Which TASK would you like removed? ")
        blnItemRemoved = False  # Creating a boolean Flag
        intRowNumber = 0
        for row in lstTable:
            task, priority = dict(row).values()
            if task == strKeyToRemove:
                del lstTable[intRowNumber]      # Deletes the matching item
                blnItemRemoved = True
            intRowNumber += 1

        #Step 5b - Update user on the status
        if blnItemRemoved == True:
            print("The task has left the building!")
        else:
            print("Sorry, I could not find that task. :(")

        #Step 5c - Show the current items in the table
        print("----- CURRENT ITEMS -----")
        for row in lstTable:
            print(row["Task"] + "(" + row["Priority"] + ")")
        print("-------------------------")
        continue  # Back to Main Menu

    # Step 6 - Save tasks to the ToDoFile.txt file
    elif strChoice == '4':
        #Step 5a - Show the current items in the table
        print("----- CURRENT ITEMS -----")
        for row in lstTable:
            print(row["Task"] + "(" + row["Priority"] + ")")
        print("-------------------------")

        #Step 5b - Ask if they want save that data
        if "y" == str(input("Save this data to the file? (y/n): 
")).strip().lower():
            objFile = open(objFileName, "w")
            for dicRow in lstTable:
                objFile.write(dicRow["Task"] + "," + dicRow["Priority"] + 
"\n")     # Writes only the items
            objFile.close()
            input("Data has been saved! Press the </ENTER> key to return to 
the Main Menu.")
        else:
            print("""
            Oh No! The data was NOT Saved!
            Don't worry! I still have the old stuff!""")
            input("Press the </ENTER> key to return to menu.")
        continue  # Back to Main Menu

    elif strChoice == '5':
        break   # Exit the program

# ---- End Processing ---- #
```