Ashton Larkin
May 24, 2022
IT FDN 110 A Spring 2022: Foundations of Programming: Python
Assignment06: "To Do List"
https://github.com/AshtonUniverse/IntroToProg-Python-Mod06
https://ashtonuniverse.github.io/IntroToProg-Python-Mod06/

Introduction:

The objective of assignment 06 is to update a provided starter script adding code to the script sections of each step with a comment "# *TODO: Add Code Here*" and to maintain the integrity of the existing logic and formatting of the original programmer. Assignment 06 expands on assignment 05's To Do List program adding functions and classes.

The "ToDoList" program (Assignment06_Starter.py) is divided into four separations of concerns: Data, Processing, Presentation (Input-Output) and Main Body of Script. The script performs the following steps (pseudocode) as scripted by the original programmer "RRoot":

Step 1 - When the program starts, Load data from ToDoFile.txt.
Step 2 - Display a menu of choices to the user.
Step 3 - Show current data.
Step 4 - Process user's menu choice:
Add a new Task
Remove an existing Task
Save Data to File
Exit Program

This document is a breakdown of the logic I used for assignment 06. I will list each step highlighting original code from the starter script and code I added or modified to complete the program.

Added my name and date to the change log in the script header:

```
# ------------------------------------------------------------------------ #
# Title: Assignment 06
# Description: Working with functions in a class,
#              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
# ALarkin,5.24.2022,Modified code to complete assignment 06
# ------------------------------------------------------------------------ #
```

Replaced:

```
# <Your Name>,<Date>,Modified code to complete assignment 06
```

```
# Data ---------------------------------------------------------------------- #
# Declare variables and constants
file_name_str = "ToDoList.txt"  # The name of the data file
file_obj = None  # An object that represents a file
row_dic = {}  # A row of data separated into elements of a dictionary {Task,Priority}
table_lst = []  # A list that acts as a 'table' of rows
choice_str = ""  # Captures the user option selection
```

In the data section, I changed the file name from "ToDoFile.txt" to "ToDoList.txt.

```
# Processing  ---------------------------------------------------------------- #
class Processor:
    """  Performs Processing tasks """

    @staticmethod
    def os_path_isfile(file_name):
        """ Validates if file_name exists

        :param file_name: (string) with name of file:
        :return: isfile_bln
        """
        import os.path
        isfile_bln = (os.path.isfile(file_name))
        return (isfile_bln)
```

I added a new function "os_path_isfile" that performs a validation on the file_name parameter passed to the function and returns a Boolean value (True or False). This function also calls the os. path module. It is useful when processing files from different places in the system and for different purposes such as for merging, normalizing, and retrieving path names in python

```
    @staticmethod
    def read_data_from_file(file_name, list_of_rows):
        """ Reads data from a file into a list of dictionary rows
        :param file_name: (string) with name of file:
        :param list_of_rows: (list) you want filled with file data:
        :return: (list) of dictionary rows
        """
        list_of_rows.clear()  # clear current data
        file = open(file_name, "r")
        for line in file:
            task, priority = line.split(",")
            row = {"Task": task.strip(), "Priority": priority.strip()}
            list_of_rows.append(row)
        file.close()
        return list_of_rows
```

```
    @staticmethod
    def add_data_to_list(task, priority, list_of_rows):
        """ Adds data to a list of dictionary rows
        :param task: (string) with name of task:
        :param priority: (string) with name of priority:
        :param list_of_rows: (list) you want filled with file data:
        :return: (list) of dictionary rows
        """
        row = {"Task": str(task).strip(), "Priority": str(priority).strip()}
        # TODO: Add Code Here!
        priority_options = ['High', 'Medium', 'Low']  # Task priority options
        while True:
            if (task != '' and priority in priority_options):
                list_of_rows.append(row)
            elif (task == '' and priority in priority_options):
                print()  # Add an extra line for looks
                print('Empty task entered.')
            elif (task != '' and priority not in priority_options):
                print()  # Add an extra line for looks
                print('Invalid priority entered')
            elif (task == '' and priority not in priority_options):
                print()  # Add an extra line for looks
                print('Empty task and invalid priority entered.')
            else:
                break
            return list_of_rows
```

In the "add_data_to_list" function, I added a loop to validate all combinations for a valid task entry (non-empty) and priority (High, Medium, Low) from the list variable "priority_options". If the user enters a valid task and priority, the row is appended to the list table "list_of_rows" which is returned by the function. If the user enters an empty task and/or invalid priority, a print() statement returns either "Empty task entered", "Invalid priority entered" or "Empty task and invalid priority entered".

```
    @staticmethod
    def remove_data_from_list(task, list_of_rows):
        """ Removes data from a list of dictionary rows
        :param task: (string) with name of task:
        :param list_of_rows: (list) you want filled with file data:
        :return: (list) of dictionary rows
        """
        # TODO: Add Code Here!
        remove_bln = False  # Use this to verify that the data was found and removed
        for row in list_of_rows:
            if (row["Task"] == task):
                list_of_rows.remove(row)
                remove_bln = True
        # Update user on the status
        if remove_bln == True:
            print()  # Add an extra line for looks
            print(task + " removed from ToDoList. ")
        else:
            print()  # Add an extra line for looks
            print(task + " is not in ToDoList. ")
        return list_of_rows
```

In the "remove_data_from_list" function, I added a Boolean variable "remove_bln" to verify if the task exists in the list table "list_of_rows". If remove_bln evaluates to "True", the task is removed from the list table, and it prints the task was removed from ToDoList. If remove_bln evaluates to "False", the task does not exist, and it prints the task is not in ToDoList.

```
    @staticmethod
    def write_data_to_file(file_name, list_of_rows):
        """ Writes data from a list of dictionary rows to a File
        :param file_name: (string) with name of file:
        :param list_of_rows: (list) you want filled with file data:
        :return: (list) of dictionary rows
        """
        # TODO: Add Code Here!
        objFile = open(file_name, "w")
        for row in list_of_rows:
            objFile.write(row["Task"] + "," + row["Priority"] + "\n")
        objFile.close()
        print()  # Add an extra line for looks
        print("Data saved to file: " + "[" + file_name + "]")
        return list_of_rows
```

In the "write_data_to_file" function, I added an open() function and a for loop to write all rows from the list table "list_of_rows" to the delimited text file and print "Data saved to file: [ToDoList.txt].

```
# Presentation (Input/Output)  -------------------------------------- #

class IO:
    """ Performs Input and Output tasks """
    @staticmethod
    def output_menu_tasks():
        """  Display a menu of choices to the user

        :return: nothing
        """
        print('''
        ****************************
        ToDoList - Menu of Options
        ****************************
        1) Show current data
        2) Add a new Task
        3) Remove an existing Task
        4) Save Data to File
        5) Exit Program
        ****************************
        ''')
        print()  # Add an extra line for looks
```

I modified the display menu name, added "Show current data" as option 1 and renumbered the menu options to match assignment 05. I also added border lines (****) for looks.

```
    @staticmethod
    def input_menu_choice():
        """ Gets the menu choice from a user
        :return: string
        """
        choice = str(input("Which option would you like to perform? [1 to 5] - ")).strip()
        print()  # Add an extra line for looks
        return choice

    @staticmethod
    def output_current_tasks_in_list(list_of_rows):
        """ Shows the current Tasks in the list of dictionaries rows
        :param list_of_rows: (list) of rows you want to display
        :return: nothing
        """
        print("******* The current tasks ToDo are: *******")
        for row in list_of_rows:
            print(row["Task"] + " (" + row["Priority"] + ")")
        print("*******************************************")
        print()  # Add an extra line for looks
```

```
    @staticmethod
    def input_new_task_and_priority():
        """  Gets task and priority values to be added to the list
        :return: (string, string) with task and priority
        """
        # TODO: Add Code Here!
        task = str(input("Enter a Task: ").strip().lower().title())
        priority = str(input("Enter a Priority: [High, Medium, Low] - ").strip().lower().title())
        return (task, priority)
```

In the "input_new_task_and_priority" function, I added input() statements that asks the user to enter a new task and its priority. I used the strip() method to remove whitespace and characters from the beginning and the end of the strings, the lower() method to return strings where all characters are lower case, and the title() method to return a string where the first character in every word is upper case. This ensures consistency in the data is stored, presented, and removed in other steps. The function returns the task and priority.

```
    @staticmethodsss
    def input_task_to_remove():
        """  Gets the task name to be removed from the list
        :return: (string) with task
        """
        # TODO: Add Code Here!
        task = str(input("Enter a task to remove from ToDoList: ").strip().lower().title())
        return (task)
```

In the "input_task_to_remove" function, I added an input() statement that asks the user to input a task to remove. As with the "input_new_task_and_priority" function, the strip(), lower(), and title() methods were added to ensure data consistency and validation on row lookup. The task is returned.

```
# Main Body of Script  ------------------------------------------------- #

# Step 1 - When the program starts, Load data from ToDoFile.txt.
Processor.read_data_from_file( file_name=file_name_str, list_of_rows=table_lst)  # read file data

# Step 1 - When the program starts, load data from ToDoFile.txt if file_name exists.
    isfile_bln = Processor.os_path_isfile(file_name=file_name_str)
    if (isfile_bln == True):
        Processor.read_data_from_file(file_name=file_name_str, list_of_rows=table_lst)  # read file data
    else:
        pass
```

When the program begins, step 1 loads data from ToDoList.txt by calling the "read_data_from_file" function which writes existing data to a list table. The logic I added calls a new function that I created, (os_path_isfile) which evaluates if the file exists and loads the data to the list table if it exists or does a pass if it does not. Without this logic, if the file does not exist, the following error will occur: FileNotFoundError: [Errno 2] No such file or directory: 'ToDoList.txt'.

```
# Step 2 - Display a menu of choices to the user
while (True):
    # Step 3 Show current data
    IO.output_current_tasks_in_list(list_of_rows=table_lst)  # Show current data in the list/table
    IO.output_menu_tasks()  # Shows menu
    choice_str = IO.input_menu_choice()  # Get menu option
```

```
while (True):
    # Step 2 - Display a menu of choices to the user
    IO.output_menu_tasks()  # Shows menu
    choice_str = IO.input_menu_choice()  # Get menu option
```

I moved "Step 2 – Display a menu of choices to the user" directly under the "while (True): statement. I also removed "Step 3 Show current data" and its associated function call "IO.output_current_tasks_in_list(list_of_rows=table_lst) # Show current data in the list/table" from this section of the script. This section now only displays the menu of options.

```
    # Step 4 - Process user's menu choice
    if choice_str.strip() == '1':  # Add a new Task
        task, priority = IO.input_new_task_and_priority()
        table_lst = Processor.add_data_to_list(task=task, priority=priority, list_of_rows=table_lst)
        continue  # to show the menu
```

```
    # Step 3 - Process user's menu choice
    if choice_str.strip() == '1':
        IO.output_current_tasks_in_list(list_of_rows=table_lst)  # Show current data in the list/table
        continue  # to show the menu
```

Step 4 was changed to step 3. Choice 1 was changed to process the added menu option "1) Show current data" and all choices after were renumbered to match the modified menu options.

```
    if choice_str.strip() == '2':  # Add a new Task
        task, priority = IO.input_new_task_and_priority()
        table_lst = Processor.add_data_to_list(task=task, priority=priority, list_of_rows=table_lst)
        continue # to show the menu

    elif choice_str == '3':  # Remove an existing Task
        task = IO.input_task_to_remove()
        table_lst = Processor.remove_data_from_list(task=task, list_of_rows=table_lst)
        continue  # to show the menu

    elif choice_str == '4':  # Save Data to File
        table_lst = Processor .write_data_to_file(file_name=file_name_str, list_of_rows=table_lst)
        continue  # to show the menu

    elif choice_str == '5':  # Exit Loop
        print("Goodbye!")
        break  # by exiting loop

# Exit the program
input("\nPress the enter key to exit.")
```

I added a final section that prompts the user with an input() function asking the user to press the "enter" key to exit and close the program.

Run the script from PyCharm.

Assigment06_Starter

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


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

******* The current tasks ToDo are: *******
Exercise (High)
Eat Healthy (High)
Unproductive Activity (High)
Study (Medium)
Downtime (High)
*******************************************


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

Assigment06_Starter

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

Enter a Task: clean house
Enter a Priority: [High, Medium, Low] - low

        ****************************
        ToDoList - Menu of Options
        ****************************
        1) Show current data
        2) Add a new Task
        3) Remove an existing Task
        4) Save Data to File
        5) Exit Program
        ****************************


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

******* The current tasks ToDo are: *******
Exercise (High)
Eat Healthy (High)
Unproductive Activity (High)
Study (Medium)
Downtime (High)
Clean House (Low)
*******************************************
```

Assigment06_Starter

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


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


Data saved to file: [ToDoList.txt]

        ****************************
        ToDoList - Menu of Options
        ****************************
        1) Show current data
        2) Add a new Task
        3) Remove an existing Task
        4) Save Data to File
        5) Exit Program
        ****************************


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

******* The current tasks ToDo are: *******
Exercise (High)
Eat Healthy (High)
Unproductive Activity (High)
Study (Medium)
Downtime (High)
Clean House (Low)
*******************************************
```

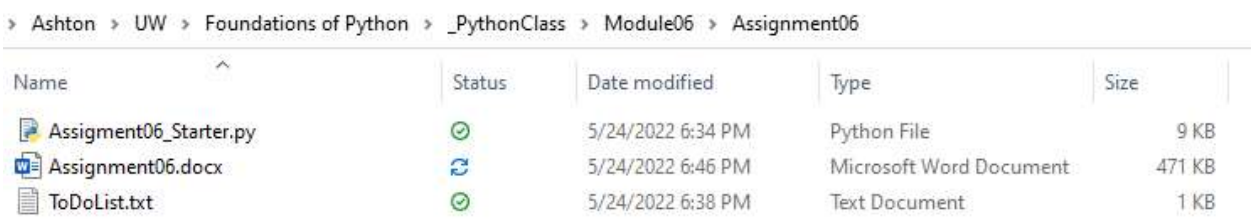
Assigment06_Starter

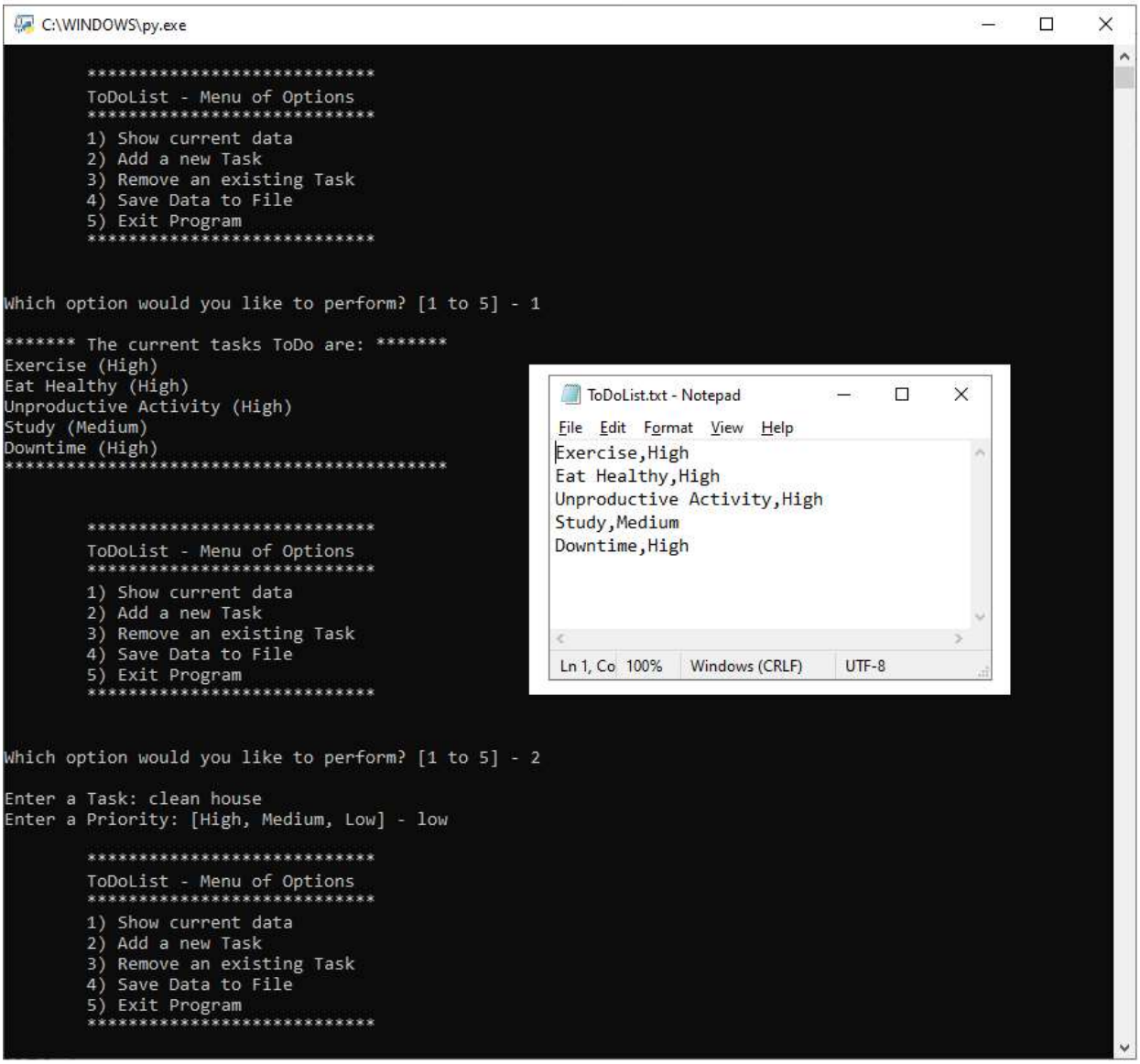
```
        ****************************
        ToDoList - Menu of Options
        ****************************
        1) Show current data
        2) Add a new Task
        3) Remove an existing Task
        4) Save Data to File
        5) Exit Program
        ****************************


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

Enter a task to remove from ToDoList: clean house

Clean House removed from ToDoList.

        ****************************
        ToDoList - Menu of Options
        ****************************
        1) Show current data
        2) Add a new Task
        3) Remove an existing Task
        4) Save Data to File
        5) Exit Program
        ****************************


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

******* The current tasks ToDo are: *******
Exercise (High)
Eat Healthy (High)
Unproductive Activity (High)
Study (Medium)
Downtime (High)
*****************************************
```

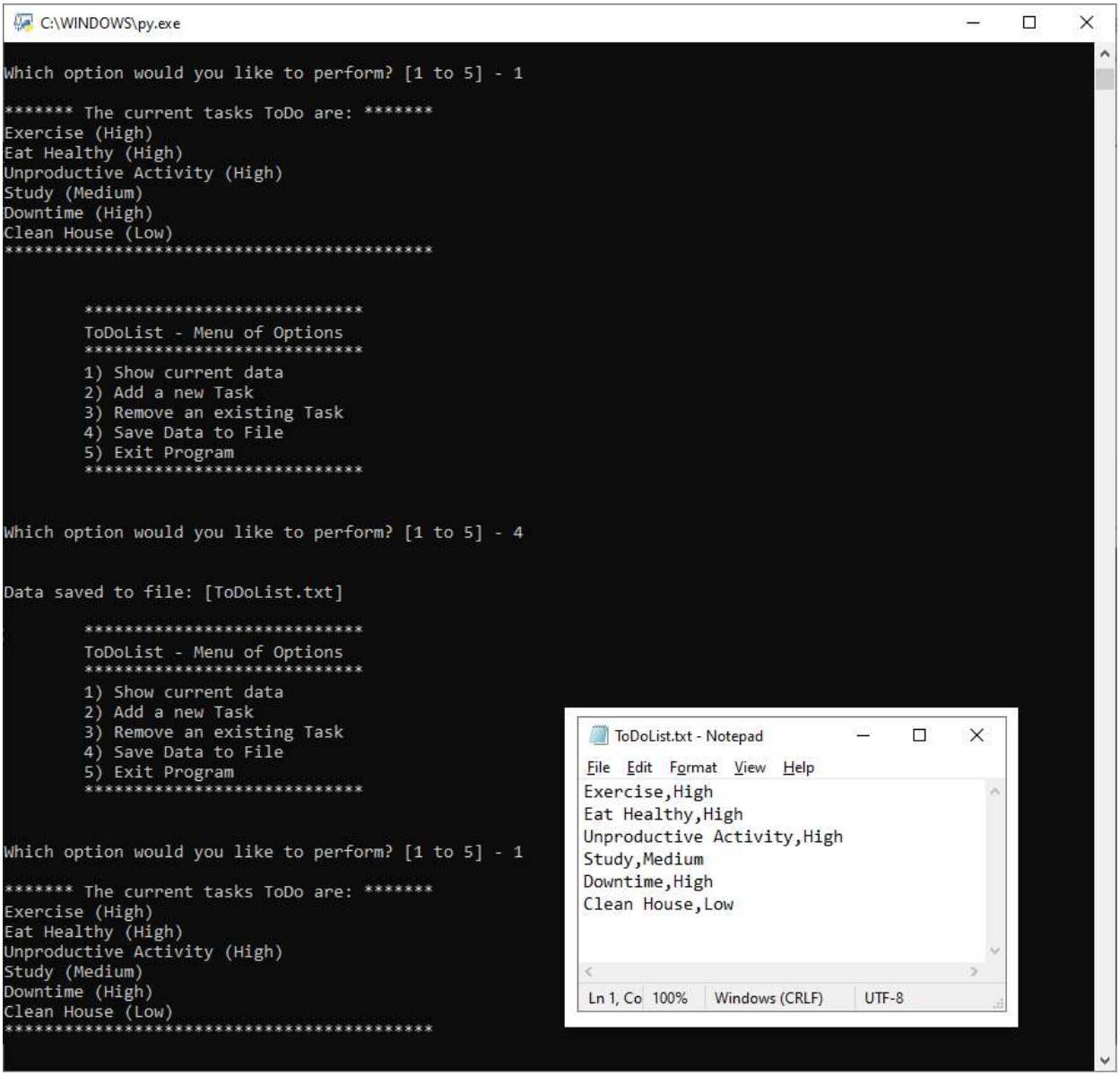
Assigment06_Starter

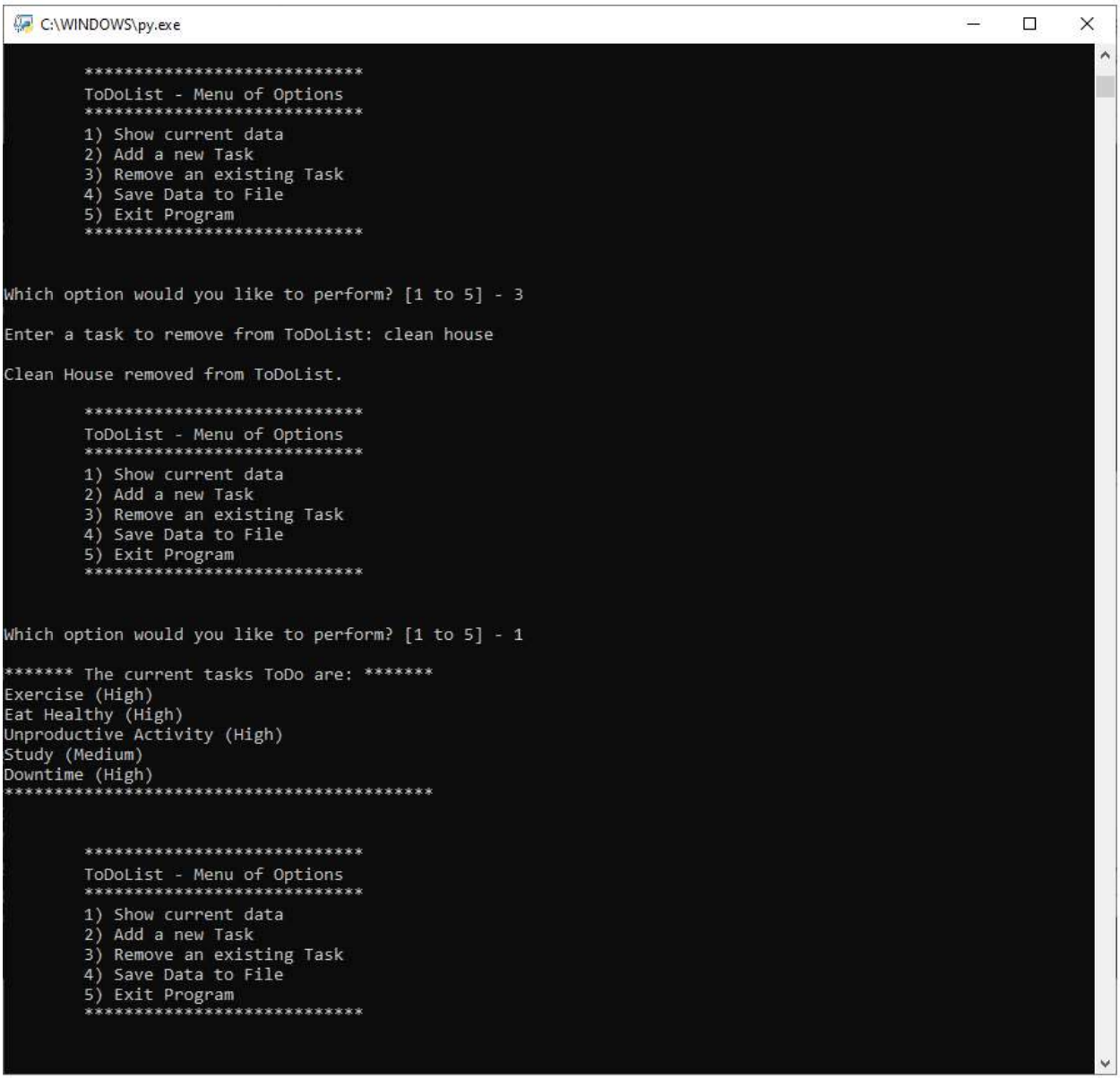
```
**********************************************


        ****************************
        ToDoList - Menu of Options
        ****************************
        1) Show current data
        2) Add a new Task
        3) Remove an existing Task
        4) Save Data to File
        5) Exit Program
        ****************************


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


Data saved to file: [ToDoList.txt]

        ****************************
        ToDoList - Menu of Options
        ****************************
        1) Show current data
        2) Add a new Task
        3) Remove an existing Task
        4) Save Data to File
        5) Exit Program
        ****************************


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

Goodbye!

Press the enter key to exit.

Process finished with exit code 0
```

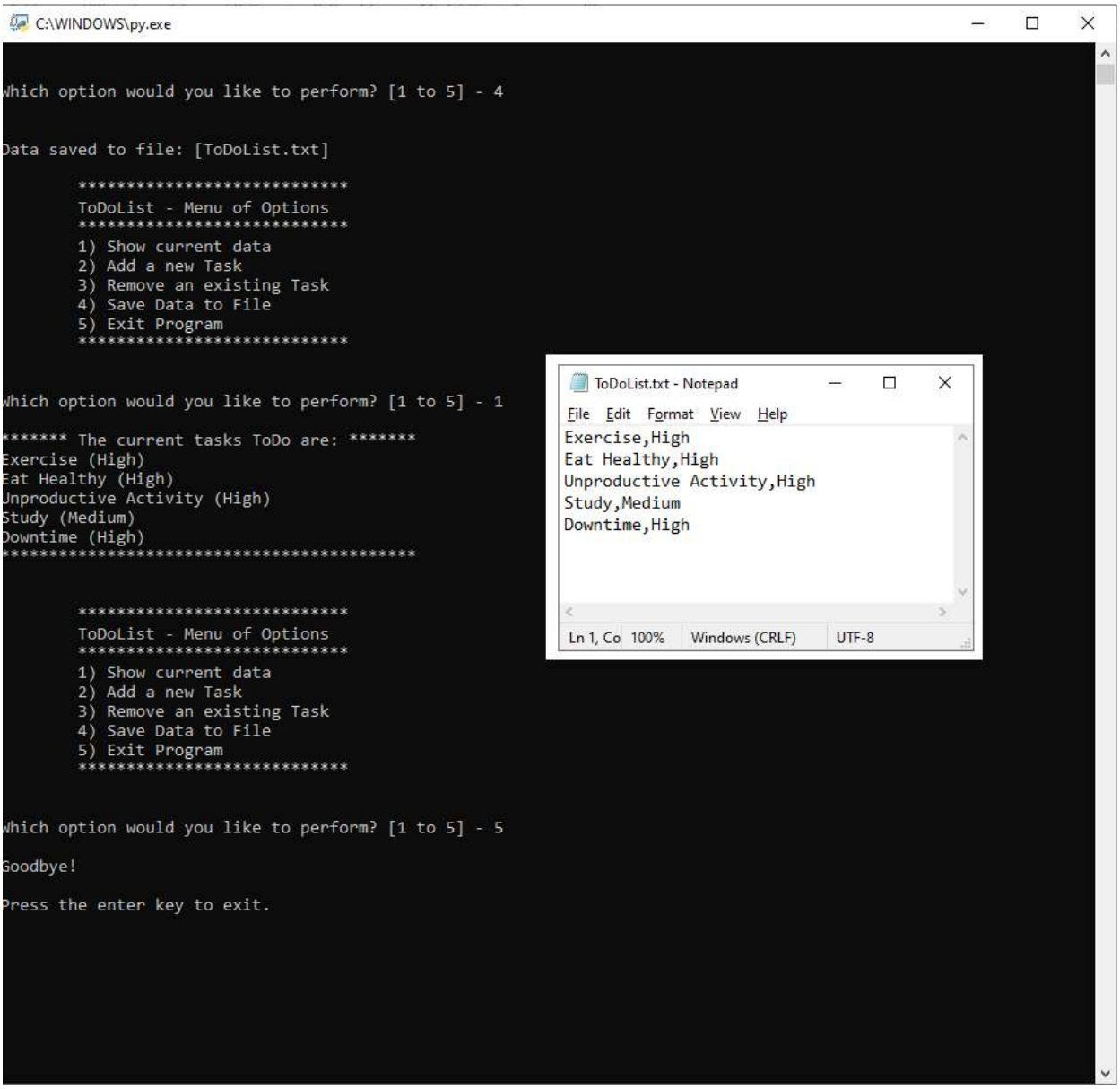
Run the Python Script from the Windows OS Command Shell and verify the data in the ToDoList.txt file.






C:\WINDOWS\py.exe
Which option would you like to perform? [1 to 5] - 1
******* The current tasks ToDo are: *******
Exercise (High)
Eat Healthy (High)
Unproductive Activity (High)
Study (Medium)
Downtime (High)
Clean House (Low)
*****
ToDoList - Menu of Options
1) Show current data
2) Add a new Task
3) Remove an existing Task
4) Save Data to File
5) Exit Program
Which option would you like to perform? [1 to 5] - 4
Data saved to file: [ToDoList.txt]
ToDoList - Menu of Options
1) Show current data
2) Add a new Task
3) Remove an existing Task
4) Save Data to File
5) Exit Program
Which option would you like to perform? [1 to 5] - 1
******* The current tasks ToDo are: *******
Exercise (High)
Eat Healthy (High)
Unproductive Activity (High)
Study (Medium)
Downtime (High)
Clean House (Low)
*****
ToDoList.txt - Notepad
File Edit Format View Help
Exercise,High
Eat Healthy,High
Unproductive Activity,High
Study,Medium
Downtime,High
Clean House,Low
Ln 1, Co 100% Windows (CRLF) UTF-8


C:\WINDOWS\py.exe
****
ToDoList - Menu of Options
****
1) Show current data
2) Add a new Task
3) Remove an existing Task
4) Save Data to File
5) Exit Program
****
Which option would you like to perform? [1 to 5] - 3
Enter a task to remove from ToDoList: clean house
Clean House removed from ToDoList.
****
ToDoList - Menu of Options
****
1) Show current data
2) Add a new Task
3) Remove an existing Task
4) Save Data to File
5) Exit Program
****
Which option would you like to perform? [1 to 5] - 1
******* The current tasks ToDo are: *******
Exercise (High)
Eat Healthy (High)
Unproductive Activity (High)
Study (Medium)
Downtime (High)
****
****
ToDoList - Menu of Options
****
1) Show current data
2) Add a new Task
3) Remove an existing Task
4) Save Data to File
5) Exit Program
****



Summary:

Assignment 06 builds on assignment 05 by organizing code into functions and classes. I modified a provided script that contains starter logic that when completed prompts the user with a menu of options that shows current data, adds a new item, removes an existing item, saves data to a file, and exits the program. This program allows the user to enter a "Tasks" and an associated "Priority" to create a ToDo list. To complete the starter program, I added code to each step in the program at the "TODO" comments. I created a function that validates a file. I updated existing functions with logic for inputs/outputs that prompt the user to enter data that appends to and removes data rows from a list table and writes to and removes data from a delimited text file.