Kate Beverly

August 17, 2022

Foundations of Programming: Python

Assignment 06

https://github.com/kbev12/IntroToProg-Python-Mod06

https://kbev12.github.io/IntroToProg-Python-Mod06/

To Do Script

Introduction

For assignment 06 for the class Foundations of Programming: Python I worked on a python script that had been started previously. The script reads from a file to memory and then gives the user the option to display the information, add a task, remove a task, write to the file, and end the program.

Creating the Script

The script begins with declaring the variables (**Figure 1**). Most of the variables are empty except for file_name_str which is set to the name of the text file and using snake case.

```
# Data ---------------------------------------------------------------------- #
# Declare variables and constants
file_name_str = "ToDoFile.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
```

Figure 1. Declaring the program variables

Class Processor

The first class, Processor (**Figure 2**) contains all the functions that perform the processing tasks. In the first function the script opens the text file in read mode. It splits each row by the comma and creates a dictionary identifying the first object as the task and the second as the priority. Each row is appended to a table and stored it in the system's memory.

```python
class Processor:
    """  Performs Processing tasks """

    @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()}
        list_of_rows.append(row)
        return list_of_rows
```

```python
    @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
        """
        file = open(file_name, "w")
        for row in list_of_rows:
            file.write(row["Task"] + "," + row["Priority"] + "\n")
        file.close()
        return list_of_rows
```

Figure 2. Processing Class – Processing the data

The second function in the Processing class, add_data_to_list appends the row to the list of rows and returns that variable.

The third function, remove_data_from_list iterates through each row and if the task that was passed to the function matches a task in the list of dictionaries the row is removed.

The fourth function write_data_to_file opens the file in write mode and each row stored in memory is written to the file. The file is then closed and list_of_rows is returned.

Class IO

The second class in the script (**Figure 3**), class IO, performs the input and output tasks.

```
class IO:
    """ Performs Input and Output tasks """

    @staticmethod
    def output_menu_tasks():
        """  Display a menu of choices to the user

        :return: nothing
        """
        print('''
        Menu of Options
        1) Add a new Task
        2) Remove an existing Task
        3) Save Data to File
        4) Exit Program
        ''')
        print()  # Add an extra line 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 4] - ")).strip()
        print()  # Add an extra line for looks
        return choice
```

```python
    @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
        """
        task = str(input("What is the task? ")).strip()
        priority = str(input("What is the task's priority? ")).strip()
        return task, priority

    @staticmethod
    def input_task_to_remove():
        """  Gets the task name to be removed from the list

        :return: (string) with task
        """
        task = str(input("What is the name of the task to remove? ")).strip()
        return task
```

Figure 3 – Class IO performing input output tasks

The first function, output_menu_tasks, in class IO prints the menus of options the user can select.

The second fuction input_menu_choice requests the user to input which of the four choices from the menu they would like to perform. The function returns the choice variable.

Output_current_tasks_in_list takes in the list_of_rows and prints each of the rows.

The next function input_new_task_and_priority requests the user to input a task and assigns it to the variable task. It then requests the user to input the task's priority and assigns it to the variable priority. The function returns both variables.

Input_task_to_remove prompts the user to input a task and assigns it to a variable named task. The variable, task is returned.

Main

The main body of the script is where each of the functions within the classes are called. It controls the flow of the script. (**Figure 4**)

```
# 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 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

    # 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

    elif choice_str == '2':  # 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 == '3':  # Save Data to File
        table_lst = Processor .write_data_to_file(file_name=file_name_str, list_of_rows=table_lst)
        print("Data Saved!")
        continue  # to show the menu

    elif choice_str == '4':  # Exit Program
        print("Goodbye!")
        break  # by exiting loop
```

Figure 4 – Main body of the script

The script loads the data from the ToDoFile.txt calling read_data_from_file from the Processor class. From the IO class the current tasks, the menu of options, and a prompt asking the user what option they would like is displayed. An if/elif is entered and based on the user's input that corresponding function is run.

Testing the Script

I tested the program in both PyCharm and in the command prompt and it successfully completed using both (**Figure 5**).

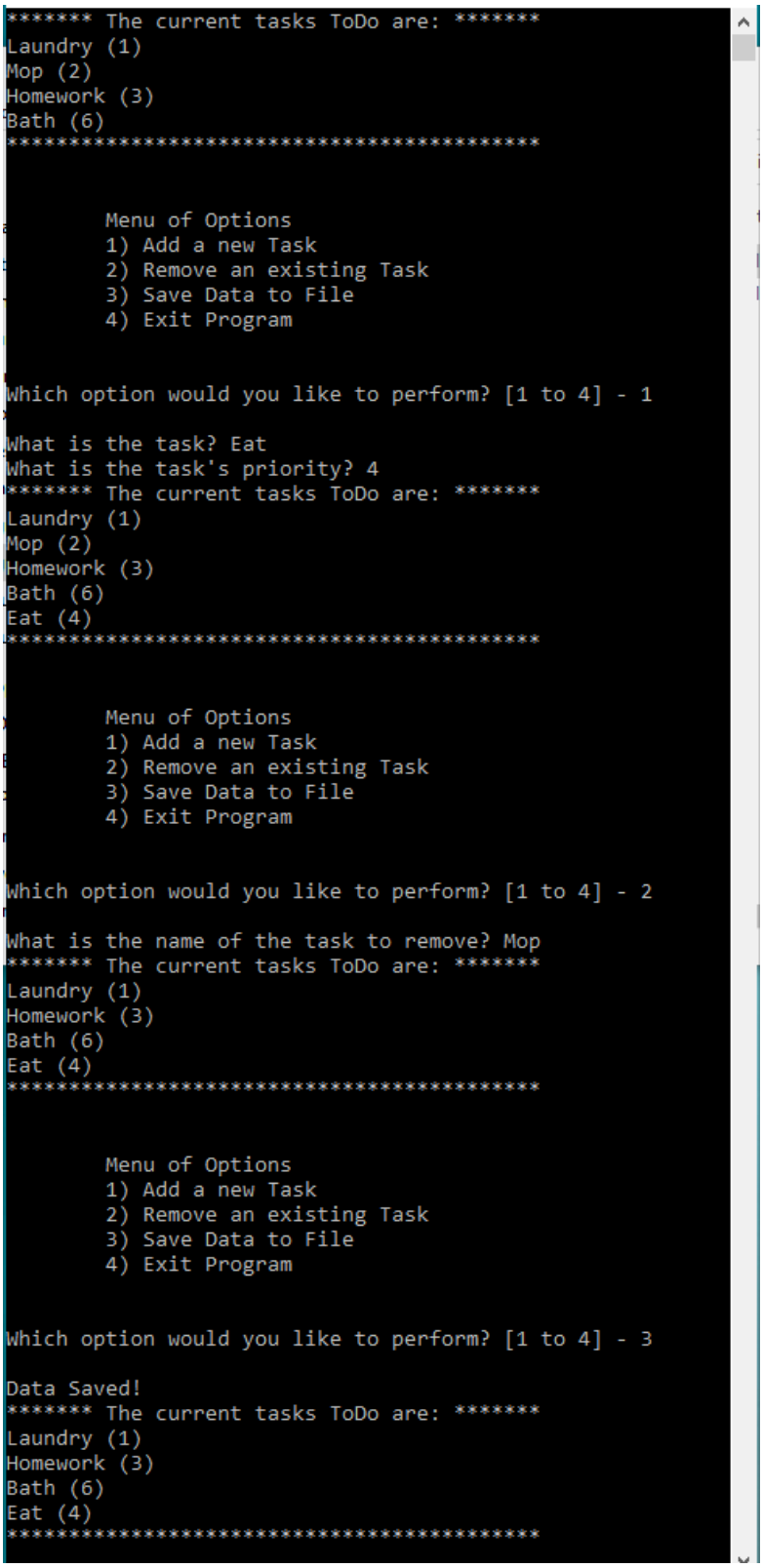

Figure 5. Testing the program

The script successfully ran and updated the text file ToDoList.txt (**Figure 6**).

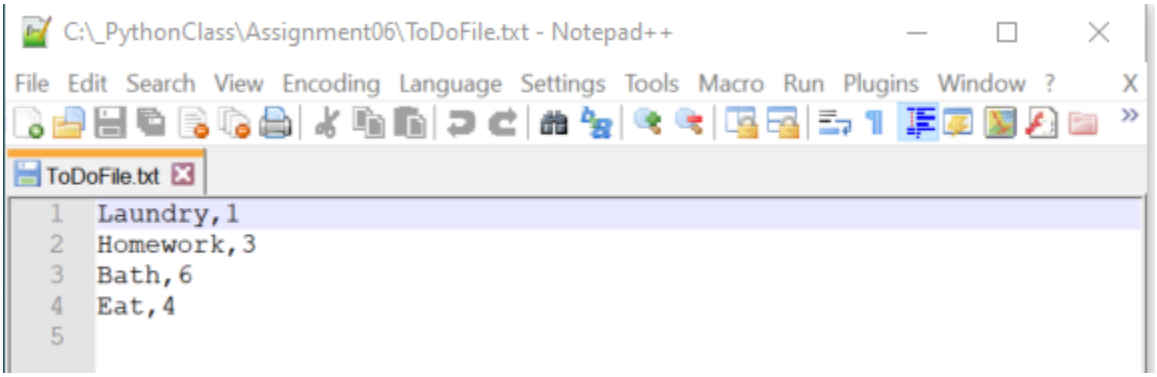

Figure 6. Updated ToDoList.txt

Summary

For Foundations of Programming: Python sixth assignment I worked on a python script that was started previously to add in some functionality. The script reads from a file to memory and then gives the user the option to display the information, add a task, remove a task, write to the file, and end the program.