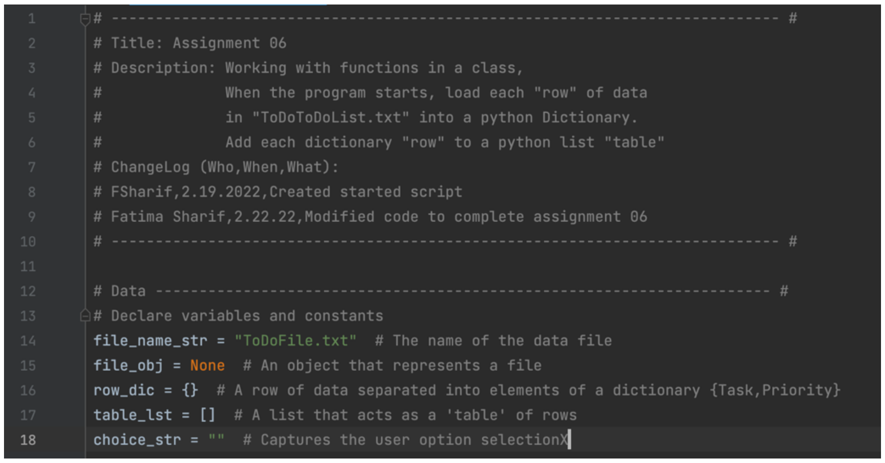
Fatima Sharif
February 22, 2022
Foundations of Programming: Python
Assignment06

Creating Functions

Introduction:
In this paper I will explain to you the steps I took to modify a script that manages a to do list while using a starter templet provided by Professor Randal. The provided starter templet contains code that loads data from a file into a Python list of dictionary objects while using only a few functions. I will modify this programming script by adding more functions to group related code. Lest get started!

Starter Script:
To start, I opened PyCharm and created a new project titled, “Assignment06.” I then created a starter file in my project titled, “Assignment06.py.” Using the starter script provided by my instructor, I edited my program header and viewed the list of declared variables and constraints (Shown in the figure below).

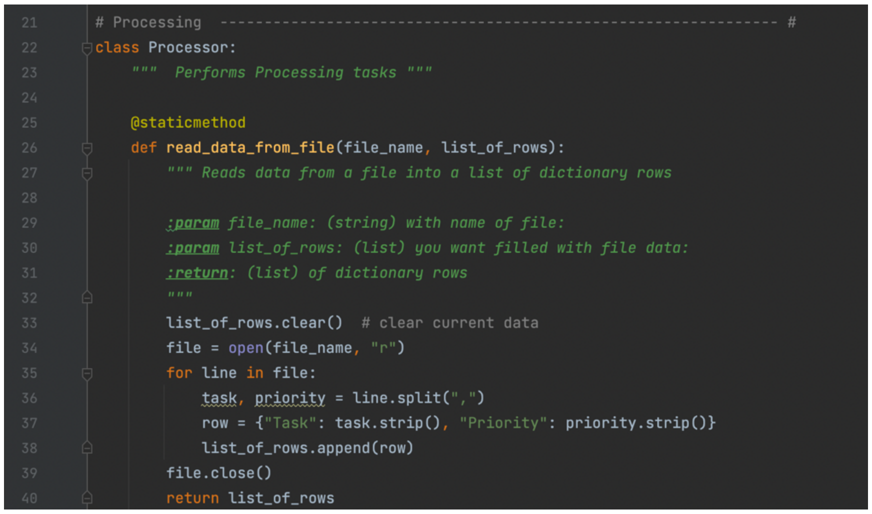


```
# ---------------------------------------------------------------------------- #
# Title: Assignment 06
# Description: Working with functions in a class,
#              When the program starts, load each "row" of data
#              in "ToDoToDoList.txt" into a python Dictionary.
#              Add each dictionary "row" to a python list "table"
# ChangeLog (Who,When,What):
# FSharif,2.19.2022,Created started script
# Fatima Sharif,2.22.22,Modified code to complete assignment 06
# ---------------------------------------------------------------------------- #

# 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 selectionX
```

PROCESSING

Function 1: The first step in writing this programming script was to create functions for the “Processor” class. Using the key word “def” followed by the function name “read_data_from_file,” along parentheses and a colon, I began creating the code block that belongs to this first function. You will see in the image below that the parameters inside the parentheses of the function “read_data_from_file,” are file_name and list_or_rows. These parameters act as variables that stores information. The code block followed by this function will pull data from our “ToDoList” text file and display a list to the user.

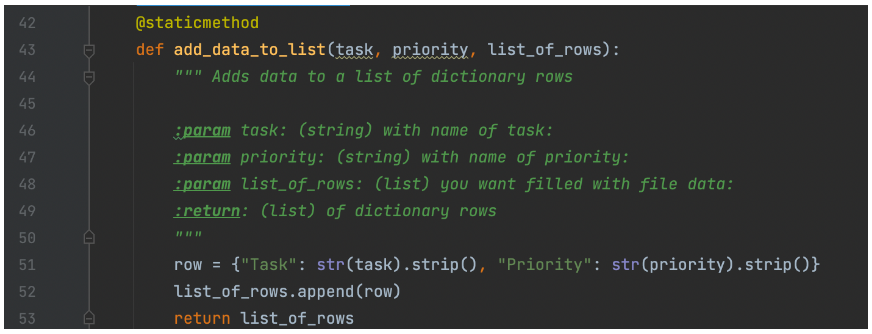


```python
# Processing  --------------------------------------------------------------- #
class Processor:
    """  Performs Processing tasks """

    @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
```

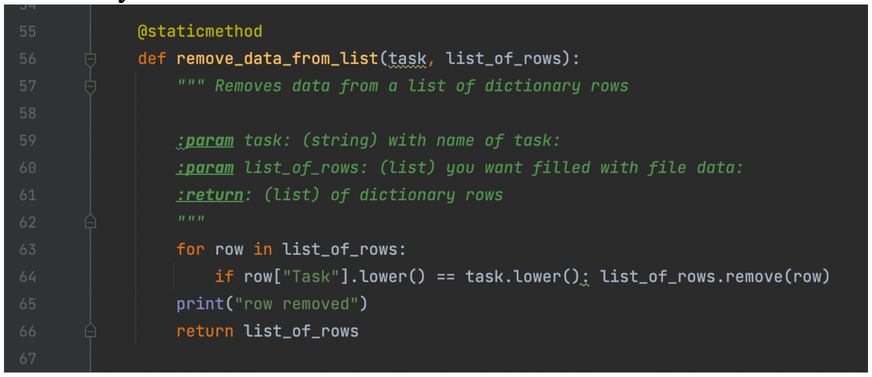
Function 2: Function two, “add_data_to_list,” will be used to add data to a list of dictionary rows.



```python
    @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
```

Function 3: This Function, “remove_data_from_list,” will be used to remove data from a list of dictionary rows.

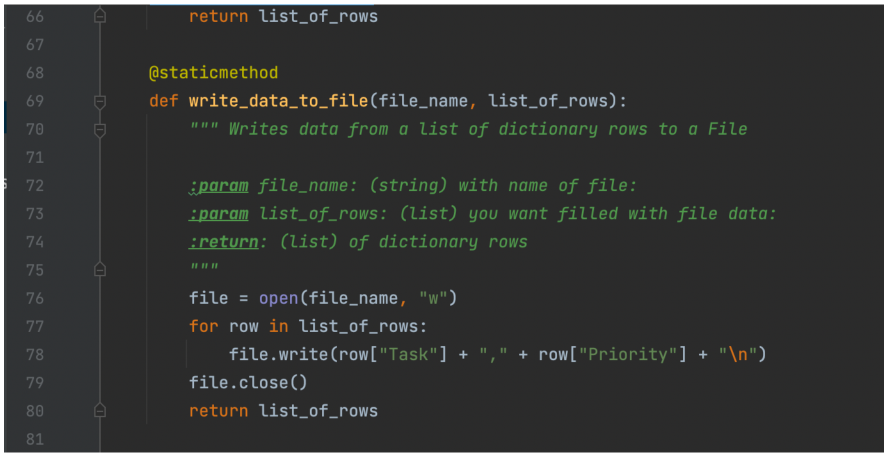


```python
    @staticmethod
    def remove_data_from_list(task, list_of_rows):
        """ Removes data from a list of dictionary rows

        :param task: (string) with name of task:
        :param list_of_rows: (list) you want filled with file data:
        :return: (list) of dictionary rows
        """
        for row in list_of_rows:
            if row["Task"].lower() == task.lower(): list_of_rows.remove(row)
        print("row removed")
        return list_of_rows

```

Function 4: This last function, “write_data_to_file,” will be used to write data from a list of dictionary rows to a file.



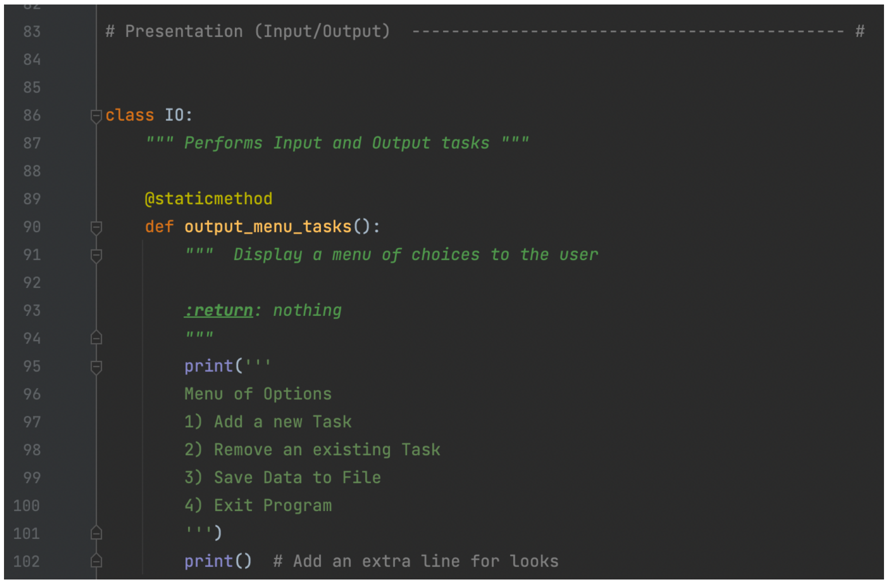
```python
        return list_of_rows

    @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
```

Presentation (Input/output)

Function 5: The second step in writing this programming script was to create functions for the "I/O" class. Function, "output_menu_task," will be used to display a menu of options for the user to choose from in order to interact with the program.



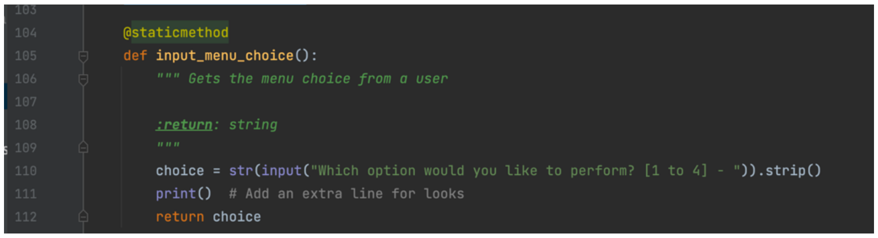
```python
# Presentation (Input/Output)  -------------------------------------------- #


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

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

        :return: nothing
        """
        print('''
        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
```

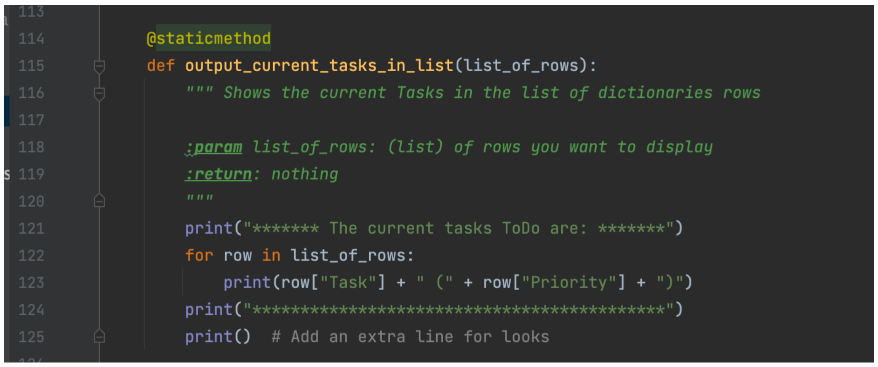
Function 6: This next function, "input_menu_choice," will work to get a menu choice from a user.



```python
    @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
```

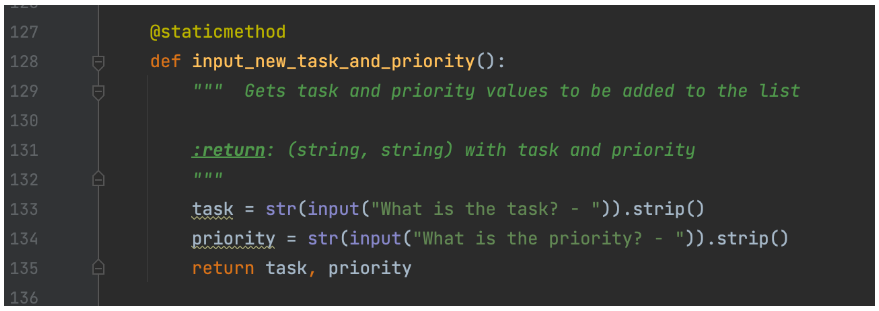
Function 7: Function, "output_current_task_in_list," will be used to show the current task in the list of dictionaries rows.



```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
```

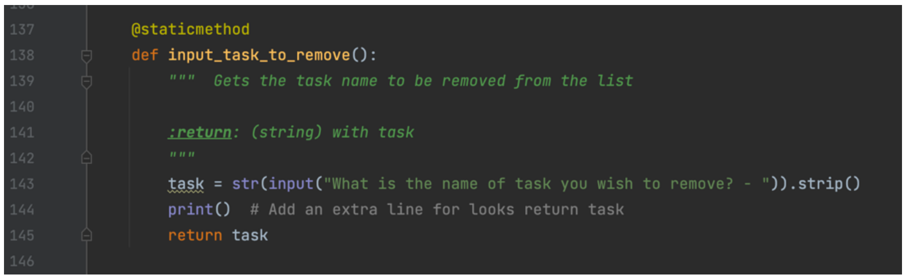
Function 8: This Function here named, "input_new_task_and_priority," will be used to get task and priority values to be added to the to do list.



```python
    @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 priority? - ")).strip()
        return task, priority
```

Function 9: This last function in this class, named "input_task_to_remove," will be used get the task name to be removed from the to do list.

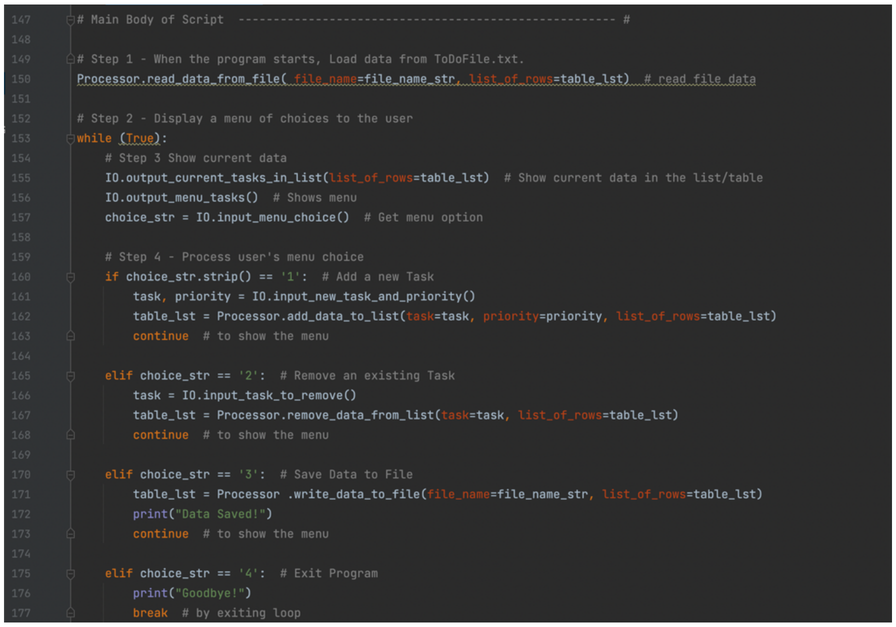


```python
    @staticmethod
    def input_task_to_remove():
        """  Gets the task name to be removed from the list

        :return: (string) with task
        """
        task = str(input("What is the name of task you wish to remove? - ")).strip()
        print()  # Add an extra line for looks return task
        return task
```

Main Body of Script

Now that all functions are defined, I will call each function in the main body script when needed (As shown in the image below).



```python
# 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
```

Summary:

In this paper I will explain to you the steps I took to modify a script that manages a to do list while using function instead of writing the same code repeatedly. I used functions to group related code and perform the task in one place.