Name: Wei Wang
Date: 8/16/2022
Course: IT FDN 110 B Su 22: Foundations Of Programming: Python

Assignment06: Functions and Classes

Introduction

In this module will introduce how to create scripts using Functions. How to create simple Classes, use the PyCharm debugger, and how to post work to a GitHub repository that includes a GitHub web page.

Create a Python Script(in a Mac book)

Assignment requirements:

Modify a starting template and use for the program called "Assigment06_Starter.py". Currently the code loads data from a file into a Python List of Dictionary objects. However, the code only uses a few functions, so I need to add more functions to organize the code.

Step1: Create a new sub-folder called Assignment06 inside the _PythonClass folder

Step2: Create a new project in PyCharm that uses the _PythonClass\Assignment06 folder as its location

Step3: Add the starter file, "Assigment06_Starter.py," to the project.

Step4: Add code to your script that will perform that assignment's task. Update the change log in the script's header.

Part1:
```
# Declare variables and constants
strFileName = "ToDoFile.txt"  # The name of the data file
objFile = None   # An object that represents a file
dicRow = {}  # A row of data separated into elements of a
dictionary {Task,Priority}
 lstTable = []  # A list that acts as a 'table' of rows
 strChoice = ""  # Captures the user option selection
 strTask = ""  # Captures the user task data
```

strPriority = "" # Captures the user priority data
strStatus = "" # Captures the status of a processing functions

Part2: # Processing data—Class Processor: Performs Processing tasks

2-1 Reads data from a file into a list of dictionary rows

Define "read_data_from_file" with parameter of "file_name" and "list_of_rows" ; Return list of dictionary rows

2-2 Writes data to a file from a list of dictionary rows

Define "write_data_to_file" with parameter of "file_name" and " list_of_rows"; Return "status of success"

2-3 Define "add_data_to_file" with parameter of "list_of_rows", "task" and "priority";

2-4 Remove a row of data from a list of dictionary rows

Define "remove_data_from_list" with parameter of "task" and "list_of_rows"; Return "status of success"

code:

```
# 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:
            data = line.split(",")
```

```
            row = {"Task": data[0].strip(), "Priority": data[1].strip()}
            list_of_rows.append(row)
        file.close()
        return list_of_rows, 'Success'

    @staticmethod
    def write_data_to_file(file_name, list_of_rows):
        # TODO: Add Code Here!
        """ Write data to a file from 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: (bool) with status of success
        """

        success_status = False
        file = open(file_name, "w")
        for row in lstTable:  # write each row of data to the file
            file.write(row["Task"] + "," + row["Priority"] + "\n")
            file.close()
            success_status = True
        return success_status


    @staticmethod
    def add_data_to_list(list_of_rows, task, priority):
        # TODO: Add Code Here!
        """ Adds data to a list of dictionary rows

        :param list_of_rows: (string) with name of list you're adding data to
        :param task: (string) with name of task
        :param priority: (string) with name of priority
        """
        row = {"Task": str(task).strip(), "Priority": str(priority).strip()}

        list_of_rows.append(row)

    @staticmethod
    def remove_data_from_list(strKeyToRemove, list_of_rows):
    # TODO: Add Code Here!
    """ Remove a row of data from a list of dictionary rows
```

```
    :param strKeyToRemove: (string) with name of task in the dictionary's 'Task' Key:
    :param list_of_rows: (list) of dictionary data to remove a row from:
    :return: (bool) with status of success
    """
    success_status = False  # Create a boolean Flag for loop
    row_number = 0  # Create a counter to identify the current dictionary row in the loop
    # Search though the table or rows for a 'Task' key match

    for row in lstTable:
        task, priority = dict(row).values()
        if task == strKeyToRemove:
            del lstTable[row_number]
            success_status = True
        row_number += 1
    return list_of_rows, success_status
```

Part3: # Presentation (Input/Output)—class IO:Performs Input and Output tasks

3-1 Define "print_menu_tasks" function

3-2 Define "input_menu_choice" function

3-3 Define "print_current_tasks_in _list" function

3-4 Define "input_yes_no_choice" function

3-5 Define "input_press_to_continue" function

3-6 Define "input_new_task_and_priority" function

3-7 Define "input_task_to_remove" function

code:

```
# Presentation (Input/Output)  ---------------------------------------- #
class IO:
    """ Performs Input and Output tasks """

    @staticmethod
    def print_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) Reload Data from File
        5) 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 5] -
")).strip()
        print()  # Add an extra line for looks
        return choice

    @staticmethod
    def print_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_yes_no_choice(message):
        """ Gets a yes or no choice from the user

        :return: string
```

```
        """
        return str(input(message)).strip().lower()

    @staticmethod
    def input_press_to_continue(optional_message=''):
        """ Pause program and show a message before continuing

        :param optional_message:  An optional message you want to display
        :return: nothing
        """
        print(optional_message)
        input('Press the [Enter] key to continue.')

    @staticmethod
    def input_new_task_and_priority():
        task = str(input("what is the task?-")).strip()
        priority = str(input("what is the priority?[high/low]-")).strip()
        print()  # add an extra line for looks
        return task, priority

    @staticmethod
    def input_task_to_remove():
        task = str(input("what is the task?-")).strip()
        print()  # add an extra line for looks
        return task
```

Part 4: Main body of the script:

```
# Step 1 - When the program starts, Load data from ToDoFile.txt.
Processor.read_data_from_file(strFileName, lstTable)  # read file data

# Step 2 - Display a menu of choices to the user
while (True):
    # Step 3 Show current data
    IO.print_current_Tasks_in_list(lstTable)  # Show current data in the list/table
    IO.print_menu_Tasks()  # Shows menu
    strChoice = IO.input_menu_choice()  # Get menu option

    # Step 4 - Process user's menu choice
    if strChoice.strip() == '1':  # Add a new Task
        # TODO: Add Code Here
        task, priority = IO.input_new_task_and_priority()
```

```
        Processor.add_data_to_list(lstTable, task, priority)
        IO.input_press_to_continue(strStatus)
        continue  # to show the menu

    elif strChoice == '2':  # Remove an existing Task
        # TODO: Add Code Here
        task = IO.input_task_to_remove()
        Processor.remove_data_from_list(task, lstTable)
        IO.input_press_to_continue(strStatus)
        continue  # to show the menu

    elif strChoice == '3':  # Save Data to File
        strChoice = IO.input_yes_no_choice("Save this data to file? (y/n) - ")
        if strChoice.lower() == "y":
            # TODO: Add Code Here!
            Processor.write_data_to_file(strFileName, lstTable)
            IO.input_press_to_continue(strStatus)
        else:
            IO.input_press_to_continue("Save Cancelled!")
        continue  # to show the menu

    elif strChoice == '4':  # Reload Data from File
        print("Warning: Unsaved Data Will Be Lost!")
        strChoice = IO.input_yes_no_choice("Are you sure you want to reload data
from file? (y/n) -  ")
        if strChoice.lower() == 'y':
            # TODO: Add Code Here!
            Processor.read_data_from_file(strFileName, lstTable)
            IO.input_press_to_continue(strStatus)
        else:
            IO.input_press_to_continue("File Reload  Cancelled!")
        continue  # to show the menu

    elif strChoice == '5':  # Exit Program
        print("Goodbye!")
        break  # and Exit
```

The main body of the script are showed as followed (figure and figure2)

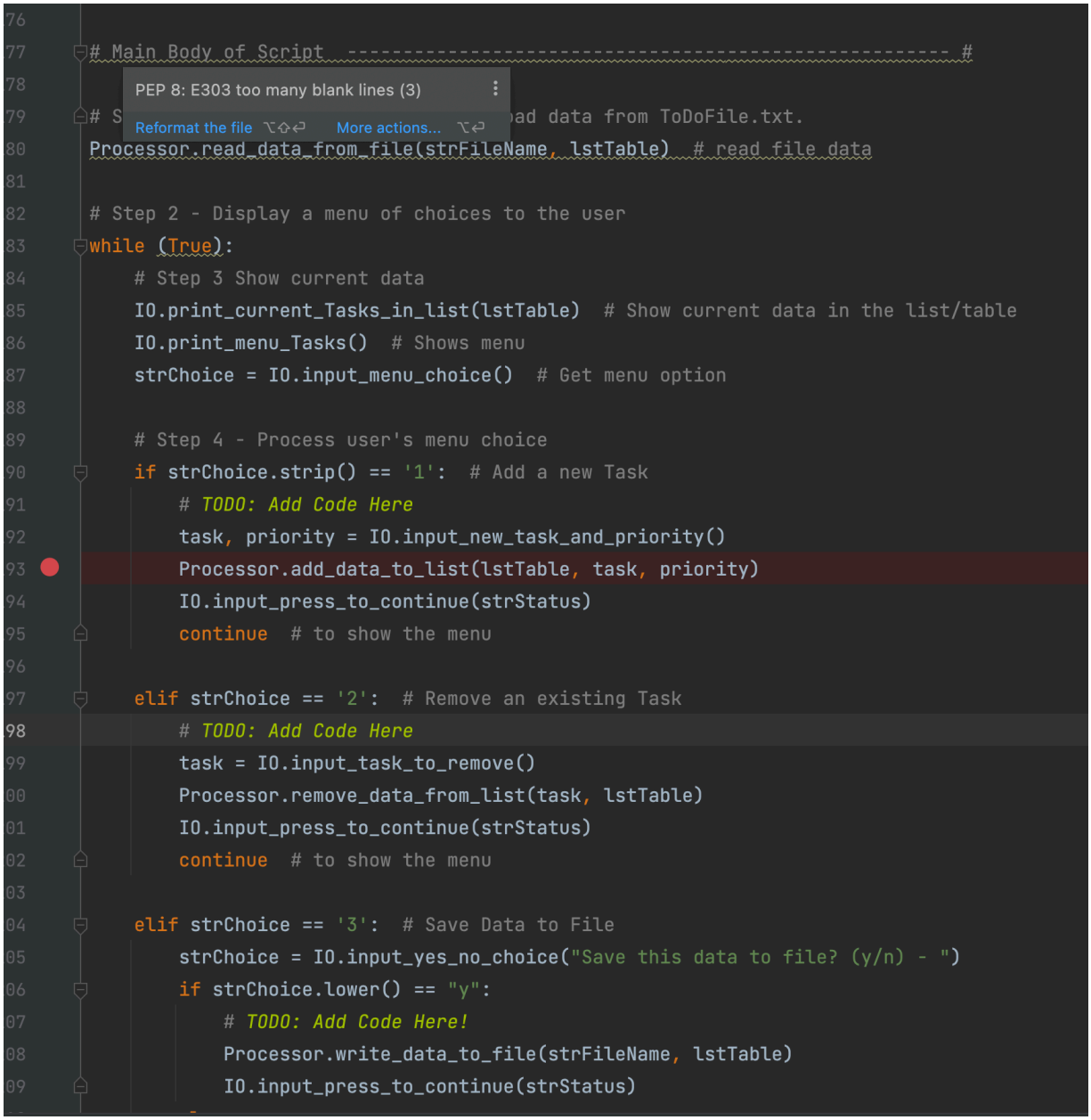

Figure 1: the main body of the script-part 1

Below is the main body of the script part 2

```
elif strChoice == '3':  # Save Data to File
    strChoice = IO.input_yes_no_choice("Save this data to file? (y/n) - ")
    if strChoice.lower() == "y":
        # TODO: Add Code Here!
        Processor.write_data_to_file(strFileName, lstTable)
        IO.input_press_to_continue(strStatus)
    else:
        IO.input_press_to_continue("Save Cancelled!")
    continue  # to show the menu

elif strChoice == '4':  # Reload Data from File
    print("Warning: Unsaved Data Will Be Lost!")
    strChoice = IO.input_yes_no_choice("Are you sure you want to reload data from file? (y/n) -  ")
    if strChoice.lower() == 'y':
        # TODO: Add Code Here!
        Processor.read_data_from_file(strFileName, lstTable)
        IO.input_press_to_continue(strStatus)
    else:
        IO.input_press_to_continue("File Reload  Cancelled!")
    continue  # to show the menu

elif strChoice == '5':  # Exit Program
    print("Goodbye!")
    break  # and Exit
```

Figure 2: the main body of the script-part 2

Step6: Run the code
Run the script both in PyCharm and an OS command/shell window and capture images of it working on my computer

These are the pictures of the screen shot of the script running in PyCharm(Figure 3-Figure 6)

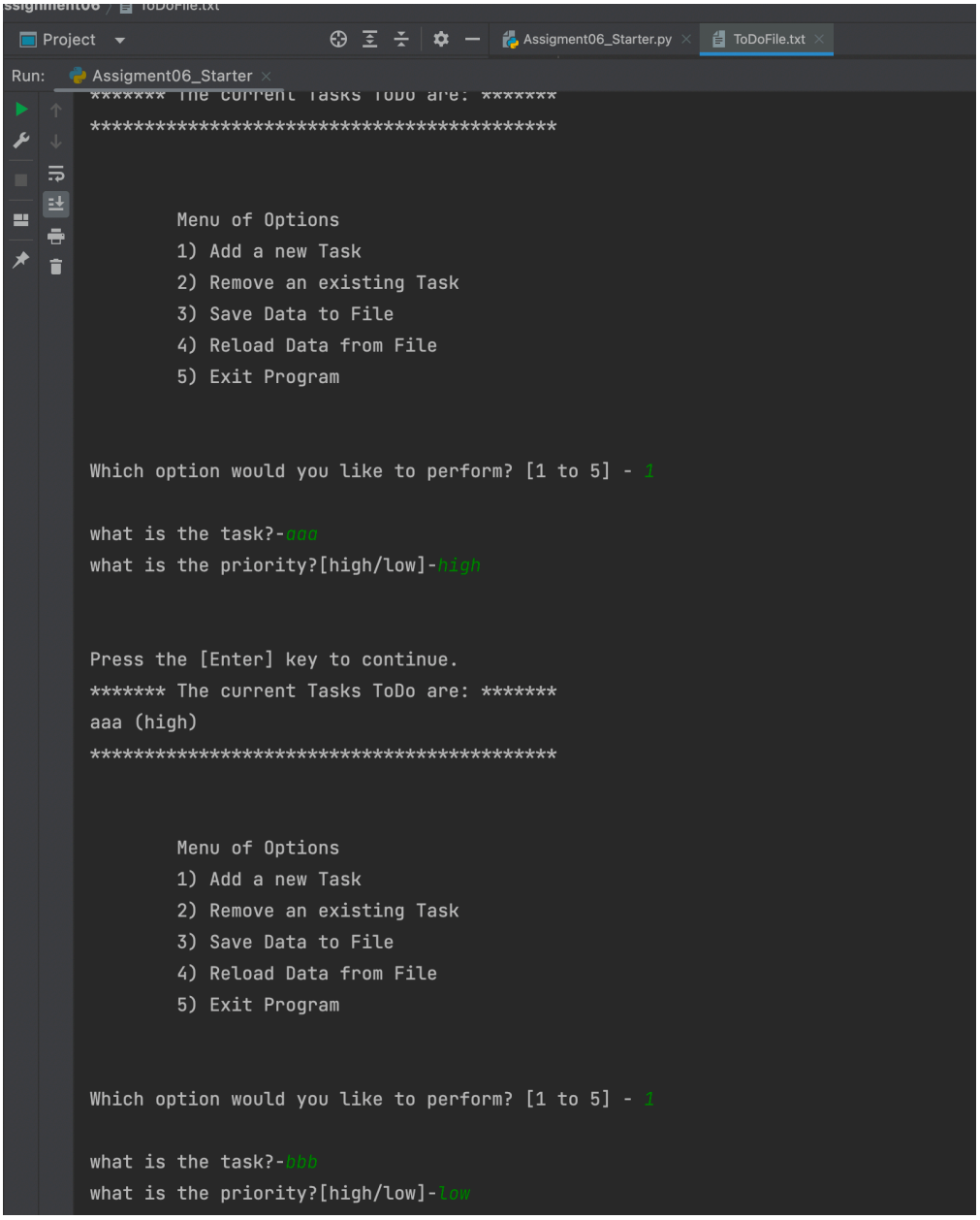

Figure 3: screenshot of the script running in PyCharm part 1

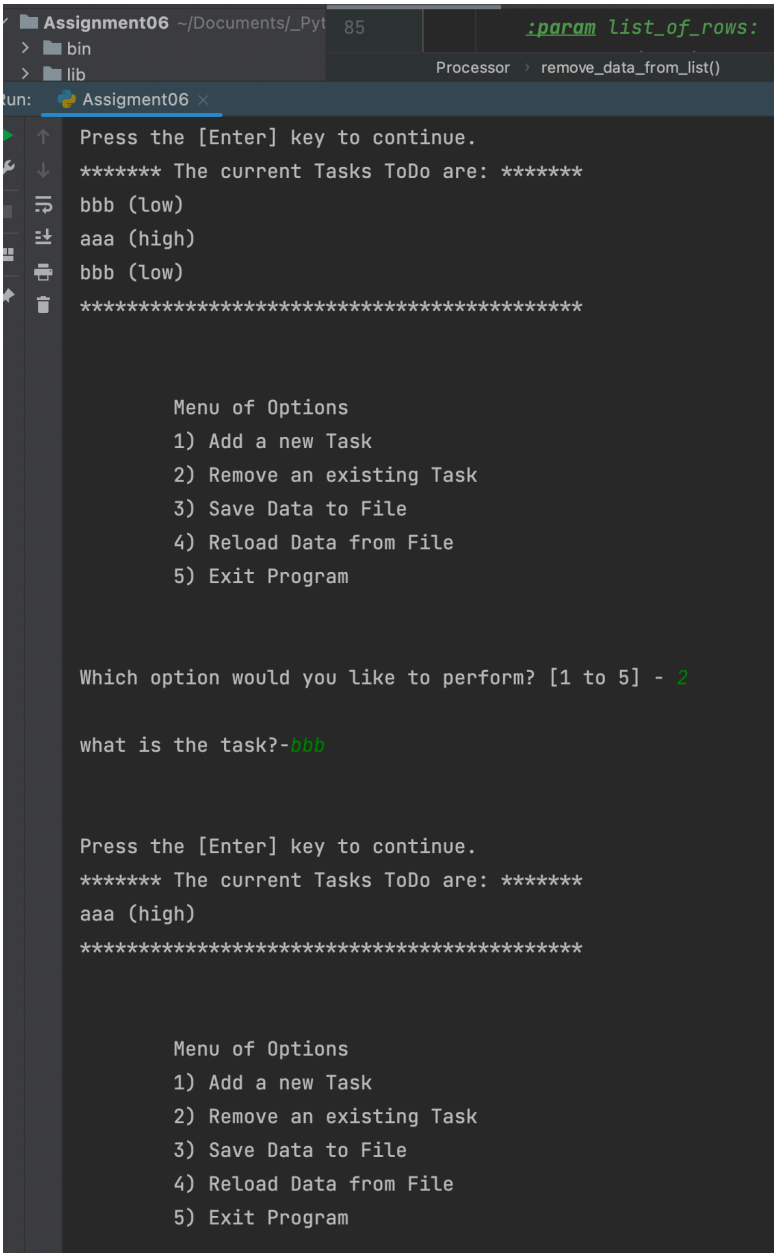

Figure 4: screenshot of the script running in PyCharm part 2

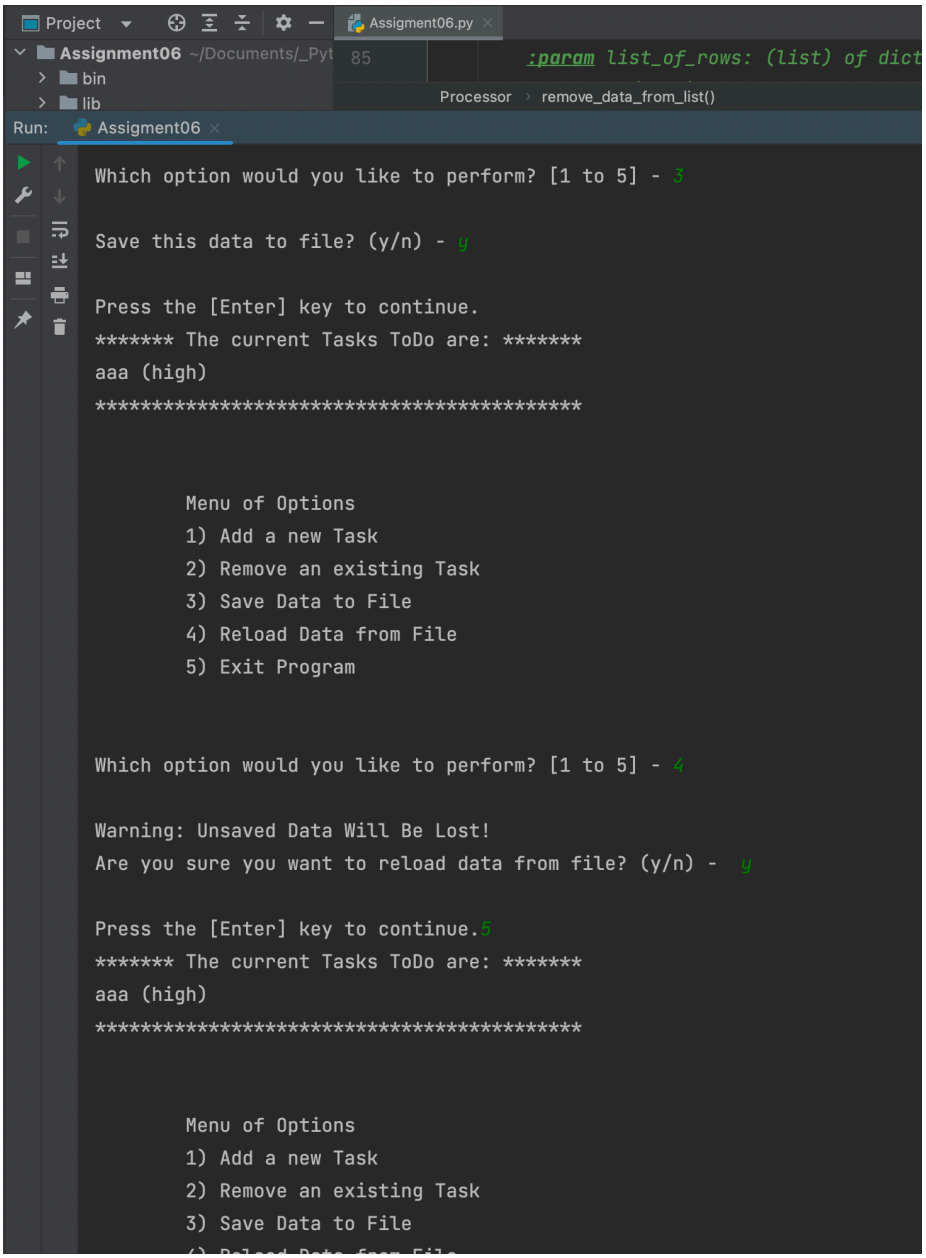

Figure 5: screenshot of the script running in PyCharm part 3

```
        Menu of Options
        1) Add a new Task
        2) Remove an existing Task
        3) Save Data to File
        4) Reload Data from File
        5) Exit Program


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

Goodbye!

Process finished with exit code 0
```

Figure 6: ***screenshot of the script running in PyCharm part 4***

Step 7 Verify that it Worked

Locate the text file and open it in a text editor (Figure 8).

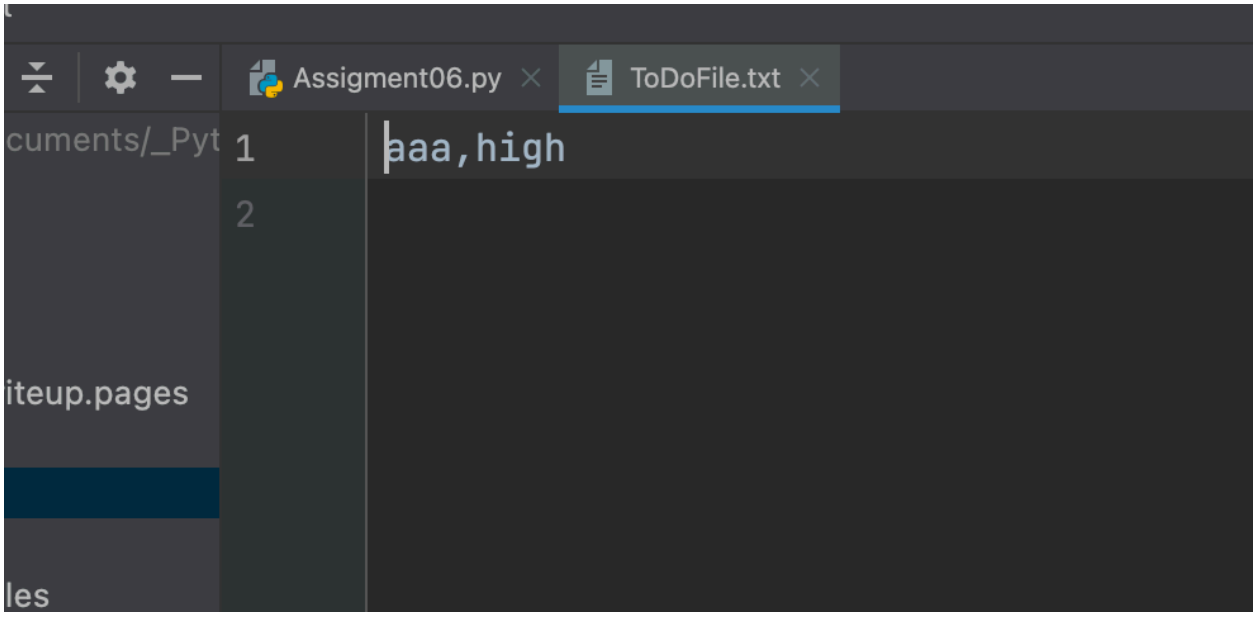

Figure 8: Verifying that the file has data

summary

In this module, I learned how to use functions