Jeff Nelson

May 25, 2020

IT FDN 100 A

Assignment #6

https://github.com/jnelson22/IntroToProg-Python-Mod06

Functions & Classes

Intro

In this paper, I will go over a few interesting and important items to remember that I learned from this week's lecture. Lastly, I will show a program that allows the user to create a To-do list using functions.

Week 6 Learnings

This week I learned about functions, classes, basic debugging in PyCharm, and the GitHub webpage. The main focus of this week was functions. Functions allow for easy organization and reuse of code. In python they have to be written above where they are called. Parameters are used to pass in values for the function to use and process. There is no limit to the number of parameters you can use. Once the processing is complete inside a function they can return values that can be used in other places in the program. One interesting use of arguments is naming them. Figure 1 shows the use of a named argument that links the variable name to the argument name. This allows them to placed in any order.

```
#     processing code     #
def AddValues(value1, value2):
    fltAnswer = value1 + value2
    return value1, value2, fltAnswer

# -- presentation (I/0) code -- #
fltV1 = float(input("Enter value 1: "))
fltV2 = float(input("Enter value 2: "))
fltR1, fltR2, fltR3 = AddValues(value1 = fltV1, value2 = fltV2)
```

Figure 1: Screenshot of named arguments

Overloaded functions

Classes are a way of grouping functions, variables, and constants. In this assignment 6 I used the input/output (IO) and processor classes.

PyCharm comes with some powerful debugging features. These features allow for a much easier way to find and fix bugs in your code.

Commonplace to store and display your code is on GitHub. I went through the process of creating a basic web page on the site.

Creating a To-do List Python Program using Functions

This program's problem statement is the same as last week but now everything has been created using functions. This allows for better organization and portability of the code. This is separated into two classes an input/output (IO) and processing.

First, the program was given with a basic outline and structure similar to last week's assignment. See Figure 2 for an example of the starter code.

```
# 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
        IO.input_press_to_continue(strStatus)
        continue  # to show the menu

    elif strChoice == '2':  # Remove an existing Task
        # TODO: Add Code Here
        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!
            IO.input_press_to_continue(strStatus)
        else:
            IO.input_press_to_continue("Save Cancelled!")
        continue  # to show the menu
```

Figure 2: Starter code

The first section of code that needed to be added was the add a new task. Once the user selected "1" from the menu they need to be prompted and those inputs need to add to memory. In Figure 3 you see the code that was added to call the functions.

```
# Step 4 - Process user's menu choice
if strChoice.strip() == '1':  # Add a new Task
    # TODO: Add Code Here
    # IO.input_press_to_continue(strStatus)
    strTask, strPriority = IO.input_new_task_and_priority()
    Processor.add_data_to_list(strTask, strPriority, list_of_rows)
    continue  # to show the menu
```

Figure 3: Add new task function calling code

The first function calls the IO to gather the user's input data. See Figure 4 for the detailed code. I copied this from my assignment 5 code and modified it to work within a function.

```
def input_new_task_and_priority():
    """ Gathers the users new task and priority inputs

    :return: string (Tasks)
    :return: string (Priority)
    """
    pass  # TODO: Add Code Here!
    strTask = input("What is the task: ").strip()
    strPriority = input("What is the priority: ").strip()
    return strTask, strPriority  # 'Success'
```

Figure 4: Function to ask the user for inputs

Next, the process function, in Figure 5, takes the user input and appends them to the dictionary list in memory. This code is the same as assignment 5 but modified to work in a function.

```
def add_data_to_list(task, priority, list_of_rows):
    """ Add user inputs into a list of dictionary rows

    :param task: (string) user input:
    :param priority: (string) user input:
    :return: (list) of dictionary rows
    """
    lstTable.append({"Task": task.title(), "Priority": priority.title()})
    return list_of_rows
```

Figure 5: Function to write the user inputs to a dictionary of rows

Below, Figure 6 is the code running in PyCharm.

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

What is the task: clean dishes
What is the priority: low
******* The current Tasks ToDo are: *******
Finish Homework (Hi)
Clean Dishes (Low)
*****************************************
```

Figure 6: Example of working code

The second section of code that needed to added was the remove a task. Once the user inputs “2” from the menu they need to be prompted to enter the task name and then it is removed from memory. Below in Figure 7 is the code that was added to call those functions.

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

Figure 7: Add remove task function calling code

Figure 8 shows the code that presents the user with a message to input which task they would like to remove from memory.

```
def input_task_to_remove():
    """ Gathers the users task they would like to remove

    :return: string (Tasks)
    """
    pass  # TODO: Add Code Here!
    task = input("Task to remove: ")
    return task
```

Figure 8: Function to ask the user which task to remove

Once the user has entered which task they want to remove. That input gets passes to the processor function, Figure 9, that then looks through the dictionary list and removes it if it finds it. If it’s not in the list then the user is presented with a message “Task not found”

```
def remove_data_from_list(task, list_of_rows):
    """ Remove user inputs from a list of dictionary rows

    :param task: (string) user input:
    :return: (list) of dictionary rows
    """
    removeItem_flag = 0  # set controller to 0
    for row in lstTable:
        if row['Task'].lower() == task.lower():
            lstTable.remove(row)
            removeItem_flag = 1
            break
    if removeItem_flag == 0:
        print("Task not found in list")
    return list_of_rows  # 'Success'
```

Figure 9: Function that takes the user input and removes it from the dictionary rows

Below, Figure 10, is an example of removing a task working in PyCharm.

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

******* The current Tasks ToDo are: *******
Finish Homework (Hi)
Clean Dishes (Low)
*******************************************

Task to remove: clean dishes
******* The current Tasks ToDo are: *******
Finish Homework (Hi)
*******************************************
```

Figure 10: Example of removing task working code

The third section of code that needed to be added is to write the data in memory into a text file. Once the user input “3” they will be prompted if they want to save to a file. If they input “y” the data in memory will be written to a file. If they input “n” saving will be canceled. Figure 11, shows the code I added to call the functions need to accomplish this.

```
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!
        # IO.input_press_to_continue(strStatus)
        Processor.write_data_to_file(strFileName, lstTable)
    else:
        IO.input_press_to_continue("Save Cancelled!")
    continue  # to show the menu
```

Figure 11: Save data to a file function call

Figure 12, shows the code used from assignment 5 to write the data in the list to a text file. This function takes the file name and current memory of data and writes them to a text file.

```
def write_data_to_file(file_name, list_of_rows):
    """ Write data from a a list of dictionary rows into 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
    """
    objFile = open(file_name, "w")
    for row in lstTable:
        objFile.write(str(row["Task"]) + ", " + str(row["Priority"]) + "\n")
    objFile.close()
    print("Tasks saved to ToDoList.txt")
    return list_of_rows  # 'Success'
```

Figure 12: Function to write everything in memory to a text file

Below, Figure 13 and Figure 14, show the code running in PyCharm and the corresponding text file.

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

Save this data to file? (y/n) - y
Tasks saved to ToDoList.txt
******* The current Tasks ToDo are: *******
Finish Homework (Hi)
Clean Dishes (Low)
Water Plants (Hi)
*******************************************
```

Figure 13: Example of adding data to a file working

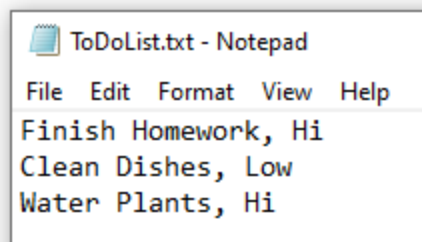

Figure 14: Example of text file after save

The final section that needed code added to it is the reload data from the file. Once the user inputs "4" they will be asked if they want to reload data from the file overwriting what is currently in memory. Figure 15, shows the functions calls I used to accomplish this. Using functions allowed for just calling already created functions that read the data from a file.

```
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!
        # IO.input_press_to_continue(strStatus)
        Processor.read_data_from_file(strFileName, lstTable)
    else:
        IO.input_press_to_continue("File Reload  Cancelled!")
    continue  # to show the menu
```

Figure 15: Function call to reload data from a file into memory

Below, Figure 16, shows the before and after reloading from a file in PyCharm.

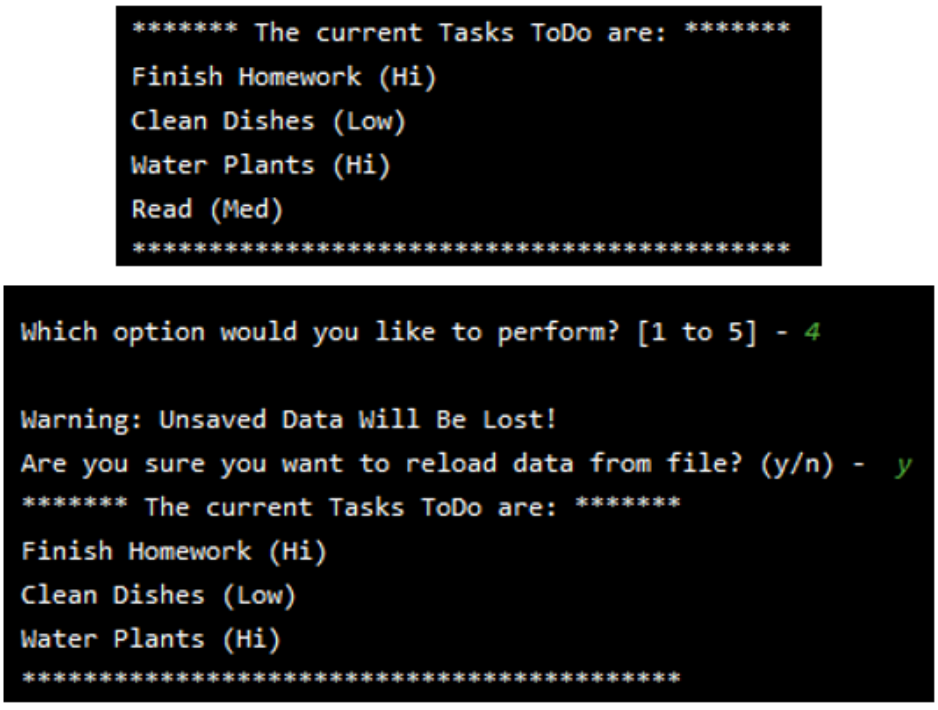

Figure 16: Example for reloading from a text file

Lastly, here is the code running in the command line.

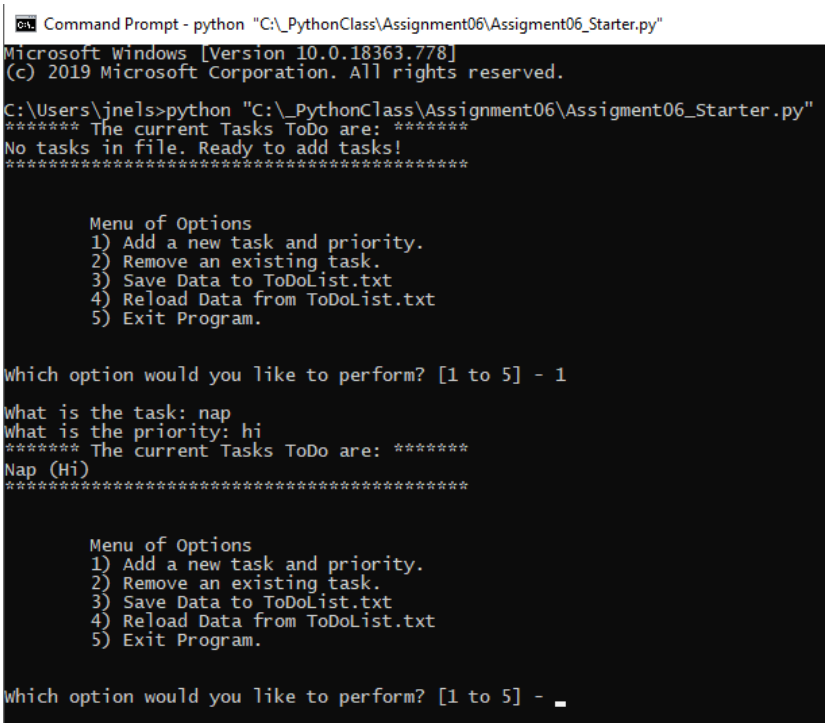

Figure 17: Example of code running in windows command prompt

Summary

In summary, I learned about functions, classes, basic debugging in PyCharm, and GitHub webpage. I went through the process of how I added to the to-do list function code to allow the user to add, remove tasks, save them to a file, and reload from the file. I found this assignment a little more difficult than the last one. The functions caused me a little difficult trying to understand what was being passed from the calling statements but I was able to work through them using the debugger in PyCharm. Overall, I'm happy with this assignment and how the use of function has expanded my understanding of programming.