

Sarah Anderson

November 22, 2022

ITFnd100

[https://github.com/sarahanderson94/ITFnd100\\_Mod06](https://github.com/sarahanderson94/ITFnd100_Mod06)

## Assignment 06: Functions and Classes

### Introduction

For this week's assignment, I learned how to create functions and classes. The program I created had the same objective as last weeks to do file assignment, but I changed the code to include functions. Functions make it easier to follow the separation of concerns principle. There are many benefits to using functions. One of the biggest advantages is that it allows the programmer to break code up into more manageable chunks. It also helps the programmer type out less code as the function just needs to be defined once and then can be used repeatedly throughout the program. This document describes how functions and classes were used to recreate the To Do List program.

### Functions

In the improved To Do File program, I've replaced much of the main body script with calling functions. This makes the main body script much cleaner to read and allows me to call the functions more than once if I need to, which saves time from writing out code over and over.

To create a function, you need to use the following syntax `"def FunctionName():"`. This tells Python you are creating a new function. The code for the function will be indented on the lines below the line to create the function. Functions can have many parameters, but it is not necessary for them to have any. Figure 1 shows a function used in the To Do program that has two parameters. When the function in Figure 1 is called, a parameter containing a string and a list are passed in. Figure 2 shows what the function looks like when it is called. The purpose of this function is to remove a task from the list of tasks. Figure 3 shows a function that takes in no parameters. The function in Figure 3 is used to input two new string variables

Using Return Values in functions is a helpful way to capture variables in other areas of the program. The functions in Figures 1 and 3 both use return values. Figure 1 shows the function returns a list and Figure 2 shows that in the main body of the script variable, `"table_lst"`, is used to store the return value. Figure 3 shows the Input New Task function and it returns two values, `taskIO` and `priorityIO`. When a function returns multiple values, they must be packed up into a collection. To capture the return values, the collection must be unpacked. Figure 4 shows the unpacking and capturing of the Input New Task function's return values.

```

@staticmethod
def remove_data_from_list(task, list_of_rows): # Sar updated
    """ 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.get("Task").lower() == task.lower():
            print("row removed")
            list_of_rows.remove(row)
            break

    # Placing the else statement outside the For loop prevents
    # the program from printing multiple lines of "row not found"
    else:
        print("row not found")
    return list_of_rows

```

Figure 1: Function with 2 Parameters (Remove Data Function)

```

elif choice_str == '2': # Remove an existing Task
    task = I0.input_task_to_remove()
    table_lst = Processor.remove_data_from_list(task=task, list_of_rows=table_lst)

```

Figure 2: Calling the Remove Data Function

```

@staticmethod
def input_new_task_and_priority(): # Sar Updated
    """ Gets task and priority values to be added to the list

    :return: (string, string) with task and priority
    """

    taskI0 = str(input("Please enter a new task: ")).strip()
    priorityI0 = str(input("Please enter task priority (low, medium, high): ")).strip()
    return taskI0, priorityI0

```

Figure 3: Function with No Parameters (Input New Task Function)

```

if choice_str.strip() == '1': # Add a new Task
    task, priority = I0.input_new_task_and_priority()

```

Figure 4: Calling the Input New Task Function

## Classes

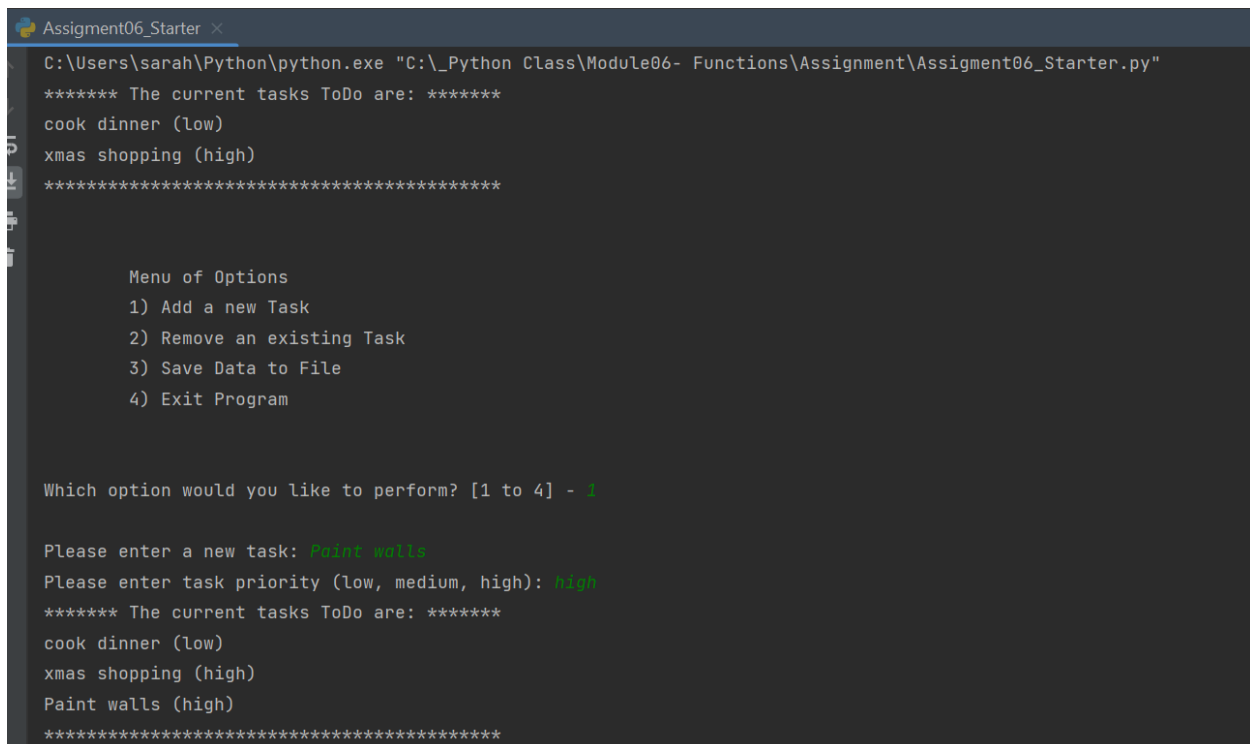
Classes are a way to organize and group functions. For this assignment, two classes were created. One called "Processor" and one called "IO". The Processor class contains all the functions that manipulate the data, such as, reading in the text file, adding new rows to the table, removing rows from the table, and writing the data to a file. The IO class contains all the functions that either take in user input or output data to the user. When calling a function within a class, the syntax used is `ClassName.FunctionName()`. Figure 5 shows an example of an IO and Processor function called with in the main script.

```
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
```

Figure 5: Using a Functions from Class IO and Processor

## Running the To Do List Program (Version 2)

Figures 6 and 7 demonstrate the program running in both PyCharm and the Command Console, respectively.



```
Assignment06_Starter x
C:\Users\sarah\Python\python.exe "C:\_Python Class\Module06- Functions\Assignment\Assignment06_Starter.py"
***** The current tasks ToDo are: *****
cook dinner (low)
xmas shopping (high)
*****

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

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

Please enter a new task: Paint walls
Please enter task priority (low, medium, high): high
***** The current tasks ToDo are: *****
cook dinner (low)
xmas shopping (high)
Paint walls (high)
*****
```

Figure 6: Running the Program in PyCharm

```

Microsoft Windows [Version 10.0.19044.2130]
(c) Microsoft Corporation. All rights reserved.

C:\Users\sarah>Python.exe "C:\_Python Class\Module06- Functions\Assignment\Assignment06_Starter.py"
***** The current tasks ToDo are: *****
cook dinner (low)
xmas shopping (high)
buy groceries (low)
*****

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

Which option would you like to perform? [1 to 4] - 2

Enter a task to remove: cook dinner
row removed
***** The current tasks ToDo are: *****
xmas shopping (high)
buy groceries (low)
*****

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

Which option would you like to perform? [1 to 4] - 3

Data Saved!
***** The current tasks ToDo are: *****
xmas shopping (high)
buy groceries (low)

```

**Figure 7: Running the Program in Command Console**

## Summary

Using functions and classes is super helpful for organizing code and making the scripting process more efficient. The new and improved To Do File program is much easier to read than last week's version.