# Natalie E. Tolliver

5/25/2020

ITNFDN 100A

Assignment 06

<https://tollivne.github.io/ITFnd100-Mod06/>

Using Classes and Functions for Separation of Concerns

## Introduction

The purpose of this assignment was to create code that allows a user to manage a task list. The previous assignment also performed the same function with the same input and output. Although the code in this assignment performs the same function as the previous assignment, such that a user of the code would only notice slight differences in the input and output and the process; the code bears little resemblance to the previous code. It uses classes and functions to organize the code into Data, Input/Output tasks and processing tasks. To accomplish this, I removed sections of code from the main body of the program (controller) to a function using the proper syntax for defining a function, documenting its purpose, its parameters, and its return values, and finally making the proper call to the function in the body of the main program.

Also, as part of Assignment06, I expanded my toolset by using the debugging tool to step through lines of code to assist in resolving issues and I also created a basic web page on the GitHub website.

## What is a function?

A function is a sequence of instructions that perform a task, bundled as a unit (Romano, Fabrizio. Learn Python Programming: The no-nonsense, beginner's guide to programming, data science, and web development with Python 3.7, 2nd Edition (p. 109). Packt Publishing. Kindle Edition.) This set of code can then be used elsewhere in the code when and where it is needed. A function is defined by using the keyword “def” and then the name of the function and a pair of parentheses containing the input parameters and ending with a colon. The def statement is typically followed by a set of comments outlining the purpose of the function (doc string), the input parameters, and the returned data. This is called a function document header or “doc string.” The body of the function is then indented four spaces under this statement. If you want the function to return a value, the “return” statement is used followed by the output variables.

## What are parameters?

Parameters are the names that appear in a function definition (<https://stupidpythonideas.blogspot.com/2013/08/arguments-and-parameters.html>, 5/23/2020, External). Unlike the main body of the programs I have written previously, the parameter names do not have a prefix (str, float, lst, bln, int, etc). The names are lower case and snake casing is used for the variable names. This naming convention is not required but is an industry preference.

## What are arguments?

Arguments are the values passed from the main body of the program to a function when calling it. Parameters define what type of arguments a function can accept.

## What is the difference between parameters and arguments?

The parameters are used in the function. The arguments are used in the body of the code.

## What are returned values?

Returned values are the output from the function. In Python, a Tuple is often used to return multiple values. This is because the individual values can be easily used by “unpacking” the tuple. If a list is used, each argument must be called out separately by using its index.

## What is the difference between a global and a local variable?

A local variable is only used within the function. Its value is not passed outside of the function. A global variable is one that is used both inside and outside of the function.

## How do you use functions to organize your code?

Functions are used to organize the code so that repeatable tasks are done within one set of code and so that the code can be separated into different sections. As an example, code can be separated more easily into input, processing, and presentation by using functions.

## What is the difference between a function and a class?

A class defines a set of functions.

## How do functions help you program using the “Separations of Concerns” pattern?

Separation of Concerns is a method of separating different aspects of code into different modules that deal with a single “concern.” Using functions can help by including processing as one “concern,” separating out “presentation.” This makes the code much more readable.

## How are the debugging tools used in PyCharm?

By setting a “break point” in PyCharm, you can use the debugging tool to step through the code and pause at these points. This helps you to isolate problem areas of code, narrow down the problem, and find and fix the bugs.

## What is a GitHub webpage?

A GitHub webpage is a personalized webpage that the developer can create within the GitHub website. The website uses markdown programming language. My GitHub website address is <https://tollivne.github.io/ITFnd100-Mod06/> (5/23/2020, External).

# Setting up the Code

For this exercise, we were given a set of fully functioning code that performed basically, the same task as the previous exercise. The intent of the code was to manage a task list. The code reads the existing task list from a file into memory, processes the data by adding tasks, or removing tasks, displays the current data, and writes it back to the file. The purpose of modifying the code is to make it more readable and more in conformance to the industry standards for separation of concerns through the use of classes and functions. To begin, I copied the starter code into a new python file called Assignment06, created a new task list in the same folder, edited the header of the code, and searched for the areas showing items that needed to be changed.

The items needing to be changes were as follows:

1. I/O code - Define and call a function that asks the user for the task and priority.
2. Processing code - Define and call a function that adds a task & priority from the dictionary row to the list table.
3. Processing code - Define and call a function that removes a task & priority from the list.
4. I/O code - Create and call a function that writes the code to a file.
5. I/O code - Prompting the user for the task to be removed from the list.
6. Processing code – Removing the item from the list.

In addition, there were other areas where there was duplication of a task. One of these was displaying an optional message to the user and prompting the user to “Press enter to return to the main menu.” Another function that can be used to replace redundant code is getting a yes/no input from the user.

1. I/O code - Creating and calling a function to display an optional message and prompting the user to press enter to return to the main menu.
2. I/O code - Creating and calling a function to get a yes/no answer from the user.

# Prompting the User for the Task and the Priority

For this function to prompt the user for the task and priority, there are no input parameters. The purpose of the function is to return the values for strTask and strPriority. I chose to call the function GetTaskandPriority in the IO Class. I commented out the code in the main body of the function and wrote the code shown in Figure 1.

The first error I encountered came from writing “Staticmethod” with a capital “S.” This was a little tricky/confusing because although it is perfectly acceptable to start the name of the functions with a capital letter, “staticmethod” must be started with a lowercase “s.”

The code in Figure 1 was placed in the I/O class.

@staticmethod  
def GetTaskandPriority ():  
 task = str(input("What is the task? - ")).strip() # Get task from user  
 priority = str(input("What is the priority? [high|low] - ")).strip() # Get priority from user  
 print() # Add an extra line for looks  
 return(task, priority)

Figure 1 - Defining the IO Function to Collect User Input

The code returned two values as a Tuple which was unpacked as shown in Figure 2.

strTask, strPriority = IO.GetTaskandPriority() # unpack tuple

Figure 2 - Unpack Tuple Values Returned from Function

The last step of defining the function was to add the comments known as the function document headers or “Doc String” shown in Figure 3:

*""" This function prompts the user to enter the task & priority****:param*** *none****:return****: Task name & Task priority (High/Low)  
"""*

Figure 3 – Doc String

# Function for Writing Dictionary Rows into the File

The next function to create was for opening the text file, looping through all the rows in the table, writing each dictionary row to the file, and closing the file. The parameters passed to the function are the file name and the values in the list table. The variable names are file\_name, and list\_of\_rows. There are no variables returned from this process because it is writing the data to a file. The function name was given in the program and is WriteListDataToFile. The code is shown in Figure 4.

@staticmethod  
def WriteListDataToFile(file\_name, list\_of\_rows):  
 # TODO: Add code Here - Done  
 *""" This function writes the task list to a text file* ***:param****: File Name, ToDo List* ***:return****: None  
 """* objFile = open(file\_name, "w")  
 for dicRow in list\_of\_rows: # Write each row of data to the file  
 objFile.write(dicRow["Task"] + "," + dicRow["Priority"] + "\n")  
 objFile.close()

Figure 4 - Function to Write Data to File

The call to the function is shown in Figure 5. The key to success in getting this code to work properly on the first pass was a good understanding of the parameters, the arguments, and which values need to be returned to the main program.

FileProcessor.WriteListDataToFile(strFileName, lstTable)

Figure 5 - Call to Function to Write Data to File

The last step was to add the comments (Doc String) to the function as shown in Figure 6.

*""" This function writes the task list to a text file****:param****: File Name, ToDo List****:return****: None  
"""*

Figure 6 - Comments (Doc String) for Function to Write Data to File

# Function for Adding a Task to the ToDo List

The next function was to add a task to the ToDo list. In this function, the arguments are strTask and strPriority. They are added to a dictionary row which is then appended to the lstTable. The updated lstTable is the final data which needs to be returned to the main program. I chose to call my function AddItemtoList and the code is shown in Figure 7.

def AdditemtoList (Task, Priority):  
 *""" This function adds a Task and Priority to the Table  
 param: Task and Priority  
 return: Updated list Table  
 """* dicRow = {"Task": Task, "Priority": Priority} # Create a new dictionary row  
 Table.append(dicRow) # Add the new row to the list/table  
 return (Table)

Figure 7 - Code for Adding an Item to the List

The call to the function is shown in Figure 8 .

lstTable = FileProcessor.AdditemtoList(strTask, strPriority)

Figure 8 - Calling the Function to Add an Item to the List

# Input and Press the Enter Key to Continue

There are several areas in the program where an optional message is printed out and the user is prompted to press the enter key to continue. Although the optional message varies, the user is always prompted to press the enter key to continue. A function can be used to automate the redundant part of this task.

The parameter would be the optional message. Nothing would be returned because it is just printing and then hitting enter to advance the program. The function will be called “input\_press\_to\_continue” and the parameter will be “optional\_message.”

The code for accomplishing this task is shown in Figure 9.

@staticmethod  
def input\_press\_to\_continue(optional\_message):  
 *""" Pause the program and display a message before continuing* ***:param*** *optional\_message: An optional message to display to the user* ***:return****: nothing  
 """* print(optional\_message)  
 input("Press the [Enter] key to continue.")

Figure 9 - Displaying an Optional Message and pressing Enter to Continue

There are three calls to this function as shown in Figure 10.

IO.input\_press\_to\_continue("Data saved to file!")

IO.input\_press\_to\_continue("New data was NOT Saved, but previous data still exists!")

IO.input\_press\_to\_continue("File data was NOT reloaded!")

Figure 10 - Displaying a Message and Pressing Enter to Continue

# Getting a Yes/No Answer from the User

For this function, the parameter would be the question that the code wants to ask the user. The return value would simply be a “y” or a “n.” The code for this function is shown in Figure 11 .

def input\_yes\_no\_choice(message):  
 *""" Get a yes or no choice from the user  
 param: Question to be asked of the user  
 return: y or n answer  
 """* return str(input(message)).strip().lower()

Figure 11 - Function to Get a Yes or No Answer from the User

# Getting the Task to be Removed

To get the task to be removed, the parameter in the function would be none and the returned value would be the string containing the task to be removed. The code is shown in Figure 12.

@staticmethod  
def input\_task\_to\_remove():  
 remove = input("Which TASK would you like removed? - ") # get task user wants deleted  
 return (remove)

Figure 12 - Function to Get Task User Wants to Remove

The call to the function is shown in Figure 13.

strKeyToRemove = IO.input\_task\_to\_remove() # get task user wants deleted

Figure 13 - Calling the Function to Get the Task to be Removed

# Removing the Task

To remove the task, the parameter the function needs are the string of the item to be removed, and the list of tasks. What should be returned is the new revised task list. The code is shown in Figure 14.

@staticmethod  
def remove\_data\_from\_list(task, list\_of\_rows, item\_removed):  
 intRowNumber = 0 # Create a counter to identify the current dictionary row in the loop  
  
 # Step 3.3.b - Search though the table or rows for a match to the user's input  
 while (intRowNumber < len(lstTable)):  
 if (strKeyToRemove == str(list(dict(lstTable[intRowNumber]).values())[0])): # Search current row column 0  
 del lstTable[intRowNumber] # Delete the row if a match is found  
 item\_removed = True # Set the flag so the loop stops  
 intRowNumber += 1 # Increase counter to get next row  
  
 return list\_of\_rows, item\_removed # Make sure to unpack tuple when values are returned!

Figure 14 - Function to Remove a Task

The code for calling the function is shown in Figure 15.

lstTable, blnItemRemoved = Processor.remove\_data\_from\_list(strKeyToRemove, lstTable, blnItemRemoved) # unpack tuple

Figure 15 - Call to function for unpacking tuple

This was the most difficult function to write. Initially, I was not returning the Boolean item removed. Although the code executed properly and removed the item, the printed message would tell the user that the item was not found. The function sets the Boolean value to true and the controller test kept coming up false. The problem was caused by the fact that the Boolean value of true was not being passed on to the controller. Although the value was set to true in the function, since it was NOT being passed on, the controller still had it set as false and would print the corresponding message for a false test. The problem was resolved by adding the Boolean value as a return parameter for the function and then, since there were multiples value being returned, unpacking the tuple in the statement that calls the function.

# Final Touches

Something strange happened when I had almost completed my code. I watched the video from the Tuesday Zoom session and noticed that my starter code was different from the starter code reviewed in class. I then found out that we needed to use a revised version of the starter code. To make mine match I had to make the following changes:

1. Changed class FileProcessor to class Processor.
2. Changed ReadFileDataToList to read\_data\_from\_file.
3. Changed WriteListDataToFile to write\_data\_to\_file.
4. Changed ShowCurrentItemsInList to print\_current\_tasks\_in\_list
5. Changed AdditemtoList to add\_data\_to\_list
6. Changed OutputMenuItems to print\_menu\_tasks
7. Changed InputMenuChoice to input\_menu\_choice
8. Changed GetTaskandPriority to input\_new\_task\_and\_priority

Although it may have been acceptable to leave the variable names the way they were in the original starter code that I used for the assignment, I decided to go ahead and change the variable names because it made the code more consistent with the standard for using lower case variable names with “snake” casing. Also, it gave me practice with using the “edit – find – replace” command in PyCharm for replacing all instances of a variable name, so it was good practice. I also changed the order of the functions to agree with the revised starter code. This should make it easer for me to follow/understand the code that is written by other students as well as the solution that will be posted by the instructor. Additional changes:

1. Removed menu choice (1) for showing current data. Made that automatic at the beginning of the “while” loop.
2. Renumbered menu items and renumbered the choices in the code.
3. Added the return parameter “success” to each of the processing function return statements

After convert my existing code by renaming variables, removing menu options, renumbering the menu, and renumbering the choices in the code, I then added the word “Success” to the return values for all the processing functions. This gave me an error message telling me that the list row indices needed to be “integers” and not “text” values. Since I had not made any changes to the list or the indices, I was stumped for a substantial period of time by this error message. I hit “undo” and tested the code repeatedly until I got to the point where I no longer received the error message. I was able to isolate the problem to having added “Success” to the list of return values without unpacking the tuple to indicate that the function was returning more than one parameter. To resolve the problem I changed the code to unpack the tuple to get the value of the strStatus. For example, the code in Figure 16 was changed to the code in Figure 17 to resolve the error message.

lstTable = Processor.add\_data\_to\_list(strTask, strPriority, lstTable)

Figure - Code before adding "Success" as a Status

lstTable, strStatus = Processor.add\_data\_to\_list(strTask, strPriority, lstTable)

Figure - Code to unpack tuple to capture status

Figure 18 shows screenshots of the final code running in PyCharm

A screenshot of a cell phone

Description automatically generated

A screenshot of a cell phone

Description automatically generated

A screenshot of a cell phone

Description automatically generated

A picture containing black

Description automatically generated

Figure 18 - Program run in PyCharm

# Running the Code in the Command Window

The code ran in PyCharm but did not run in the command window. To resolve this problem, it was necessary to change directories so that I was in the path for Assignment06 and then it ran properly as shown in Figure 19. Notice the error message on the first attempt to run the program, then the change directory command, and finally, the program running as expected.

A screenshot of a cell phone

Description automatically generated

A screenshot of a cell phone

Description automatically generated

Figure - Program run in CMD window

# Summary

This was the most complex assignment yet as the code is getting longer and I am building upon my programming knowledge. I used starter code which initially made it easier to “fill in the blanks.” I could easily locate the “ToDo” items, move the code into a function and replace it with the appropriate function call statements. This went smoothly for all of the functions except the “remove” function where, although I set a Boolean value to “false” in the function, I had not passed the value back to the main body of the program. This was the hardest part of troubleshooting this code, until I found out that I was editing the wrong code.

This more serious complication was the fact that I began with “starter code” that was later modified so that there were two different sets of starter code. This was discovered after I had finished writing all my code except for the “remove” function. Since I was so close to finishing, I was tempted to just finish the “remove” function and submit my assignment. Instead, I decided to convert my existing code by renaming variables, removing menu options, renumbering the menu, and renumbering the choices in the code.

What seemed like the simplest of changes (adding the word “Success” to the processing functions) turned out to be the most difficult because it gave me an error message that I could not relate to the changes I had made. The error messages referred to indices in a list. Performing this troubleshooting took longer than the combined time spent writing all the other functions. The problem was simply that in returning more than one variable, the values were automatically put into a tuple and needed to be unpacked, but the error message did not really help with that.

After I added the correct code to unpack the tuple, I continued with working on the functions in the code and I fixed the problem with the “Remove” function and tested it in the final version. Overall, it was a good learning experience for identifying redundant activities in code that could be moved to a function, it helped with getting the proper industry preferred architecture for coding, and helped me learn the art of calling functions with the necessary arguments. It gave me practice with debugging, finding, renaming, and re-structuring code as well as reinforcing the fact that if you return multiple items from a tuple then you have to unpack the tuple to extract the values.

# Final Note

After losing sleep over what to do with the “success” string, I decided to just submit my assignment and stop worrying about it. After submitting it, I then proceeded with the rest of my assignment which was to review other student’s papers and their code. I found a brilliant solution for what to do with the elusive “success” string. The string is replaced by the definition of success for that function! So, If the goal of the function is to read data from a file, the string “success” should be replaced by “Data was read from file” and then passed into the body of the function with the table and unpacked as a tuple and printed using the IO call. Likewise, “success” for adding data to a file should read “Data added to the file.” Success for writing data to the file should read “Data written to file.” For removing the data it was a bit more complex and I decided it should say “Data removed from file is: “ and then append the string of the Boolean value of whether it was true or false. Credit goes to Mr. Erik Knighton for this solution and I am not sure if I am allowed to use my updated code for grading purposes…or is that the purpose of GitHub and collaboration? We shall see! All screenshots in this paper are from the program before it was amended per Mr. Knighton’s solution.

Also, Figure 20 shows a screenshot of the ToDo list as amended by all of my iterations through the code.

A screenshot of a cell phone

Description automatically generated

Figure - ToDo List