Darivanh Vlachos

February 21, 2021

IT FDN 110 B Wi 21: Foundations of Programming: Python

Assignment06

https://darivanhatuw.github.io/IntroToProg-Python-Mod06/

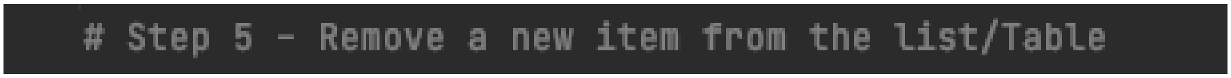
What's Your Function?

Introduction

This week's assignment challenges students by tasking them to manipulate data in dictionary rows and tables separated into functions within a class then calling said functions. Most of the code from Assignment05 was applicable in Assignment06 with a few exceptions: 1) Selective data removal from a list in response to user input, 2) Separate functions by input and output as well as processing, and 3) Call data from a main body. Once I was able to overcome my challenge of packing and unpacking tuples from the function to the "Main Body" I was able to better understand functions.

Selective Data Removal

In Assignment05, I misunderstood pseudo code "Remove a new item from the list/table," Figure 1, in the Assignment05_starter.py starter code, which I interpreted to mean the removal of the newest entry, or the last entry, in the table, which, to me, meant using the .pop() function.

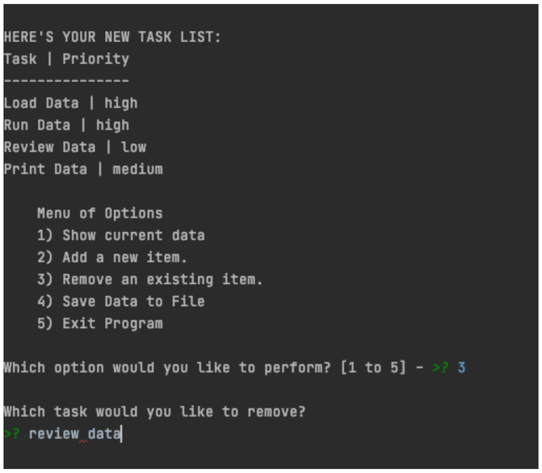


```
# Step 5 - Remove a new item from the list/Table
```

Figure 1 Assignment05's pseudo code that instructs students remove a new item from the list/Table.

Instead, users were supposed to choose the data they wanted to delete from their task list, via user input, which requires the program to scan through the task list, each dicRow within the lstTable, for a matching string within the lstTable and delete the dicRow in which the string is located. Below is the step-by-step illustration of the looping process in which the removal of the selected dicRow occurs.

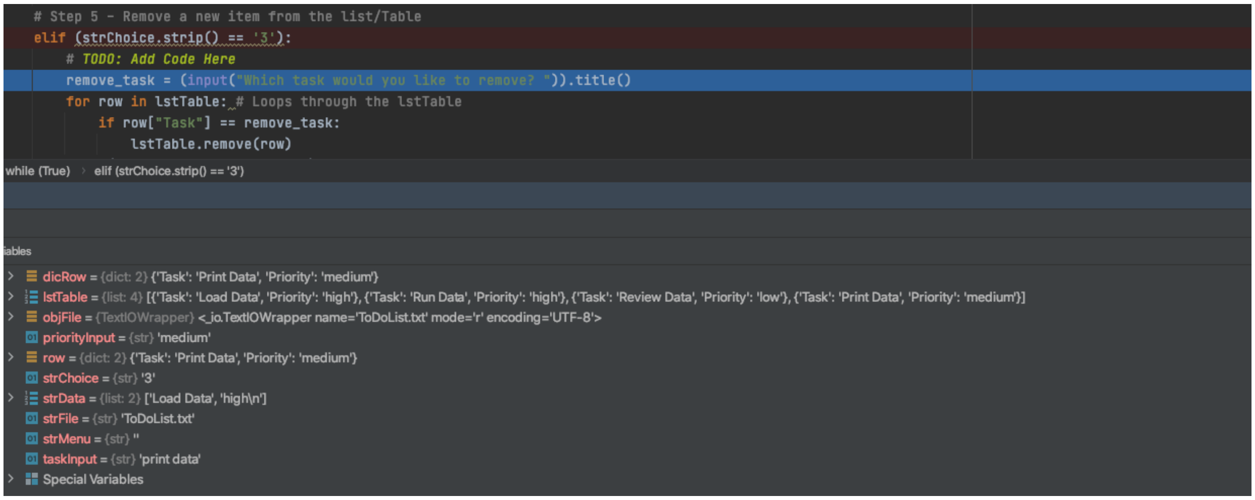
After selecting “3” Remove an existing item, the user is prompted to enter the task to be removed, (Figure 2).



```
HERE'S YOUR NEW TASK LIST:
Task | Priority
--------------
Load Data | high
Run Data | high
Review Data | low
Print Data | medium

    Menu of Options
    1) Show current data
    2) Add a new item.
    3) Remove an existing item.
    4) Save Data to File
    5) Exit Program

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

Which task would you like to remove?
>? review_data
```

Figure 2 Screenshot of the Debugging feature in PyCharm illustrating the user inputting the item they want removed.

Meanwhile, the program enters the elif statement and initiates a prompt to the user. Below, Figure 3, is the program in the Debugging mode within PyCharm capturing the moment the program prompts the user for input. Underneath the code is the Debugger window that allows the programmer to view all the stored data and the data being processed.

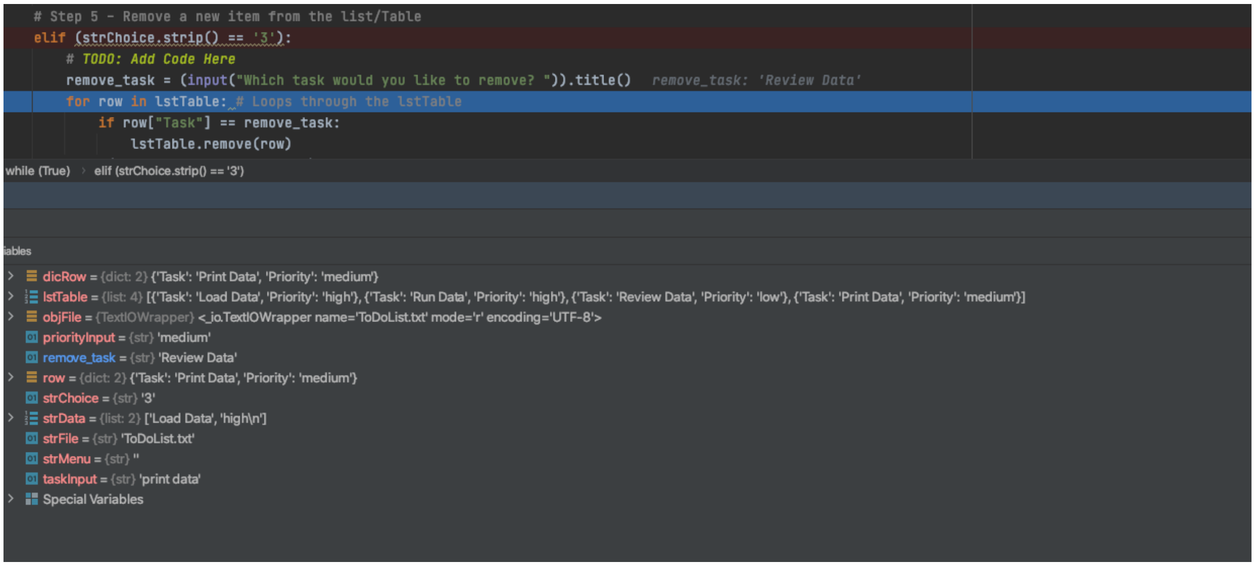


```
    # Step 5 - Remove a new item from the list/Table
    elif (strChoice.strip() == '3'):
        # TODO: Add Code Here
        remove_task = (input("Which task would you like to remove? ")).title()
        for row in lstTable: # Loops through the lstTable
            if row["Task"] == remove_task:
                lstTable.remove(row)
```

```
while (True) › elif (strChoice.strip() == '3')

iables
> dicRow = {dict: 2} {'Task': 'Print Data', 'Priority': 'medium'}
> lstTable = {list: 4} [{'Task': 'Load Data', 'Priority': 'high'}, {'Task': 'Run Data', 'Priority': 'high'}, {'Task': 'Review Data', 'Priority': 'low'}, {'Task': 'Print Data', 'Priority': 'medium'}]
> objFile = {TextIOWrapper} <_io.TextIOWrapper name='ToDoList.txt' mode='r' encoding='UTF-8'>
  priorityInput = {str} 'medium'
> row = {dict: 2} {'Task': 'Print Data', 'Priority': 'medium'}
  strChoice = {str} '3'
> strData = {list: 2} ['Load Data', 'high\n']
  strFile = {str} 'ToDoList.txt'
  strMenu = {str} ''
  taskInput = {str} 'print data'
> Special Variables
```

Figure 3 Screenshot of the Debugging feature in PyCharm illustrating the program as it progresses throughout the elif statement from Assignment05.

The program then enters the for loop in which it will scan the contents of the lstTable containing the dictionary of the task list, Figure 4.



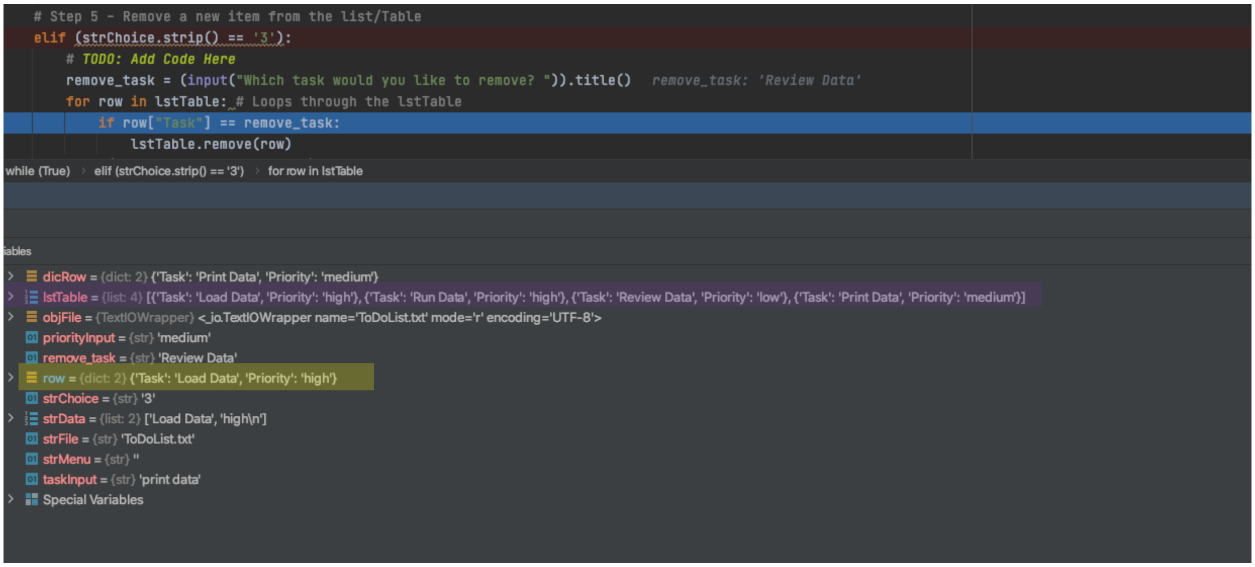
```
# Step 5 - Remove a new item from the list/Table
elif (strChoice.strip() == '3'):
    # TODO: Add Code Here
    remove_task = (input("Which task would you like to remove? ")).title()   remove_task: 'Review Data'
    for row in lstTable: # Loops through the lstTable
        if row["Task"] == remove_task:
            lstTable.remove(row)
```

```
while (True) › elif (strChoice.strip() == '3')
```

```
iables
> dicRow = {dict: 2} {'Task': 'Print Data', 'Priority': 'medium'}
> lstTable = {list: 4} [{'Task': 'Load Data', 'Priority': 'high'}, {'Task': 'Run Data', 'Priority': 'high'}, {'Task': 'Review Data', 'Priority': 'low'}, {'Task': 'Print Data', 'Priority': 'medium'}]
> objFile = {TextIOWrapper} <_io.TextIOWrapper name='ToDoList.txt' mode='r' encoding='UTF-8'>
  priorityInput = {str} 'medium'
  remove_task = {str} 'Review Data'
> row = {dict: 2} {'Task': 'Print Data', 'Priority': 'medium'}
  strChoice = {str} '3'
> strData = {list: 2} ['Load Data', 'high\n']
  strFile = {str} 'ToDoList.txt'
  strMenu = {str} ''
  taskInput = {str} 'print data'
> Special Variables
```

Figure 4 Screenshot of the Debugging feature in PyCharm that illustrates the for loop in which the program progresses to find a matching string within the item.

In Figure 5, the program checks to see if the key, 'Task', matches the remove_task, input string from the user. The program checks the contents within the lstTable, highlighted in purple, for key value matches in order, the first dicRow, in this case, being 'Task': 'Load Data', 'Priority': 'high', highlighted in yellow. The program will continue to loop through the list until there are no more items within the lstTable.



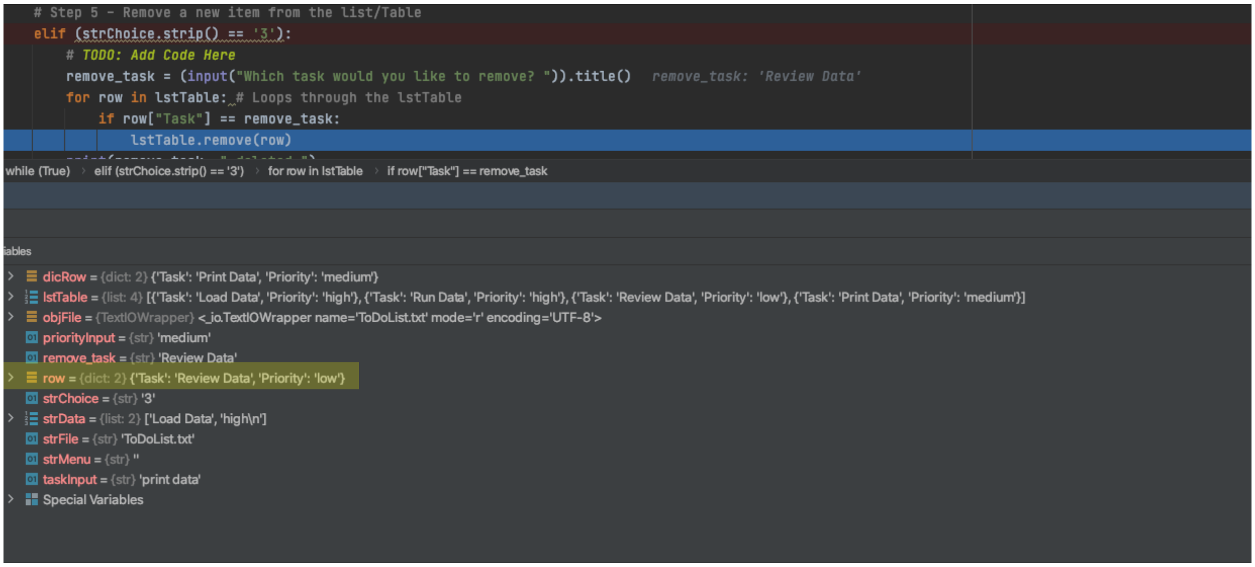
```
# Step 5 - Remove a new item from the list/Table
elif (strChoice.strip() == '3'):
    # TODO: Add Code Here
    remove_task = (input("Which task would you like to remove? ")).title()   remove_task: 'Review Data'
    for row in lstTable: # Loops through the lstTable
        if row["Task"] == remove_task:
            lstTable.remove(row)
```

```
while (True) › elif (strChoice.strip() == '3') › for row in lstTable
```

```
iables
> dicRow = {dict: 2} {'Task': 'Print Data', 'Priority': 'medium'}
> lstTable = {list: 4} [{'Task': 'Load Data', 'Priority': 'high'}, {'Task': 'Run Data', 'Priority': 'high'}, {'Task': 'Review Data', 'Priority': 'low'}, {'Task': 'Print Data', 'Priority': 'medium'}]
> objFile = {TextIOWrapper} <_io.TextIOWrapper name='ToDoList.txt' mode='r' encoding='UTF-8'>
  priorityInput = {str} 'medium'
  remove_task = {str} 'Review Data'
> row = {dict: 2} {'Task': 'Load Data', 'Priority': 'high'}
  strChoice = {str} '3'
> strData = {list: 2} ['Load Data', 'high\n']
  strFile = {str} 'ToDoList.txt'
  strMenu = {str} ''
  taskInput = {str} 'print data'
> Special Variables
```

Figure 5 Screenshot of the Debugging feature of PyCharm that illustrates the program scanning the lstTable for a matching item to the user input string.

Once the program has found a matching key value, Figure 6 - highlighted in yellow, the program will enter the nested if statement where it will then remove the item from the lstTable.



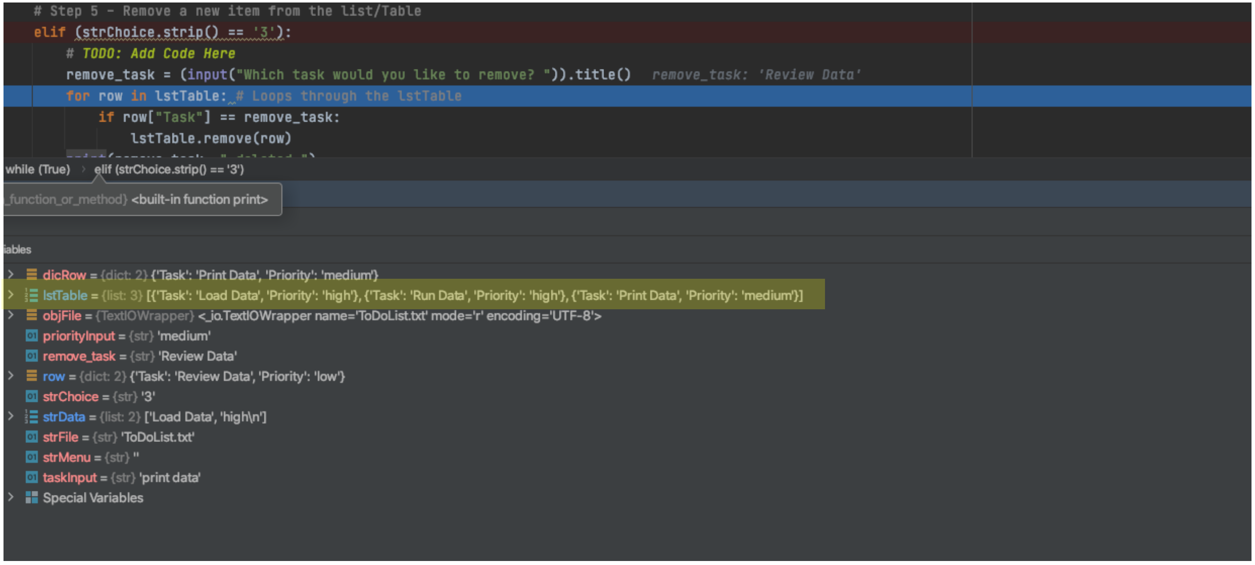
```
# Step 5 - Remove a new item from the list/Table
elif (strChoice.strip() == '3'):
    # TODO: Add Code Here
    remove_task = (input("Which task would you like to remove? ")).title()   remove_task: 'Review Data'
    for row in lstTable: # Loops through the lstTable
        if row["Task"] == remove_task:
            lstTable.remove(row)
```

```
while (True) › elif (strChoice.strip() == '3') › for row in lstTable › if row["Task"] == remove_task
```

```
iables
> dicRow = {dict: 2} {'Task': 'Print Data', 'Priority': 'medium'}
> lstTable = {list: 4} [{'Task': 'Load Data', 'Priority': 'high'}, {'Task': 'Run Data', 'Priority': 'high'}, {'Task': 'Review Data', 'Priority': 'low'}, {'Task': 'Print Data', 'Priority': 'medium'}]
> objFile = {TextIOWrapper} <_io.TextIOWrapper name='ToDoList.txt' mode='r' encoding='UTF-8'>
  priorityInput = {str} 'medium'
  remove_task = {str} 'Review Data'
> row = {dict: 2} {'Task': 'Review Data', 'Priority': 'low'}
  strChoice = {str} '3'
> strData = {list: 2} ['Load Data', 'high\n']
  strFile = {str} 'ToDoList.txt'
  strMenu = {str} ''
  taskInput = {str} 'print data'
> Special Variables
```

Figure 6 Screenshot of the Debugging feature in PyCharm that illustrates the progression of the program once it has found a matching item to the user input string.

The program will then continue through the for loop until there are no longer matching statements and items within the lstTable. Below, Figure 7, you notice that the 'Review Data' item is no longer in lstTable, highlighted in yellow.



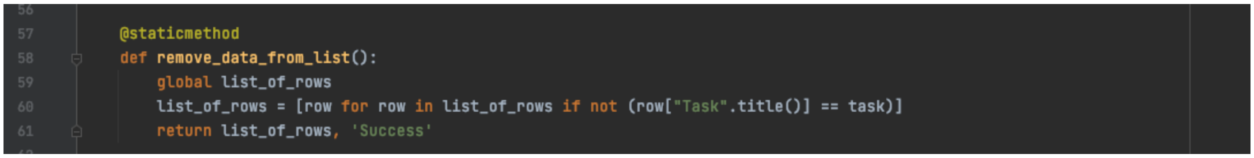
```
# Step 5 - Remove a new item from the list/Table
elif (strChoice.strip() == '3'):
    # TODO: Add Code Here
    remove_task = (input("Which task would you like to remove? ")).title()   remove_task: 'Review Data'
    for row in lstTable: # Loops through the lstTable
        if row["Task"] == remove_task:
            lstTable.remove(row)
```

```
while (True) › elif (strChoice.strip() == '3')
_function_or_method} <built-in function print>
```

```
iables
> dicRow = {dict: 2} {'Task': 'Print Data', 'Priority': 'medium'}
> lstTable = {list: 3} [{'Task': 'Load Data', 'Priority': 'high'}, {'Task': 'Run Data', 'Priority': 'high'}, {'Task': 'Print Data', 'Priority': 'medium'}]
> objFile = {TextIOWrapper} <_io.TextIOWrapper name='ToDoList.txt' mode='r' encoding='UTF-8'>
  priorityInput = {str} 'medium'
  remove_task = {str} 'Review Data'
> row = {dict: 2} {'Task': 'Review Data', 'Priority': 'low'}
  strChoice = {str} '3'
> strData = {list: 2} ['Load Data', 'high\n']
  strFile = {str} 'ToDoList.txt'
  strMenu = {str} ''
  taskInput = {str} 'print data'
> Special Variables
```

Figure 7 Screenshot of the Debugging feature in PyCharm that illustrates the progression of the program once it has deleted the matching item, the lstTable is then updated and the program continues to loop.

My personal challenge for Assignment06 was not only to find a way to remove specific data within the lstTable but to do so through a program organized with separate functions. My initial code, Figure 8, seemed to pull that same type of data looping through the list_of_rows, but did not seem to update the global list_of_rows. My solution was to add a global variable, forgetting that in the February 16, 2021 video lecture for ITDFN110B Intro to Programming – Python class, lecturer Randal Root introduced the class to packing and unpacking tuples.



```
    @staticmethod
    def remove_data_from_list():
        global list_of_rows
        list_of_rows = [row for row in list_of_rows if not (row["Task".title()] == task)]
        return list_of_rows, 'Success'
```

Figure 8 Screenshot of my initial for loop with an if not statement and global variable.

During a study session with a few of my classmates, we determined that there may have been something wrong with the loop I presented. And, although we were unsure what it could be, we all agreed to break the loop out into three lines as seen in Figure 3 because we all knew that the code worked and were familiar with that style of coding. At the end of the session, we also discovered that I had not unpacked my tuple in the "Main Body" of the program. With those adjustments I was able to make the code run without the global variable. I eventually tested the old loop, and realized that it was in fact the lack of unpacking of the tuple within the "Main Body." I will keep the three lined code for removal for Assignment06, but I now have multiple codes to apply to my repertoire if I so choose. At this point I am unsure which code is more beneficial to my overall program, but it may come to me as I continue to learn more about Python.

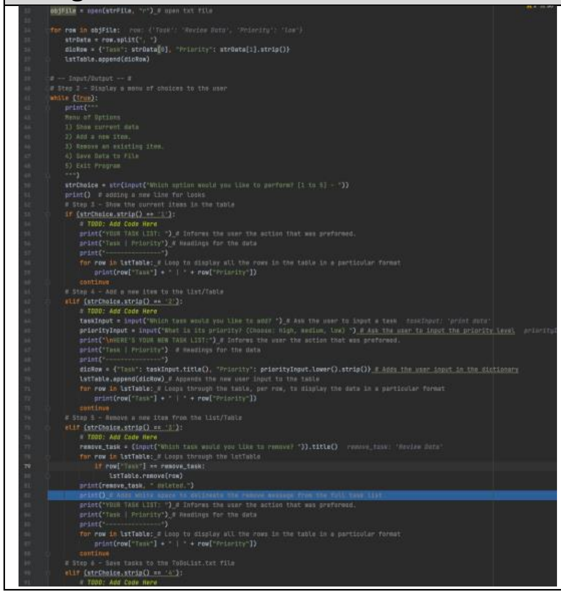
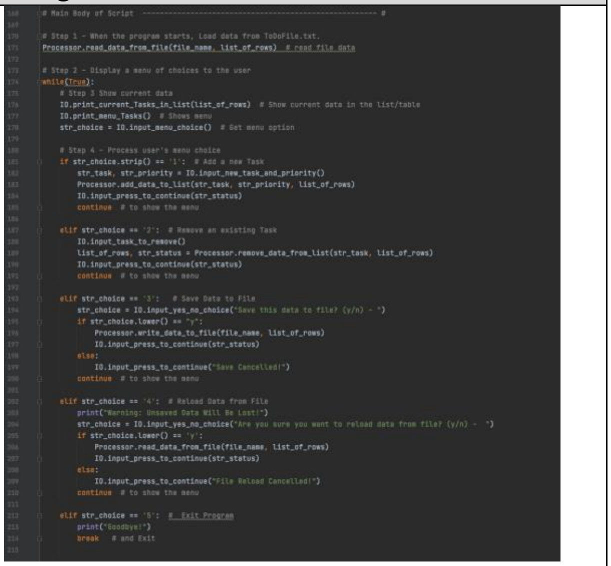
Organizing Functions

When initially tackling Assignment06, students were provided with most of the code necessary for the assignment as most of the code could be pulled from Assignment05. What is a learning opportunity if we aren't provided with challenges, right? My personal challenge focused on my inability to recall packing and unpacking of tuples from its corresponding functions. Organizing the program into functions that could be called multiple times and within processing and input/output groupings, classes, allows programmers the flexibility and declutters the "Main Body." Once I was reminded about the packing and unpacking of tuples throughout the program I was able to fully understand how unnecessary global variables (at least within this program) were.

Calling Functions in the Main Body

As I reflect on Assignment05, it is absolutely clear why organizing a program into classes and providing a "Main Body" allows for a cleaner structure that is easy to follow eventually easy to edit, if needed. Below, Table 1, is a comparison of my "Main Body" from Assignment05 and Assignment06. Aesthetically speaking Assignment05 seems messy due to its repetitive use of the print command, whereas in Assignment06 the print command has minimal impact in the "Main Body" as most of the repetitive print statements are provided in one function. If one need to make adjustments to the code in Assignment06, it would then be done in the functions, whereas it would require quite a bit of work and repetitive editing in Assignment05.

Table 1 Comparing the "Main Body" of both Assignment05 and Assignment06 to demonstrate how necessary organizing classes, functions and a "Main Body" remarkably improves readability within ones code.

Assignment05	Assignment06
	

```python
objFile = open(strFile, "r")_# open txt file

for row in objFile:  row: {'Task': 'Review Data', 'Priority': 'low'}
    strData = row.split(", ")
    dicRow = {"Task": strData[0], "Priority": strData[1].strip()}
    lstTable.append(dicRow)

# -- Input/Output -- #
# Step 2 - Display a menu of choices to the user
while (True):
    print("""
    Menu of Options
    1) Show current data
    2) Add a new item.
    3) Remove an existing item.
    4) Save Data to File
    5) Exit Program
    """)
    strChoice = str(input("Which option would you like to perform? [1 to 5] - "))
    print()  # adding a new line for looks
    # Step 3 - Show the current items in the table
    if (strChoice.strip() == '1'):
        # TODO: Add Code Here
        print("YOUR TASK LIST: ")_# Informs the user the action that was preformed.
        print("Task | Priority")_# Headings for the data
        print("-----------------")
        for row in lstTable:_# Loop to display all the rows in the table in a particular format
            print(row["Task"] + " | " + row["Priority"])
        continue
    # Step 4 - Add a new item to the list/Table
    elif (strChoice.strip() == '2'):
        # TODO: Add Code Here
        taskInput = input("Which task would you like to add? ")_# Ask the user to input a task  taskInput: 'print data'
        priorityInput = input("What is its priority? (Choose: high, medium, low) ")_# Ask the user to input the priority level  priority
        print("\nHERE'S YOUR NEW TASK LIST:")_# Informs the user the action that was preformed.
        print("Task | Priority")  # Headings for the data
        print("-----------------")
        dicRow = {"Task": taskInput.title(), "Priority": priorityInput.lower().strip()}_# Adds the user input in the dictionary
        lstTable.append(dicRow)_# Appends the new user input to the table
        for row in lstTable:_# Loops through the table, per row, to display the data in a particular format
            print(row["Task"] + " | " + row["Priority"])
        continue
    # Step 5 - Remove a new item from the list/Table
    elif (strChoice.strip() == '3'):
        # TODO: Add Code Here
        remove_task = (input("Which task would you like to remove? ")).title()  remove_task: 'Review Data'
        for row in lstTable:_# Loops through the lstTable
            if row["Task"] == remove_task:
                lstTable.remove(row)
        print(remove_task, " deleted.")
        print()_# Adds white space to delineate the remove message from the full task list
        print("YOUR TASK LIST: ")_# Informs the user the action that was preformed.
        print("Task | Priority")_# Headings for the data
        print("-----------------")
        for row in lstTable:_# Loop to display all the rows in the table in a particular format
            print(row["Task"] + " | " + row["Priority"])
        continue
    # Step 6 - Save tasks to the ToDoList.txt file
    elif (strChoice.strip() == '4'):
        # TODO: Add Code Here
```

```python
# Main Body of Script  ------------------------------------------------------ #

# Step 1 - When the program starts, Load data from ToDoFile.txt.
Processor.read_data_from_file(file_name, list_of_rows)  # 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(list_of_rows)  # Show current data in the list/table
    IO.print_menu_Tasks()  # Shows menu
    str_choice = IO.input_menu_choice()  # Get menu option

    # Step 4 - Process user's menu choice
    if str_choice.strip() == '1':  # Add a new Task
        str_task, str_priority = IO.input_new_task_and_priority()
        Processor.add_data_to_list(str_task, str_priority, list_of_rows)
        IO.input_press_to_continue(str_status)
        continue  # to show the menu

    elif str_choice == '2':  # Remove an existing Task
        IO.input_task_to_remove()
        list_of_rows, str_status = Processor.remove_data_from_list(str_task, list_of_rows)
        IO.input_press_to_continue(str_status)
        continue  # to show the menu

    elif str_choice == '3':   # Save Data to File
        str_choice = IO.input_yes_no_choice("Save this data to file? (y/n) - ")
        if str_choice.lower() == "y":
            Processor.write_data_to_file(file_name, list_of_rows)
            IO.input_press_to_continue(str_status)
        else:
            IO.input_press_to_continue("Save Cancelled!")
        continue  # to show the menu

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

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

Summary

Overcoming challenges is always tough. Understanding new ways to code, as in Assignment06 with 1) Selective data removal from a list in response to user input, 2) Separate functions by input and output as well as processing, and 3) Call data from a main body, reinforced that sentiment. For me, understanding the communication between functions and tuples strengthened my overall knowledge of the program. Moving forward, I feel confident that I can build cleaner more organized programs.

References

Root, Randall. *YouTube*. 25 January 2021. https://www.youtube.com/watch?v=jiXmXhwgHp8&feature=youtu.be&ab_channel=IntelliJIDEAbyJetBrains. 17-22 Feb 2021.