Erik Knighton

5/22/2020

IT FDN 100 A

Assignment06

<https://github.com/erikjkUW/IntroToProg-Python-Mod06>

Functional Task Management

# Introduction

I have been enjoying the iterative nature of this course. This assignment was more difficult, but incrementally so, and because we have already experienced functions in a limited sense this feels more like a step than a climb. Again, working with someone else's script framework was disorienting at first, and I had to perform some trial and error for certain sections; however, there were some surprises there as well, which I will describe below.

# Defining Functions

I tackled this week's module in a similar fashion to the previous one. First, I worked through the functions I knew I could accomplish easily: IO Add and Remove. Once I had finished those, testing with simple print statements to make sure they operated correctly, I moved on. Next was the two associated Processor Add and Remove functions. Everything looked good, but for the life of me I could not get the IO to share its values with the Processor functions.

Here, I made my first mistake. In an attempt to solve my issue of the IO not communicating its variable values with its Processor brethren, I nested the latter within the former. I kept getting errors, yet they weren't about not having missing arguments, instead the list was not getting updated correctly, or just jumping over the Processor functions altogether.

I had believed that I could handle this part of the assignment, so familiar and comprehendible, before I had watched the class video or read through the module document. This was in large part to disruptions in my typical daily routine, but I know that I had let my hubris command me into ineptitude. Hindsight and all that. I resolved to take a break from my computer and to review the textbook and return the next day.

I figured out how to not only get the 'Success' string to print, but also how it should work with the strStatus global variable, the optional\_message argument in IO.input\_press\_to\_continue function, and how to fit it all into the Main Body of the script. I will keep the aside here, as it originally was submitted, and then have added a new subsection **Printing Success** below, as well as some screen captures showing the new code working as intended. (As a quick aside, one question I had, and I will make a point to post this to the module discussion board, was why there were 'Success' strings in most of the Processor function return lines, and if I should include them in my own return lines. Googling strings in return lines failed to clarify their inclusion, nor do I recall seeing them mentioned in the video or documents for the assignment. Also, the docstrings I added to the functions I wrote are inadequate, I'm sure, but I was unclear whether I needed to add all of the variables within the scope of each individual function as parameters.)

Moving on with the script itself, I spent a lot of time pondering where I went wrong. My primary lapse was to resort to global variables, such as strTask and strPriority, as a bandage where the script functioned, but was not in keeping with the language's graphic aesthetic. For the interim I left them in, avowing to resolve their blight on my otherwise elegant script, avoiding that trap in the future.

Another pitfall I fell into, related to the misuse of global variables, was to forget to use the global variables as arguments in the Main Body of the code. Second verse, same as the first. After I watched the video and worked through the examples in the labs I realized my mistake, and was able to alter my code. Running multiple revisions, about a dozen or so, all of which worked, my code finally looked to be not only without grammatical error, but to maintain proper stylistic integrity. On the following pages is what I ended up with.

I chose in this document to keep the font size reduced to preserve lines, and because the script will be readable in the editor and on my GitHub repository:

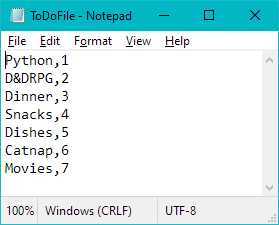
#### # ---------------------------------------------------------------------------- # # Title: Assignment 06 # Description: Working with functions in a class, # When the program starts, load each "row" of data # in "ToDoToDoList.txt" into a python Dictionary. # Add the each dictionary "row" to a python list "table" # ChangeLog (Who,When,What): # RRoot,1.1.2030,Created started script # RRoot,1.1.2030,Added code to complete assignment 5 # erikjk,5.21.2020,Compared previous script to framework on this assignment # erikjk,5.21.2020,Completed IO for Add and Remove steps # erikjk,5.21.2020,Completed Processor for Add and Remove steps, nested in IO # erikjk,5.21.2020,Completed Processor for Write step # erikjk,5.22.2020,Moved global variables as arguments to Main Body from IO functions # erikjk,5.22.2020,Removed nested Processor functions from functions (Whoops!) # erikjk,5.22.2020,Cleaning up function docstrings and comments # erikjk,5.22.2020,Running and testing, checking debugger # erikjk,5.25.2020,Connecting the dots of three missing pieces of existing code # erikjk,5.25.2020,Return strings for strStatus variable properly print # erikjk,5.25.2020,Double checking all functions work, resubmitting assignment # ---------------------------------------------------------------------------- # # Data ---------------------------------------------------------------------- # # 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 an processing functions # 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: task, priority = line.split(",") row = {"Task": task.strip(), "Priority": priority.strip()} list\_of\_rows.append(row) file.close() return list\_of\_rows, "\nYour Tasks have been loaded from the document.\n" @staticmethod def add\_data\_to\_list(task, priority, list\_of\_rows): *""" Assigns the user's input values to dictionary keys Appends new dictionary to list :return: (list) of dictionary rows """* row = {"Task": task.strip(), "Priority": priority.strip()} list\_of\_rows.append(row) return list\_of\_rows, "\nThe task \"" + task.title() + "\" has been added.\n" @staticmethod def remove\_data\_from\_list(task, list\_of\_rows): *""" Deletes the dictionary value that matches the user's input :return: (list) of dictionary rows """* for row in list\_of\_rows: if row["Task"].lower() == task.lower(): list\_of\_rows.remove(row) return list\_of\_rows, "\nThe task \"" + task.title() + "\" has been removed.\n" @staticmethod def write\_data\_to\_file(file\_name, list\_of\_rows): *""" Opens file and overwrites new list data into the file :return: (list) of dictionary rows """* file = open(file\_name, "w") for task in list\_of\_rows: file.write(task["Task"] + "," + task["Priority"] + "\n") file.close() return list\_of\_rows, "Your Tasks document has been successfully updated." # 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.') print() # Add an extra line for looks @staticmethod def input\_new\_task\_and\_priority(list\_of\_rows): *""" Captures input for Task and its assigned Priority :return: string """* task = str(input("What is the Task you wish to add? ").title()).strip() priority = str(input("What is the Priority for " + task.title() + "? ")).strip() return task, priority @staticmethod def input\_task\_to\_remove(list\_of\_rows): *""" Captures input for Task removal function :return: string """* task = str(input("Which Task do you wish to remove? ")).strip() return task # Main Body of 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 strTask, strPriority = IO.input\_new\_task\_and\_priority(lstTable) lstTable, strStatus = Processor.add\_data\_to\_list(strTask, strPriority, lstTable) IO.input\_press\_to\_continue(strStatus) continue # to show the menu elif strChoice == '2': # Remove an existing Task strTask = IO.input\_task\_to\_remove(lstTable) lstTable, strStatus = Processor.remove\_data\_from\_list(strTask, 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": lstTable, strStatus = 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': lstTable, strStatus = 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

# Running and Debugging

Satisfied with how everything looked, I made sure to double check all of my string methods were intact. I mainly used .lower() for checking values in conditionals, and .title() for storing the Task value in the dictionary, which would then be preserved in the text file. Using .strip() was a precaution in case the user entered a extraneous characters or spaces after the desired input. I feel like I am slowly understanding how complicated backend development can be in order for the frontend to keep its appearance and ease of use.

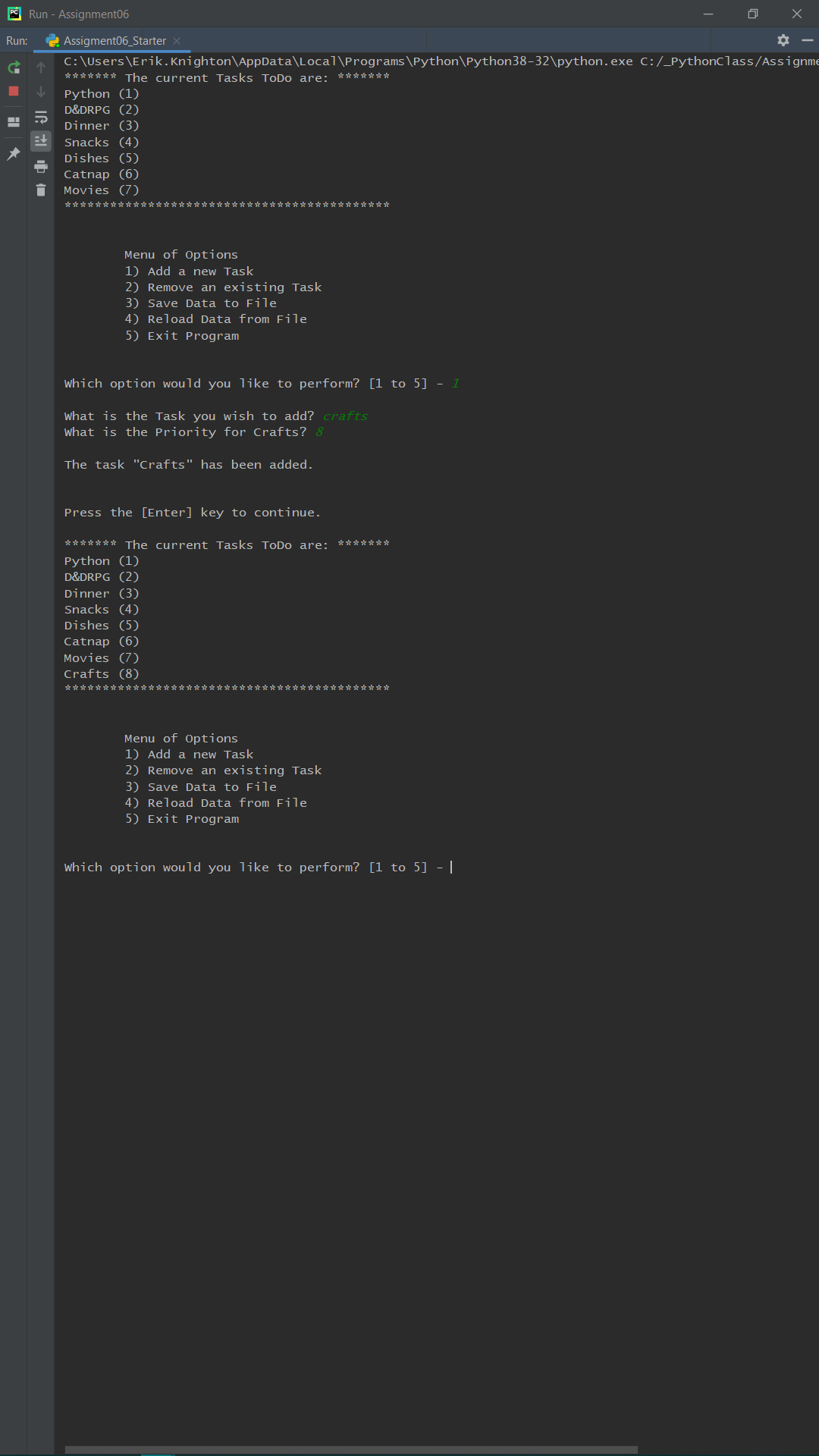
So many little things can go wrong, which also made me appreciate how different a lot of my classmates approached and completed the last assignment. Various skill sets, backgrounds, professions, and education all assembled to attack a common foe. I look forward to perusing this week's accomplishments as well.

Because I had run more than a few times through my program, the text file I began with looked different than when I started. I also chose, as I was building it by adding and removing line items in PyCharm, to manually sort them. I looked into sorting functions for Python, but the lambda approach I encountered everywhere I looked would've been nice, but I will wait until I understand it better. For now, this was the text I began editing (Figure 1).



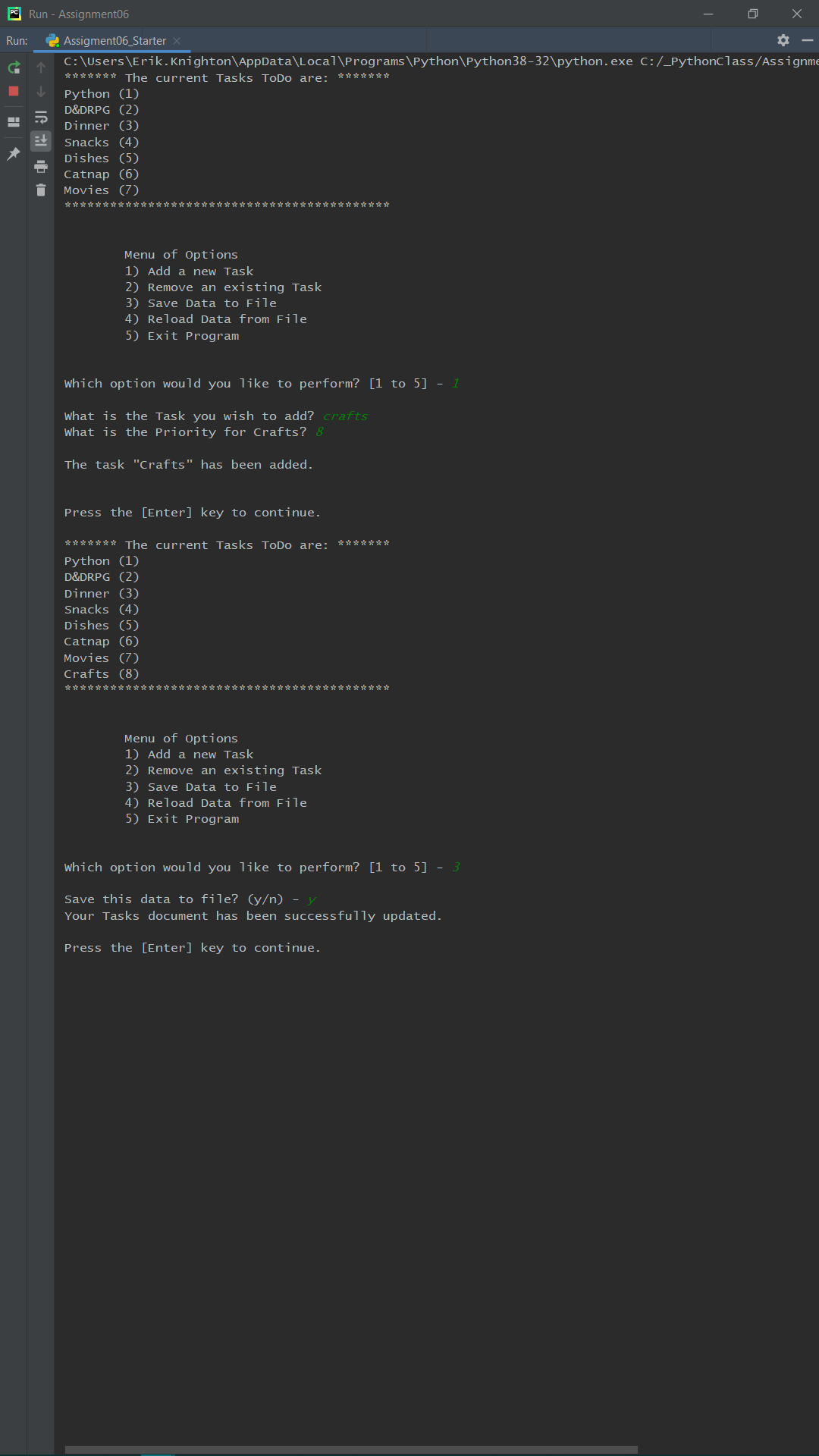
### Figure 1. Text Document with List of Tasks,Priority Values

Back to the code, I ran through PyCharm debugger, but it didn't reveal any problems. In future assignments, I will endeavor to debug multiple times throughout writing the script instead of relying on the error messages and trying to work backwards from their often cryptic lines. Content that I could run and sure that all of the menu option functions would perform their written tasks, I did one final run in PyCharm, adding Crafts (8) then capturing the image for this assignment (Figure 2).



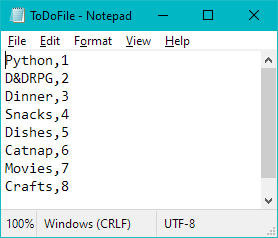
### Figure 1. Adding a New Task in PyCharm

The next step I performed was the Save Data to File option, and though the "n" input exception worked to display the message "Save Cancelled!" I decided not to display that here, as that code existed in the starter program (Figure 3).



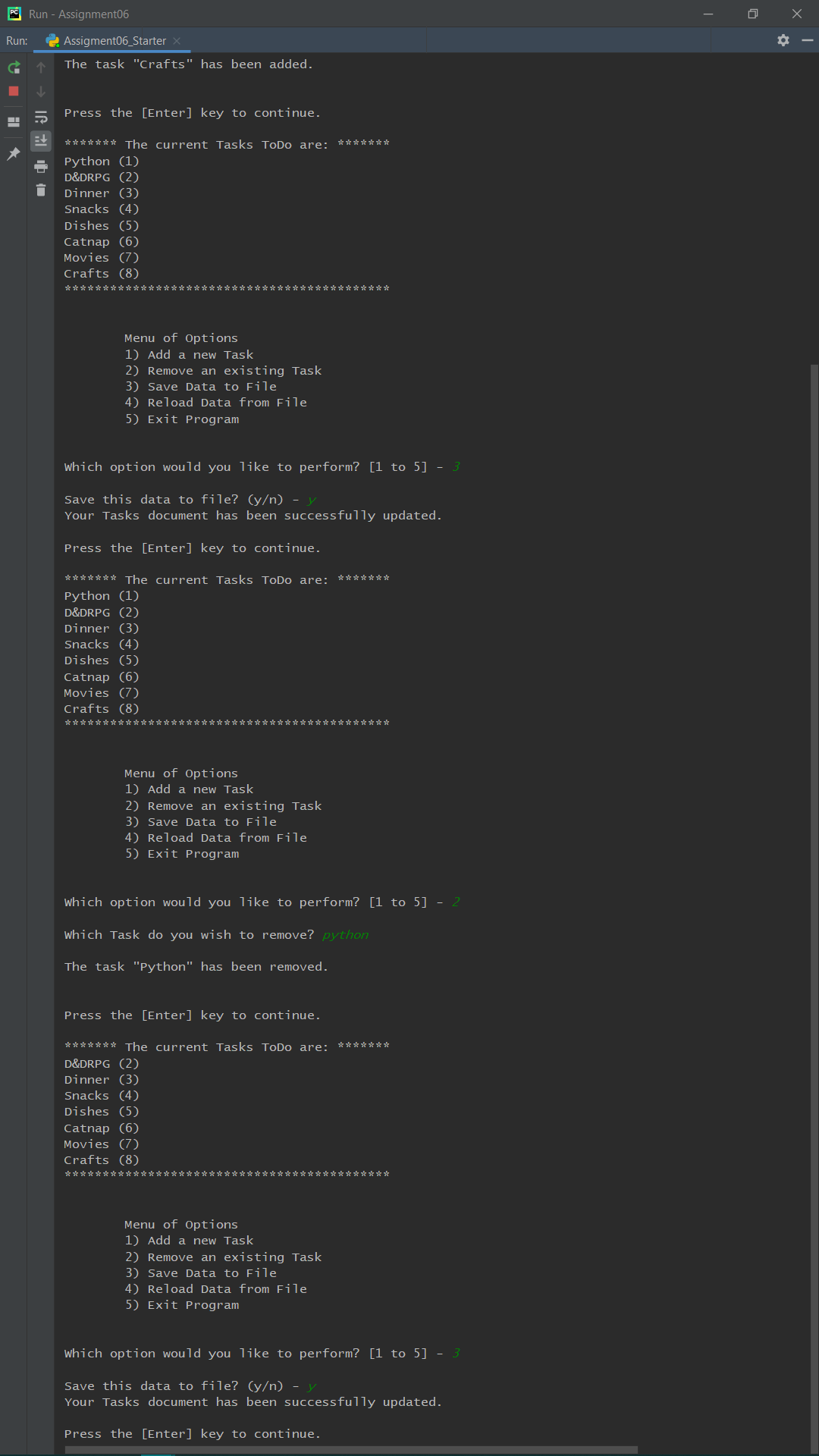
### Figure 3. Saving New Task Data to File in PyCharm

This allowed me to verify that the new task, Crafts (8) was correctly written to the file, which was the case (Figure 4).



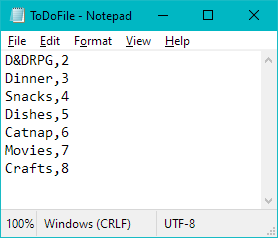
### Figure 4. Checking Text Document for New Task

The final step was to remove a task from the list and verify that the text document captured the change. In PyCharm I kept the program running and chose menu option 2 in order to remove a task from the list. I will mention here that as I did last time I chose to keep all of my tasks six characters long and all of my priorities a single digit in order to fake some orderliness until I know how to create a nice table within Python. The task removal IO and Processor functions both went off without a hitch (Figure 5).



### Figure 5. Saving after Task Removal in PyCharm

Here is the text file in its final form after adding Crafts (8) and removing Python (1), which means I'm done with the assignment, right (Figure 6)?



### Figure 6. Text Document after Task Removal

With all of that, I finished this assignment by creating my GitHub webpage <https://erikjkuw.github.io/IntroToProg-Python-Mod06/> (External Site). It displays only what I copied and pasted from the assignment document, which is a header and hyperlinks to the GitHub help and Google's search pages. My repository link is at the beginning of this paper, as requested. Once I am done typing this section, I will upload both this and my Python file to that repository.

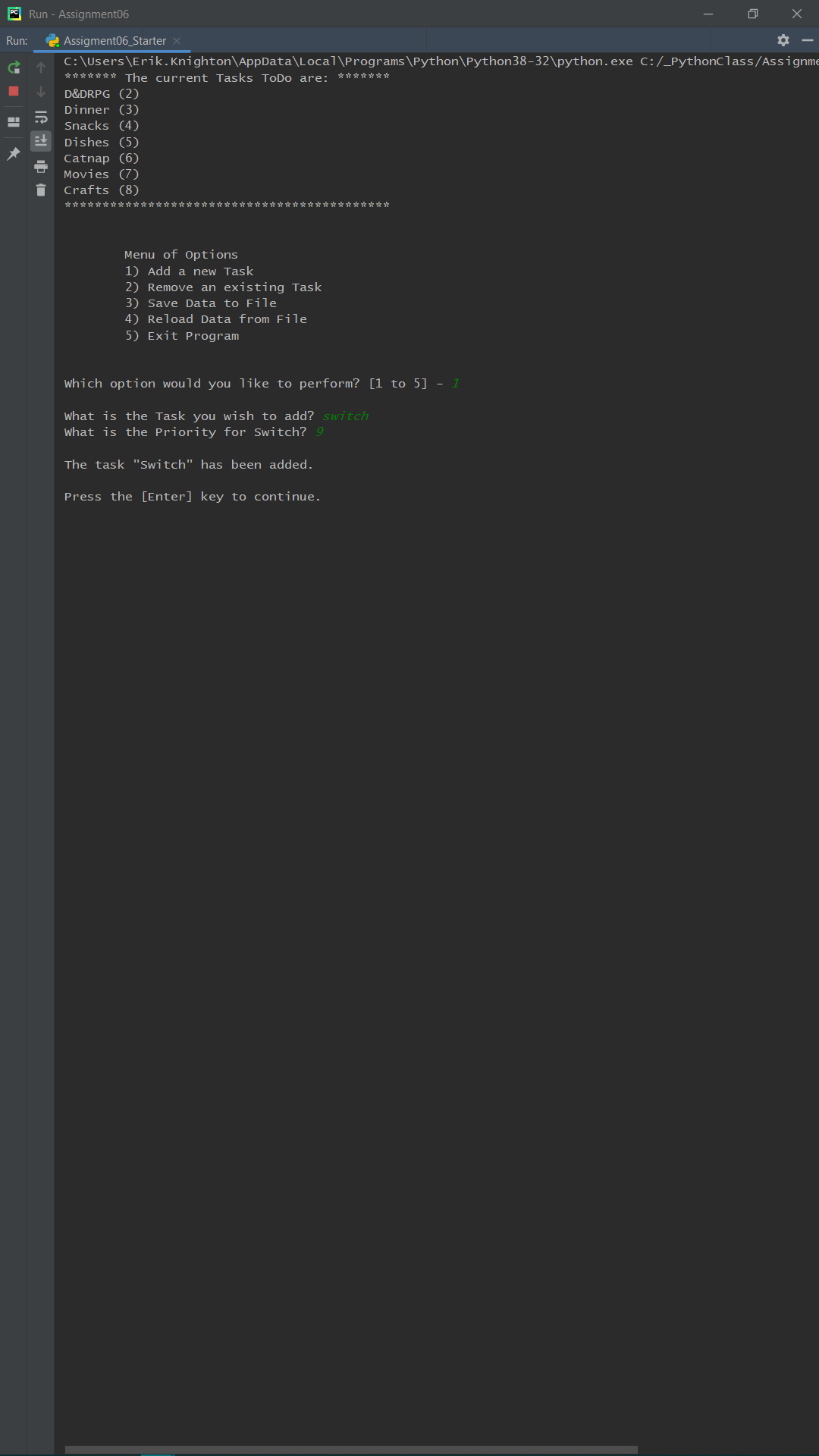
# Printing Success

Eureka! The issues with why there were strings in the return lines of the Processor functions eluded me until just this morning, the dawn of the assignment due date. What I realized is that where the optional\_message appeared in the IO.input\_press\_to\_continue function was looking for something to put there, and that 'Success' being returned fit too perfectly. I solved this little conundrum by changing the Main Body script to use these arguments whenever the status was needed:

#### lstTable, strStatus = Processor.function

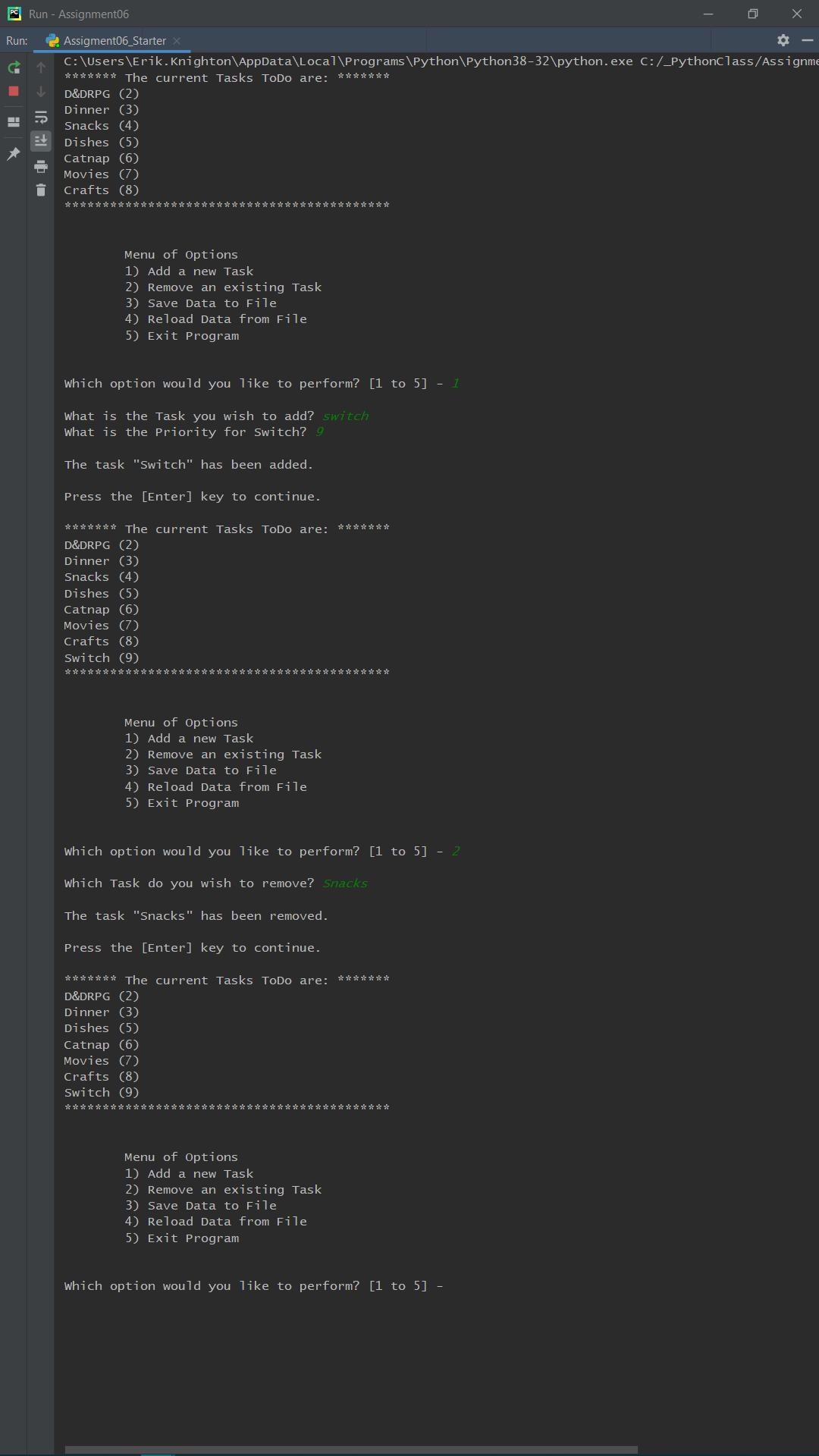
#### IO.input\_press\_to\_continue(strStatus)

And now, with that little solution in hand, I was able to remove all of the unnecessary print() lines from each of my own functions and move those strings to the return where 'Success' was a placeholder. I tested this first with adding to the list (Figure 7).



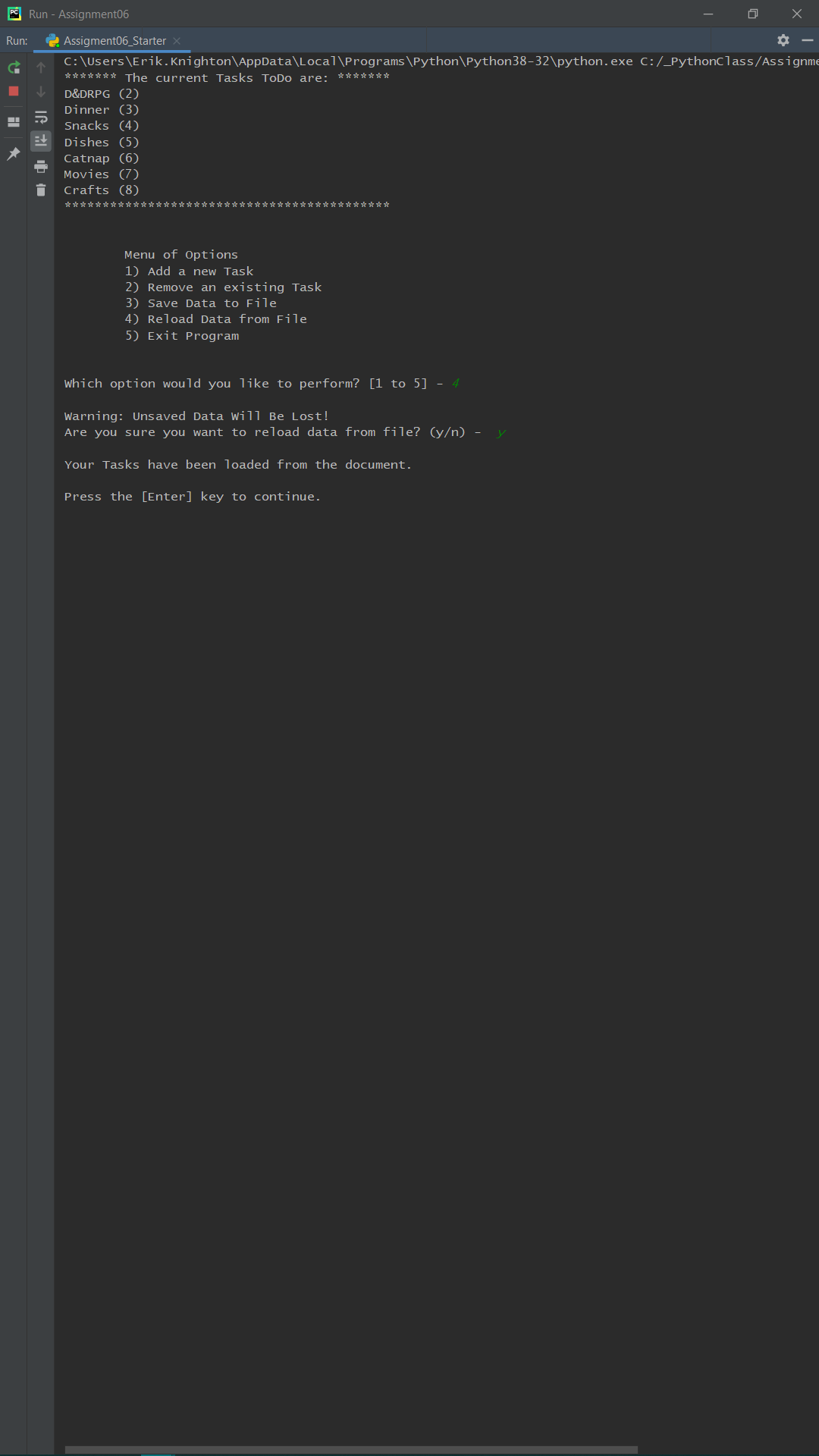
### Figure 7. Adding Task to List after Revisions

Then I copied it over to the functions which would remove a task from the list, and it worked flawlessly (Figure 8).



### Figure 8. Removing Task from List after Revisions

The last thing I did was look for where the strStatus variable was used in other parts of the main body, and I found it on the lines which reread the file, and so added another message that would print here (Figure 9).



With all of that was resolved, I have finished updating this document so I can resubmit my assignment, and then will update my GitHub repository as well. I am glad I caught this problem before my submission was reviewed, and I hope that these amendments are correct. I stated before that I made numerous changes without the program crashing, and because nothing seemed amiss here, I truly value any constructive criticism.

# Summary

Saying I learned a lot is an understatement. Overall, I felt like this assignment went smoothly, but I did fail in my order of operations. Typically I have plenty of time on Thursdays or Fridays to read and work through the process of taking notes, pondering my approach, and then watching the video lecture. As is typical, life happens, and I was not able to come at this week's module in my usual way. In a way I am grateful, as I might have just bungled my way through without learning from my mistakes, namely nesting functions and mishandling global variables. It is my wish going forward that I am able to step back, take some time to recoup, and come back at the work fresh, because I know that is how I work best in other aspects. Making it a habit to not simply brute-force my way through my schoolwork has prevented me from fatigue in the humanities, and why should it be any different here in computer science? Revelations like that are invaluable.

Revelations about how I botched or ignored outright another part of the existing code are likewise incomprehensibly worthwhile. It was also reassuring that I am now invested enough in Python, after so short a time, that I was able to work through the problem in my waking hours without sitting at my desk, without typing away at my computer. I resolve to be more thoughtful and methodical in the future.