Chani Orr

8/16/21

Foundation of Programming: Python

Assignment06

Functions

Purpose of The Lab

This lab intends to teach more in depth about functions and how to create classes and functions of your own. We also practiced using the Pycharm debugging tool. We then created a GitHub webpage. Additionally, we answered the following questions: What is a *function*?; What are *parameters*?; What are *arguments*?; What is the difference between *parameters* and *arguments*?; What are *return values*?; What is the difference between a *global* and a *local variable*?; How do you use *functions* to organize your code?; What is the difference between a *function* and a *class*?; How do *functions* help you program using the *“Separations of Concerns" pattern*?; How are the *debugging tools* used in PyCharm?; What is a GitHub *webpage*?

Getting Started (Project, Script, & Pseudocode)

Since this assignment is predicated heavily on the code from the previous assignment (Assignment05), I first made some suggested changes to that code based on the feedback from the class T.A. Though we won’t be rewriting this code per se, the idea is to transform it into something versatile with the use of functions and a heavy focus on separating the Concerns. So, like a pupa to a butterfly, we now liquify this code into a gross, sentient jelly and reform it for the same purpose with a different form.

Beginning with the Assignment06\_Starter that was given, I wrote my pseudocode into that very starter. Because I am easily confused, I felt compelled to compile the parts of the code into a check-list and color code each part a quick reference later. Thus, my pseudocode ended up looking very reflective of the final code rather than a vague algorithm.  
I must apologize ahead of time for the kaleidoscopic color palette you are about to experience (warning: dim your screen now).

**Pseudocode for Assignment 06** 

| **Class (Processor)**   | Functions |  | | --- | --- | | Read Data From A File(file\_name, list\_of\_rows), return list\_of\_rows | **✔** | | Add Data To The Master List(task, priority, list\_of\_rows), return list\_of\_rows  Add global ‘task’ and ‘priority’ to the global ‘dicRow’  Add dicRow to the list\_of\_rows | **✔** | | Remove Data From The List(task, list\_of\_rows), return list\_of\_rows | **✔** | | Write Data to The ToDoFile file(file\_name, list\_of\_rows), return list\_of\_rows | **✔** |   Class (IO)   | Functions |  | | --- | --- | | Print The Menu() | **✔** | | Input Menu Choice(): the the users menu choice and, return: string choice | **✔** | | Print Current Tasks In List(list\_of\_rows) , return: nothing | **✔** | | Input Yes or No Choice(message), return: return str(input(message)).strip().lower() | **✔** | | Input Script Pause - Press To Continue(optional message), return: nothing | **✔** | | input New Task and Priority(), return: task, priority  Assign data to the global ‘task’ and ‘priority’? | **✔** | | Input A Task To Remove(): return: task | **✔** |     #-------------------------------------- Main Body of Script ---------------------------------------- #    When the program starts, Load data from ToDoFile.txt  While(True)  Print the To-Do list  Print the Menu  Get menu choice from user  “Continue” to show the menu  1) If User wants to Add a new Task:  Get users new Task and Priority  Make a dictionary and Add user input to the Master List  Script pause  “Continue” to menu    2) If User wants to Remove and existing:  Show current task list  Ask what task they want to remove  Ask if they’re sure  If yes,  Search list for task and Remove task  Script pause  “Continue” to menu  3) If User wants to Save Data to a File:  strChoice = IO.input\_yes\_no\_choice("Save this data to file? (y/n) - ")  if strChoice.lower() == "y":  Write Data to The ToDoFile  Script pause  else:  Script pause("Save Cancelled!")  “Continue” to menu    4) If User wants to Reload Data from File:  print("Warning: Unsaved Data Will Be Lost!")  strChoice = IO.input\_yes\_no\_choice("Are you sure? (y/n) - ")  if strChoice.lower() == 'y':  Read Data From A File  Script pause  else:  Script pause("File Reload Cancelled!")  “Continue” to menu  5) If User wants to Exit Program  print("Goodbye!")  Break |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |

Add Code To The Script

Data: Constants and Variables

First and foremost, I do not understand why the list of declared Variables and Constants does not match the code used in the main body. I changed each variable to match my assumption of where it was being used in the script.

| **Original:**  strFileName = "ToDoFile.txt" # The name of the data file  objFile = None # An object that represents a file  dicRow = #{"priority": priority.strip(), "task": task.strip()}  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 | **Changed to:**  file\_name  objFile  dicRow  list\_of\_rows  menuChoice and choice  task  priority  strStatus |
| --- | --- |

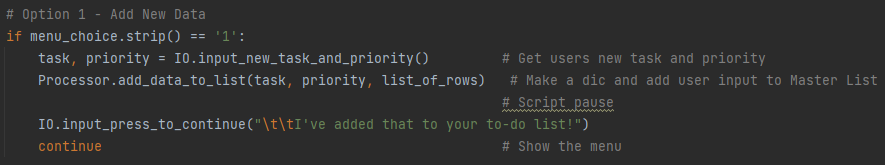
Processing and Presentation  
Next I went down the list in my pseudocode and checked off any functions already provided for me. As I finished writing each individual Function, I also checked it off the list. I often found that I was asking a function to do more than its share of the work, and had to separate actions out across the appropriate functions, but eventually I had all the working parts. Next it was time to plug them into their appropriate spots.   
I noted that the Processor functions operated unseen, and always returned a value, where the functions for presentation (“IO”) could return a value but sometimes did not return anything at all and instead printed out a message to the user.

Main Body of the Code

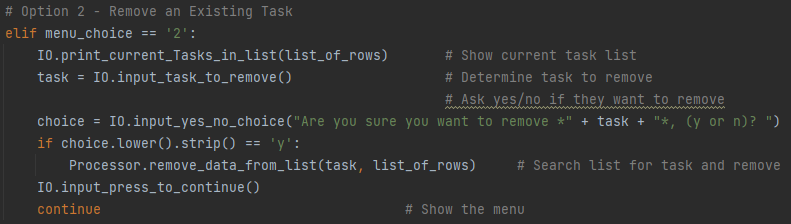
| (Image 1.0 - Beginning of Main Body of code & Main Loop) | When the program begins, the first line of code reads data from the chosen file (file\_name) and packs that data into a dictionary. Then that dictionary is added to a list previously designated in the Data section to be a constant. This process is not displayed to the user yet. In the very next bit of the program,a familiar, overarching While loop |
| --- | --- |

begins, at which point the list is printed out in its current form at the apex of the loop. The next function prints out the menu. The function after invites the user to input a numeric choice from the menu and assigns that choice to a variable.

If the user chooses Option 2, this portion of the code would take two values of input from the user and assign those values to the key/variables “task” and “priority”, then assign those to global variables of the same name. Those values are then converted into a dictionary and appended to a list with the next function. A script pause marks the end of the If statement and Continue brings the user back to the loop apex.

(Image 2.0 - Code for Option 1, Assignment 06)

Option 2 was perhaps the more difficult piece to code. If Option 2 is chosen, the first function prints the list of tasks as they currently read. The next function asks the user which task they would like to remove and assigns that answer to the local value “task”. This is used later in the If statement. The third function asks the user if they are certain they want to delete the item, at which point if they wish to continue, the fourth function is called to remove the named “task” from the greater list. Another script pause instructs the user to ‘press enter to continue’, and Continue brings the user back to the loop apex.

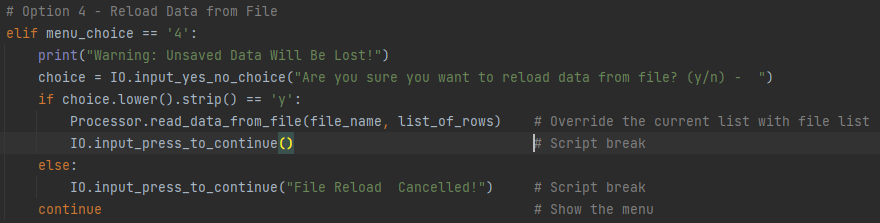


(Image 3.0 - Code for Option 2, Assignment 06)

| In Option 3, the same yes/no function asks the user right away if they are certain of their choice. If yes, the condition of the If statement is fulfilled and the next function writes the current list to the same data file as | (Image 4.0 - Code for Option 3, Assignment 06) |
| --- | --- |

was read from previously. The script pause function this time come with the message “To-Do list has been saved”, before leaping to the Continue command.  
If the first function returns anything besides ‘y’, the else clause is run and, after a script pause, Continue brings the user back to the loop apex.

Option 4 again employs the yes/no function right out front. If ‘y’, it reloads the data from the file, wiping the current list and restoring it to the condition under which it was last saved. A script pause function follows, then jumps to Continue. If no, the script pause includes a message - “File Reload Cancelled!” - before Continue brings the user back to the loop apex.



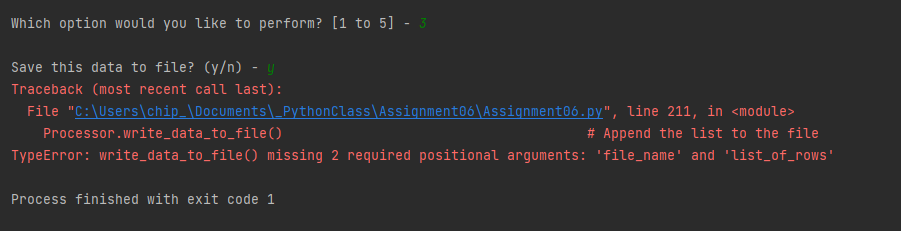
(Image 5.0 - Code for Option 4, Assignment 06)

| This would  annoying to  around, | (Image 6.0 - Code for Option 5, Ass. 06) | be very  try to read  would it not? |
| --- | --- | --- |

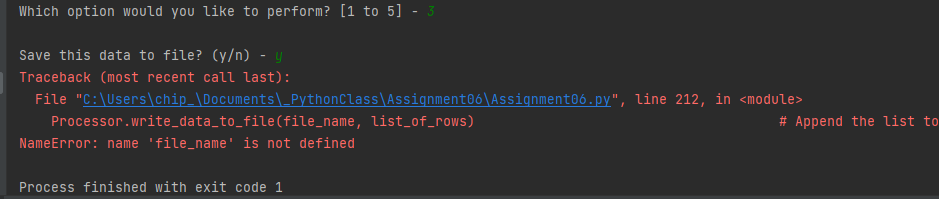
The final stage of the greater While loop uses none of the new functions we created in this module, but breaks the larger loop and exits the program.

Run the Script

I thought I was starting to understand this code, so I was thrown when I attempted option 3 and got a TypeError. I filled in the missing arguments and ran it again, confident this time that I had figured it out. Then, a NameError appeared. It was only after some time that I discovered that, while I had declared my intention to change my Constants, there was one I had missed.



(Image 7.0 - TypeError)



(Image 7.1 - NameError)  


(Image 7.2 - The constant that was almost forgotten)

Conclusion

In conclusion, this module allowed us to test our understanding of *classes*, *functions* and all of their parts, the Separation of Concerns, the ideas of Abstraction and Encapsulation, *global* and *local variables*, and other tools that PyCharm gives us access to. In addition, we built a web page through GitHub.

| What is a function?: | A function is a block of premade code that runs when it is called and returns a function if one is specified. By itself, a function does nothing - it must be given variables through its parameters. |
| --- | --- |
| What are *parameters*?: | Function parameters are the names listed in the function definition. |
| What are *arguments*?: | Function arguments are the real values received by the function when it is invoked. |
| What is the difference between *parameters* and *arguments*?: | Parameters are passed during the definition of function while Arguments are passed during the function call. |
| What are *return values*?: | The return statement is a special statement that sends the function's result back to the caller. A return statement consists of the return keyword followed by an optional return value. |
| What is the difference between a *global* and a *local variable*?: | A global variable is accessible throughout the entire program, while a local variable is one that is only accessible to the current scope, such as temporary variables used in a single function definition. |
| How do you use *functions* to organize your code?: | Functions use abstraction to shorthand the code desired rather than writing it out in its entirety every time that particular bit of code is desired. |
| What is the difference between a *function* and a *class*?: | A class is a definition of an Object, while a function is merely a piece of code. To Sum - Functions do specific things but classes are specific things. |
| How do *functions* help you program using the *“Separations of Concerns" pattern*?: | Functions always begin with a def keyword that hints at what those functions are used for. If the class they are assigned to can be defined by whether its primary use is to process or to present, then it can be used at a glance to help decide where these concerns belong. |
| How are the *debugging tools* used in PyCharm?: | First set the breakpoints. Then the debugging tools walk through the code you have isolated and perform each isolated action, including allowing you to see the current value of a variable. |
| What is a GitHub *webpage*?: | GitHub Pages is a static site hosting service that takes files straight from a repository on GitHub, optionally runs the files through a build process, and publishes a website. |