Functions

# Introduction

For the sixth assignment, the concept of functions was explored. **Functions** are a convenient way to divide code into useful blocks, allowing code to be ordered for separation of concerns, made more readable and reused. They are defined in the code before they are called leading to their statements to be executed. Their unique purpose can be defined in the docstrings that follow the functions definition. Docstrings act as function document headers that can be called in PyCharm through ctrl + q commands. Functions served a greater purpose for this assignment in allowing the coder to work in piece meal while editing and modifying an existing code provided. To assist in the debugging of the code with its enhanced layering, the use of breakpoints and the walking through code feature of PyCharm allowed me to address issues one by one until the program was fully operational. Finally, I built on my experience of uploading to a Github account by creating a Github webpage.

# Arguments, Parameters and Variables

## Positional versus Named Arguments

**Parameters** are inputs for functions that pass values into them for processing. Values passed into parameters are called **arguments** and are not to be used interchangeably with parameters. There are no limits to how many parameters can be included. Common naming conventions for parameters include not having a prefix and utilizing snake casing. A parameter can be called through its position, name, or as a default. **Default parameters** are set within the function. When the function is called, the function is forced to use the default parameter. **Named arguments** explicitly call for the name of the parameter within the function’s inputs. **Positional arguments** call values explicitly based of their position within the function’s input parentheses.

## Global vs Local Variables

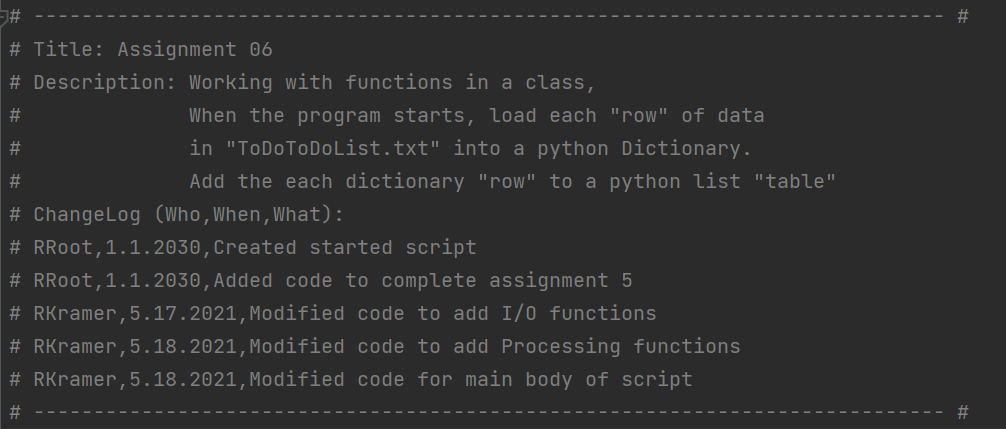
**Global variables** are those that are declared in the “main body” of the script and can be used anywhere in the script. **Local variables** are declared within the function and considered local to the containing function. Any code inside that same function can see the local variable as its inside its scope. However, any code outside of the function can not see the local variable and thus can not use it within it for inputs or processing. For global variables used within functions, it is wise to use the keyword global to avoid the local variable shadowing the global variable whenever a value is assigned to a variable with the same name. When a variable is read within a function with the same name as a global variable, the shadowing does not happen automatically.

# Classes and Functions

**Classes** group functions, variables and constants. In this way, the design principle of Separation of Concerns is applied to distinguish which functions are part of which process in a script. In this way, a class can be created for displaying while another class can be created for processing.

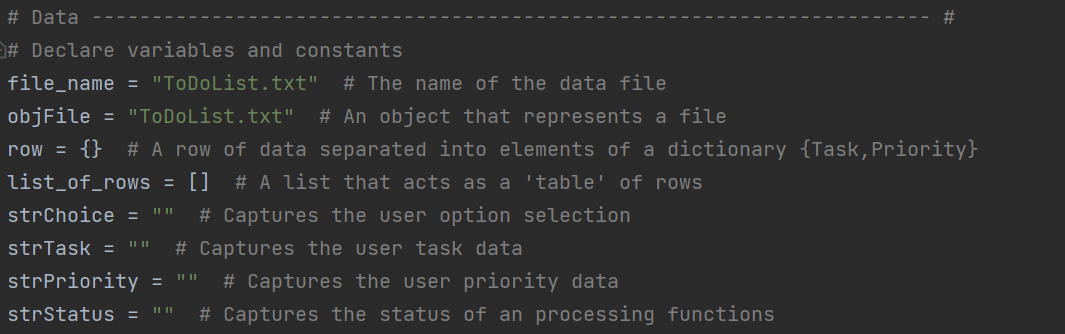
# Code Outlining

I began my script file with the script header (Figure 1). The original code was provided by Professor Root so he was credited with creating the script. To alter the program for this assignment, the first major change was adding to the input / output functions within the io Class. The next version of edits led to modifications of the processing functions within the processor class. The final version implemented the functions within the main body of script so that the adding lines, removing lines, saving to the file and reloading the file options were executable.



**Figure 1. Script Header**

The first concern addressed was the data (Figure 2) where the variable and constants for the dictionary and table were declared. Modifications were made to the original script file in the form of renaming the .txt file to the name for the file I created for assignment 05. In addition, it was noted that changes had to be made to my original code from Assignment 05 with regards to the row and list\_of\_row variable names. This was done through searching through the document and replacing the appropriate names.

**Figure 2. Data Section**

The processing section (Figure 3) began with establishing the processor class. The read\_data\_from\_file function already existed yet the add\_data\_to\_list, remove\_data\_from\_list and write\_data\_to\_file functions were modified with code copied from my past assignment 05. I added docustrings to distinguish each function’s purpose, parameters and returns. These parameters and returns were then incorporated into the code.

# 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, 'Success'  
  
 @staticmethod  
 def add\_data\_to\_list(task, priority, list\_of\_rows):  
 *""" Adds data to a list of dictionary rows* ***:param*** *task: (string) with name of task:* ***:param*** *priority: (string) with priority level:* ***:param*** *list\_of\_rows: (list) you want filled with file data:* ***:return****: (list) of dictionary rows  
 """* row = {"Task": str(task).strip(), "Priority": str(priority).strip()}  
 list\_of\_rows.append(row)  
 return list\_of\_rows, 'Success'  
  
 @staticmethod  
 def remove\_data\_from\_list(task\_to\_remove, list\_of\_rows):  
 *""" Removes task data and associated priority from list of dictionary rows* ***:param*** *task\_to\_remove: (string) with name of task to be removed:* ***:param*** *list\_of\_rows: (list) you want filled with file data:* ***:return****: (list) of dictionary rows  
 """* for row in list\_of\_rows:  
 if row["Task"].lower() == task\_to\_remove.lower():  
 list\_of\_rows.remove(row)  
 #print("row removed")  
 return list\_of\_rows, 'Success'  
  
 @staticmethod  
 def write\_data\_to\_file(file\_name, list\_of\_rows):  
 *""" Writes task data from list of rows to file* ***:param*** *file\_name: (string) with name of file being overwritten:* ***:param*** *list\_of\_rows: (list) you read from to fill file:* ***:return****: (list) of dictionary rows  
 """* file = open(file\_name, 'w')  
 for row in list\_of\_rows:  
 file.write(row["Task"] + "," + row["Priority"] + "\n")  
 file.close()  
 return list\_of\_rows, 'Success'

**Figure 3. Processing Section**

The presentation of the data (Figure 4) was taken into account with the class IO. Along with the functions that printed the task menu, the current menu of tasks, asked for yes or no responses and asked for the user to press the enter button to proceed, three new functions were created for asking for new data inputs, which data to remove and what the current tasks are. The modification process followed the same course as the processing function by reusing code from assignment 05. The appropriate parameters in terms of the list\_of\_rows for the function for printing the current tasks and none for the adding and deleting functions were added. Returns were created for the new input Tasks and Priority as well as the Task to be deleted. These inputs are used in the processing class above which need not be after the input output class.

# 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 To Do 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.')  
  
 @staticmethod  
 def input\_new\_task\_and\_priority():  
 *""" Asks for new task and task's priority* ***:return****: (strings) of task and priority  
 """* task = str(input("Task:")).strip()  
 priority = str(input("Priority:")).strip()  
 return task, priority  
  
 @staticmethod  
 def input\_task\_to\_remove():  
 *""" Asks for task to remove* ***:return****: (strings) of task to remove  
 """* task\_to\_remove = str(input("Which task would you like to remove:")).strip()  
 print()  
 return task\_to\_remove

**Figure 4. Input / Output Separation of Concern**

The processing and input and output functions were combined and implemented in the main body of script. The read\_data\_from\_file function starts the main body by reading the file. The user is then shown and asked for information through several functions from the IO class. Once the user decides the process they want the script to conduct, the code shifts to the if-elseif functions that determine which function to conduct. After each option is completed, the elseif statement is completed with a continue statement to return the user to the option menu except after the fifth option is selected which terminates the program.

# -------------------------------------------------#  
*# Main Body of Script ------------------------------------------------------ #  
  
# Step 1 - When the program starts, Load data from ToDoList.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* strChoice = IO.input\_menu\_choice() *# Get menu option  
   
 # Step 4 - Process user's menu choice* **if** strChoice.strip() == **'1'**: *# Add a new Task* task, priority = IO.input\_new\_task\_and\_priority()  
 Processor.add\_data\_to\_list(task, priority, list\_of\_rows)  
 IO.input\_press\_to\_continue(strStatus)  
 **continue** *# to show the menu* **elif** strChoice == **'2'**: *# Remove an existing Task* task\_to\_remove = IO.input\_task\_to\_remove()  
 Processor.remove\_data\_from\_list(task\_to\_remove, list\_of\_rows)  
 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"**:  
 Processor.write\_data\_to\_file(file\_name, list\_of\_rows)  
 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'**:  
 Processor.read\_data\_from\_file(file\_name, list\_of\_rows)  
 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*

**Figure 5. Main Body of Script**

# Program Test

C:\Python\python.exe C:/\_PythonClass/Assignment06/Assigment06\_Starter.py

\*\*\*\*\*\*\* The current Tasks To Do are: \*\*\*\*\*\*\*

Brush Teeth (High)

Laundry (High)

Sweep Floor (Low)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

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

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

Task:Wash Car

Priority:Low

Press the [Enter] key to continue.

\*\*\*\*\*\*\* The current Tasks To Do are: \*\*\*\*\*\*\*

Brush Teeth (High)

Laundry (High)

Sweep Floor (Low)

Wash Car (Low)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

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

Which option would you like to perform? [1 to 5] - 2

Which task would you like to remove:Brush Teeth

Press the [Enter] key to continue.

\*\*\*\*\*\*\* The current Tasks To Do are: \*\*\*\*\*\*\*

Laundry (High)

Sweep Floor (Low)

Wash Car (Low)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

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

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

Save this data to file? (y/n) - n

Save Cancelled!

Press the [Enter] key to continue.

\*\*\*\*\*\*\* The current Tasks To Do are: \*\*\*\*\*\*\*

Laundry (High)

Sweep Floor (Low)

Wash Car (Low)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

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

Which option would you like to perform? [1 to 5] - 4

Warning: Unsaved Data Will Be Lost!

Are you sure you want to reload data from file? (y/n) - y

Press the [Enter] key to continue.

\*\*\*\*\*\*\* The current Tasks To Do are: \*\*\*\*\*\*\*

Brush Teeth (High)

Laundry (High)

Sweep Floor (Low)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

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

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

Goodbye!

Process finished with exit code 0

**Figure 6. Output from running Assignment06\_Starter.py in pyCharm**

Microsoft Windows [Version 10.0.19041.985]

(c) Microsoft Corporation. All rights reserved.

C:\Users\robsk>CD C:\\_PythonClass\Assignment06\

C:\\_PythonClass\Assignment06>Python Assignment06\_Starter.py

\*\*\*\*\*\*\* The current Tasks To Do are: \*\*\*\*\*\*\*

Brush Teeth (High)

Laundry (High)

Sweep Floor (Low)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

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

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

Task:Change Oil

Priority:Medium

Press the [Enter] key to continue.

\*\*\*\*\*\*\* The current Tasks To Do are: \*\*\*\*\*\*\*

Brush Teeth (High)

Laundry (High)

Sweep Floor (Low)

Change Oil (Medium)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

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

Which option would you like to perform? [1 to 5] - 2

Which task would you like to remove:Brush Teeth

Press the [Enter] key to continue.

\*\*\*\*\*\*\* The current Tasks To Do are: \*\*\*\*\*\*\*

Laundry (High)

Sweep Floor (Low)

Change Oil (Medium)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

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

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

Save this data to file? (y/n) - y

Press the [Enter] key to continue.

\*\*\*\*\*\*\* The current Tasks To Do are: \*\*\*\*\*\*\*

Laundry (High)

Sweep Floor (Low)

Change Oil (Medium)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

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

Which option would you like to perform? [1 to 5] - 4

Warning: Unsaved Data Will Be Lost!

Are you sure you want to reload data from file? (y/n) -

File Reload Cancelled!

Press the [Enter] key to continue.

\*\*\*\*\*\*\* The current Tasks To Do are: \*\*\*\*\*\*\*

Laundry (High)

Sweep Floor (Low)

Change Oil (Medium)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

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

Which option would you like to perform? [1 to 5] - 4

Warning: Unsaved Data Will Be Lost!

Are you sure you want to reload data from file? (y/n) - y

Press the [Enter] key to continue.

\*\*\*\*\*\*\* The current Tasks To Do are: \*\*\*\*\*\*\*

Laundry (High)

Sweep Floor (Low)

Change Oil (Medium)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Menu of Options

1) Add a new Task

2) Remove an existing Task

3) Save Data to File

4) Reload Data from File

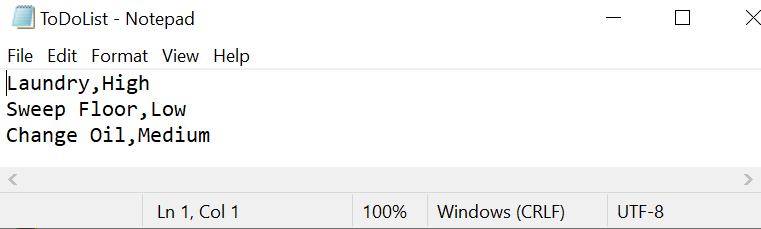
5) Exit Program

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

Goodbye!

C:\\_PythonClass\Assignment06>

**Figure 7. Output from running Assignment06\_Starter in the Command Prompt**



**Figure 8. Evidence of the text file’s save**

C:\\_PythonClass\Assignment05\ The program test was run through Pycharm (Figure 8) and through the command prompt (Figure 9). They both proved to be successful as the file was read and the program could display data, add to the list table, delete from the list table, save to the file and exit the program (Figure 9).

# Summary

This assignment was a timely exercise in how to edit another’s code. Whereas before we were asked to mostly originate our own code, this assignment allowed the programmer to edit another’s code. This is akin to what can be expected in the developing world where one must be able to read, edit and incorporate code to improve its performance or function. The use of functions better enables this by not needing changes to be done in multiple places. Functions allow changes to cascade through the rest of the code once they are performed within the function or class itself. Finally, the knowledge of how to debug one’s own code allows me to identify my mistakes sequentially and how to better locate them.