Thomas Farmer

2-22-2022

Foundations Of Programming

Assignment06

https://github.com/MtnWolf82/IntroToProg-Python-Mod06

Working with Functions

INTRODUCTION

This week we learned about fun with functions. Once again we continue to build off of prior lessons by creating functions that pack away bits of our script for use whereever we deem necessary. By the time we actually write the main body script, we're simply referencing the functions we defined prior. Once again, we'll present the user with several different menu options for interacting with the program. The user will again be able to add or delete tasks, save the data to an external Text document, and exit the program.

THE PROCESS

1. Open a new PyCharm file and enter your program's header comments. An example of this assignment's comments can be seen below. Note: A starter script was provided in this instance.

```
# ------------------------------------------------------------------------------ #
# Title: Assignment 06
# Description: Working with functions in a class,
#              When the program starts, load each "row" of data
#              in "ToDoList.txt" into a python Dictionary.
#              Add each dictionary "row" to a python list "table"
# ChangeLog (Who,When,What):
# RRoot,1.1.2030,Created started script
# TFarmer,2.20.2022,Modified code to complete assignment 06
# TFarmer,2.21.2022,Continued to modify code to complete assignment 06
# TFarmer,2.22.2022,Finished modifying code to complete assignment 06
# ------------------------------------------------------------------------------ #
```

2. First, let's declare our variables and constants. Enter the code shown below.

```
# Data ------------------------------------------------------------------------ #
# Declare variables and constants
file_name_str = "ToDoList.txt"  # The name of the data file
file_obj = None  # An object that represents a file
row_dic = {}  # A row of data separated into elements of a dictionary {Task,Priority}
table_lst = [] # A list that acts as a 'table' of rows
list_of_rows = []  # A list that acts as a list of rows
choice_str = ""  # Captures the user option selection
```

3. We'll start out be creating functions for our 'Processor' class. The first function – read_data_from_file – will be used to pull data from our “ToDoList” text file and display its contents in the form of a list to the user. Enter the code shown below.

```
# 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
```

4. Function – add_data_to_list – will add the “task” and “priority” variables, as defined in a later function, to our table. Enter the code shown below.

```
    @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 name of priority:
        :param list_of_rows: (list) you want filled with file data:
        :return: (list) of dictionary rows
        """
        dicRow = {"Task": task, "Priority": priority}
        list_of_rows.append(dicRow)
        print("Task added.")
        print()
        return list_of_rows
```

5. Function – remove_data_from_list – will remove the “task” variable, as defined in a later function, from our table. Enter the code shown below.

```
    @staticmethod
    def remove_data_from_list(task, list_of_rows):
        """ Removes data from a list of dictionary rows

        :param task: (string) with name of task:
        :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.lower():
                list_of_rows.remove(row)
                print("Task removed.")
                print()
            else:
                print("Task not found.")
        return list_of_rows
```

6. The final processing function – write_data_to_file – will write the updated list in the program's memory to our “ToDoList” text file. Enter the code shown below.

```
    @staticmethod
    def write_data_to_file(file_name, list_of_rows):
        """ Writes data from a list of dictionary rows to a File

        :param file_name: (string) with name of file:
        :param list_of_rows: (list) you want filled with file data:
        :return: (list) of dictionary rows
        """
        objFile = open(file_name, "w")
        for row in list_of_rows:
            objFile.write(row["Task"] + "," + row["Priority"] + "\n")
        objFile.close()
        return list_of_rows
```

7. We will now create functions for our 'I/O' class. The first function – output_menu_tasks – simply provides a menu of options for the user to select from in order to interface with the program. Enter the code shown below.

```
# Presentation (Input/Output)  -------------------------------------------- #
class IO:
    """ Performs Input and Output tasks """

    @staticmethod
    def output_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) Exit Program
        """)
        print()  # Add an extra line for looks
```

8. Function – input_menu_choice – gathers the user's menu choice, to be used in a later function. Enter the code shown below.

```
    @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 4] - ")).strip()
        print()  # Add an extra line for looks
        return choice
```

9. Function – output_current_tasks_in_list – will print out the tasks currently contained in the memory's list. This will reflect and additions or removals made since initially running the program. Enter the code below.

```
    @staticmethod
    def output_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 are: *********")
        for row in list_of_rows:
            print(row["Task"] + " (" + row["Priority"] + ")")
        print("******************************************")
        print()  # Add an extra line for looks
```

10. Function – input_new_task_and_priority – will collect the “task” and “priority” inputs from the user and outputs them for use in the 'add_data_to_list' function previously created. Enter the code show below.

```
    @staticmethod
    def input_new_task_and_priority():
        """  Gets task and priority values to be added to the list

        :return: (string, string) with task and priority
        """
        task = input("What is the task? ").strip()
        priority = input("What is the priority? [High|Low] ").strip()
        print()
        return task, priority
```

11. Function – input_task_to_remove – will collect the “task” to be removed from the list from the user and outputs it for use in the 'remove_data_from_list' function prevously created. Enter the code shown below.

```
    @staticmethod
    def input_task_to_remove():
        """  Gets the task name to be removed from the list

        :return: (string) with task
        """
        task = input("Which task would you like removed? ")
        print()
        return task
```

12. With the functions defined, let's wrap up by putting them all to the test. We will now write the main body of the program's script. As most of the following following coding is simply recalling the entered functions, we'll address multiple at a time. Let's start out with presenting our list data and user interface. Enter the code shown below.
    a. Step 1 of the code below, we're recalling the function defined in step 3 to load data from our “ToDoList” text file.
    b. Step 2 utilizes the functions defined in steps 7 & 9 to load the current data in the table and then display the user menu.

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

# Step 1 - When the program starts, Load data from ToDoFile.txt.
Processor.read_data_from_file( file_name=file_name_str, list_of_rows=table_lst)  # read file data

# Step 2 - Display a menu of choices to the user
while (True):
    # Step 3 Show current data
    IO.output_current_tasks_in_list(list_of_rows=table_lst)  # Show current data in the list/table
    IO.output_menu_tasks()  # Shows menu
    choice_str = IO.input_menu_choice()  # Get menu option
```

13. Now we'll address each of the options presented to the user. Step 4 has assigned functions to each available user option. Enter the code shown below.

```
    # Step 4 - Process user's menu choice
    if choice_str.strip() == '1':  # Add a new Task
        task, priority = IO.input_new_task_and_priority()
        table_lst = Processor.add_data_to_list(task=task, priority=priority, list_of_rows=table_lst)
        continue  # to show the menu

    elif choice_str == '2':  # Remove an existing Task
        task = IO.input_task_to_remove()
        table_lst = Processor.remove_data_from_list(task, list_of_rows=table_lst)
        continue  # to show the menu

    elif choice_str == '3':  # Save Data to File
        table_lst = Processor .write_data_to_file(file_name=file_name_str, list_of_rows=table_lst)
        print("Data Saved!")
        print()
        continue  # to show the menu

    elif choice_str == '4':  # Exit Program
        print("Goodbye!")
        break  # by exiting loop
```

SUMMARY

This assignment was particularly difficult for me. Coming into someone else's code and modifying it threw me for a bit of a loop. Several trips to the book and reference videos were required. Even then, it didn't really click for me what I was missing until I watched the assignment help videos. I had been reading too deep into each function's purpose, and therefore overcomplicating things. Taking the function comments in a much more literal sense eased my stress quite a bit. Just as in prior assignments, practice and patience are key.