IMarshall

02/21/2023

IT FDN 110 A

Assignment 06

IMarshallUW/IntroToProg-Python-Mod06 (github.com)

Assignment 06 – Functions

Introduction

In this paper we will build upon the knowledge we gained in module 05 where we simulated picking up where someone left on a project and building on the start of a code to build something functional to serve out purposes. This time however, we did so using user-defined functions.

Writing Code

To begin this assignment we took the starter code we were given and read through it to see what had been accomplished and what still needed to be done. The code provided defined our starter code declared our global variables for the code, a process code to read existing data in a file when the program started, 3 input/output (IO) tasks, and the actual code that would run the code using all the functions once they were built out (figures 1-4). Leaving us with the task of writing the processor tasks to add and remove data to memory, as well as write the data in memory to a file. We would also need to write the IO code to get the user input on what would be added or removed.

```python
# Data ---------------------------------------------------------------------- #
# Declare variables and constants
file_name_str = "ToDoFile.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
choice_str = ""  # Captures the user option selection


# 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
```

Figure 1: Starter file; global variables and read data processor function

```python
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

    @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
```

Figure 2: Starter file; input/output functions

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

Figure 3: Starter file; input/output functions cont.

```python
# 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

    # 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=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!")
        continue  # to show the menu

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

Figure 4: Starter file; program script

The first bit of code that we added was to define the process class function that would add data to a list held in memory. Initially we reworked the add data code from the module 05 assignment. Which would take the inputs from the user in addition to adding the input data. Later it was realized that taking the input data would be handled by a separate IO task so that part of the code was commented out. Leaving us with only the append statement to add a new row to our list in memory, a print statement to give the user feedback, and a return statement to update the definition of the current “list_of_rows” variable (figure5).

```python
@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 to add more data to:
    :return: (list) of dictionary rows
    '''
    row = {'Task': str(task).strip(), 'Priority': str(priority).strip()}
    # TODO: Add Code Here!
    '''strTask = str(input(task)).strip()
    strPrior = str(input(priority)).strip()
    row_dic = {'Task': strTask, 'Priority': strPrior}
    '''  # Overcomplicated, above will be handled in IO class function
    list_of_rows.append(row)
    print('Data was added to list')
    return list_of_rows
```

Figure 5: process class add data to memory function

The next code we had to complete was the process to remove data from the memory. We tried to recycle from the previous assignment, but it would return multiple instances of our print statement to the user. One for every row in memory. So instead we reworked some code presented by Professor Root in last weeks assignment. This code creates a boolean flag and sets it to false. Then, starting with row zero, we use a for loop to check if the task in the list equals the task entered by the user and delete it. This instance will change our flag from false to true to print one statement versus the other, eliminating the multiple statements being printed (figure 6).

```python
    check = False  # Creating a boolean Flag
    intRowNumber = 0
    for row in list_of_rows:
        if row['Task'].lower() == task.lower():
            del list_of_rows[intRowNumber]
            check = True
        intRowNumber += 1
        # end if
    # end for loop
    # Step 5b - Update user on the status
    if (check == True):
        print("The task was removed.")
    else:
        print("Sorry, the task could not be found.")
    return list_of_rows
```

Figure 6: process class, remove data from memory function

The last process code we needed to write was to write the memory to a external file. This was accomplished by recycling our module 05 code and changing the variables to match the established ones in this program (figure 7).

```
@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
    '''
    # TODO: Add Code Here
    file_obj = open(file_name, 'w')
    for row_dic in list_of_rows:
        file_obj.write(str(row_dic['Task']) + ',' + str(row_dic['Priority']) + '\n')
    file_obj.close()
    print(' Data was saved to file', file_name)
    return list_of_rows
```

Figure 7: process class, write memory to file function

Our next two bits of code we needed to add were our missing IO functions, gathering the user inputs to add and remove data via the processes we just wrote. Both were written using recycled code from our previous assignment. To save space and make it easier for someone to read when looking at my code our code we edited how we limit the selections available to the users for their priority designation. In the initial code we created a separate while loop and would check if the string entered was 1-5 and break the loop is it was, or print an error statement if it was outside of those parameters. This time we accomplished the same result by using an in statement to designate the entries that would be considered valid (figure 8).

```python
    @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
        '''
        # TODO: Add Code Here!
        task = str(input('Please enter a Task: ').strip())
        while True:
            priority = str(input('Please enter the task\'s priority from 1-5, 1 = low & 5 = high: ').strip().lower())
            if priority in ['1', '2', '3', '4', '5']:
                break
            else:
                print('Please enter a valid priority (1-5)')
                '''Holds user to defined priority range'''
        return task, priority

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

        :return: (string) with task
        '''
        # TODO: Add Code Here!
        task = str(input('Please enter a task to remove: ').strip())
        return task
```

Figure 8: IO class, input task/priority to add & input task to remove functions

Now that all the missing code has been defined the program can run as intended. If we reference figure 5 we can see that the program is set up to run the same way as assignment 05, but now instead of the code being written out you call the functions. The benefit of this being that as we write longer and more complicated code that need to perform similar or the same task in different parts, we can save time by calling the function that have been defined instead of rewriting the code entirely. The additional benefit of splitting the actions that we defined as a process from an IO task is that they can be mixed and matched to provide better versatility and the code becomes more complicated.

Conclusion

In this paper we will built upon the knowledge we gained in module 05 to add onto a preexisting code that could read data from an external file to store in local memory, add/remove data from the memory, and write the data currently in the memory to the same external file using user-defined functions. Below are screen captures of how the program runs in both PyCharm and Command Prompt (figures 9-14).

```
C:\Users\nezum\AppData\Local\Programs\Python\Python310\python.exe C:\_PythonClass\Assigntment06\Assigment06.py
******* The current tasks ToDo are: *******
sweep (2)
sleep (5)
Feed Children (5)
*****************************************


        Menu of Options
        1) Add a new Task
        2) Remove an existing Task
        3) Save Data to File
        4) Exit Program


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

Please enter a Task: Mop
Please enter the task's priority from 1-5, 1 = low & 5 = high: 2
Data was added to list
******* The current tasks ToDo are: *******
sweep (2)
sleep (5)
Feed Children (5)
Mop (2)
*****************************************


        Menu of Options
        1) Add a new Task
        2) Remove an existing Task
        3) Save Data to File
        4) Exit Program


Which option would you like to perform? [1 to 4] - 2
```

Figure 9: Program run in PyCharm

```
Please enter a task to remove: feed children
The task was removed.
******* The current tasks ToDo are: *******
sweep (2)
sleep (5)
Mop (2)
****************************************


        Menu of Options
        1) Add a new Task
        2) Remove an existing Task
        3) Save Data to File
        4) Exit Program


Which option would you like to perform? [1 to 4] - 3

 Data was saved to file ToDoFile.txt
Data Saved!
******* The current tasks ToDo are: *******
sweep (2)
sleep (5)
Mop (2)
****************************************


        Menu of Options
        1) Add a new Task
        2) Remove an existing Task
        3) Save Data to File
        4) Exit Program
```

Figure 10: Program run in PyCharm cont.

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

Goodbye!

Process finished with exit code 0
```

Figure 11: Program run in PyCharm cont.

```
Microsoft Windows [Version 10.0.19044.2604]
(c) Microsoft Corporation. All rights reserved.

C:\Users\nezum>cd C:\_PythonClass\Assigntment06\

C:\_PythonClass\Assigntment06>Python C:\_PythonClass\Assigntment06\Assigment06.py
******* The current tasks ToDo are: *******
sweep (2)
sleep (5)
mop (4)
*****************************************


        Menu of Options
        1) Add a new Task
        2) Remove an existing Task
        3) Save Data to File
        4) Exit Program


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

Please enter a Task: Feed Children
Please enter the task's priority from 1-5, 1 = low & 5 = high: 5
Data was added to list
******* The current tasks ToDo are: *******
sweep (2)
sleep (5)
mop (4)
Feed Children (5)
*****************************************


        Menu of Options
        1) Add a new Task
        2) Remove an existing Task
        3) Save Data to File
        4) Exit Program


Which option would you like to perform? [1 to 4] - 2

Please enter a task to remove: mop
The task was removed.
```

Figure 12: Program run in Command Prompt

```
******* The current tasks ToDo are: *******
sweep (2)
sleep (5)
Feed Children (5)
*****************************************


        Menu of Options
        1) Add a new Task
        2) Remove an existing Task
        3) Save Data to File
        4) Exit Program


Which option would you like to perform? [1 to 4] - 3

 Data was saved to file ToDoFile.txt
Data Saved!
******* The current tasks ToDo are: *******
sweep (2)
sleep (5)
Feed Children (5)
*****************************************


        Menu of Options
        1) Add a new Task
        2) Remove an existing Task
        3) Save Data to File
        4) Exit Program


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

Goodbye!

C:\_PythonClass\Assigntment06>
```

Figure 13: Program run in Command Prompt cont

Command Prompt

```
Microsoft Windows [Version 10.0.19044.2486]
(c) Microsoft Corporation. All rights reserved.

C:\Users\nezum>cd C:\_PythonClass\Assignment05

C:\_PythonClass\Assignment05>Python C:\_PythonClass\Assignment05\Assigment05.py
Task:                                   Priority:
laundry                                 5
dishes                                  4
Clean dog area                          3
^Data currently in file

    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] - 2

Type in a task and priority from 1 (low) to 5 (high)
Enter a task: mop
Enter a priority: 8
Please enter a value between 1 and 5
Enter a priority: 1

    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] - 1

The current data is:
Task:                                   Priority:
laundry                                 5
dishes                                  4
Clean dog area                          3
mop                                     1
Press ENTER to continue

    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

What task would you like to remove? clean dog ArEa
data not found in row
data not found in row
row removed
```

Figure 14: Program running in Command Prompt 1/2

Command Prompt

```
    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] - 1

The current data is:
Task:                                   Priority:
laundry                                 5
dishes                                  4
mop                                     1
Press ENTER to continue

    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] - 4

 Data was saved to file

    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] - 5


           Thank you for using our services.
           Have a nice day


C:\_PythonClass\Assignment05>
```

Figure 15: Program running in Command Prompt 2/2