Zach DeVries
Feb 22, 2023
IT FDN 110 A
Assignment 06

Lists and Dictionaries

Introduction:
As part of this week's module, I worked with lists and dictionaries set within formulas to create a to Do List program similar to the one developed in Assignment 5.

Program:
In this week's lessons, the project roadmap was already laid out in the starter file I was provided. I looked over the header information, read through the file and analyzed the the various sections of code (Data, Processing, I/O, Main). After reviewing the doc initially, I then took note of the variables that had been declared and ran the program to take note of the current state. Nexty I set off to add my content.

Step 1
I began by looking through the import data section of code. I set up a file in my working directory and reviewed the processor.read_data_from_file function. After a few tweaks, I got that portion of the code working and then I set out to work on the next phase.

Ste 2-3
Steps 2 and 3 were already done at the point when I received the code. I looked over the main body structure, reviewed the functions, and tested the code to make sure it worked smoothly before moving on to step 4.

Step 4.1
Step 4.1 was to add data to the to Do List. The main body calls to the functions were already in place, so I analyzed the structure, made sure I understood how the arguments were being used and then proceeded up into the function definition portion of the project to add my code. I started with the I/O function as it was called in the main body first. I knew from last weeks assignment that I would need to capture user inputs for both Task and Priority and then arrange them into a list that that could be passed into my next function in the processing section, which would append the list to the existing set of data. That is what I did and then I jumped to the aforementioned function to work out the processing logic.

Figure 1: I/O.input_new_task_and_priority

```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
    """

    str_task = input('Enter a Task: ')  # Collect user input
    str_priority = input('What is the priority? [high/low] ')   # Collect user input
    lst_user_input = [str_task, str_priority.upper()]  # Store input to list
    print()  # Add an extra line for looks

    return lst_user_input
```

Knowing the output of the I/O function to be a list with the Task and Priority data, I set off on finishing out the add_data_to_list function. The portion of the function that pulls data from the list passed in from the I/O function and places it into a dictionary row was already set up, so I added the simple step of appending the row to our list_of_rows variable which would be passed out of the function and saved in the main body of the program.

Figure 2: processing.add_data_to_list

```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()}
    list_of_rows.append(row)

    return list_of_rows
```

Step 4.2

For step 4.2, I followed the same methodology as in step 4 and reviewed the main body calls to the functions to generate an understanding of the implementation. Then I navigated up to the I/O portion of the code and set to work generating the code to capture user input about which Task they would like to remove. I created a local variable and generated the code to capture

user input like I did in assignment 05. Then I made sure to return the parameter out of the function to be used as the input in the processing function.

Figure 3: input_task_to_remove

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

    :return: (string) with task
    """
    str_task_remove = input('Enter a task to remove: ')  # Collect user input
    print()  # Add an extra line for looks
    return str_task_remove
```

I started off this function development by reviewing my code from assignment 5. I then set up a similar structure for each row I pulled out the data and assigned it to local variables. Then I ran a comparison between the task data in the list to the data entered by the user in the I/O function that was passed through to the processing function. Using the equality comparison inside the If function, I added in a line of code to remove the current row if the equality condition returns True. Finally, I added the return section to pass the updated list variable out of the function. I then ran the program to check my code and made a few small updates to get it working properly before moving on to the save data portion of the code.

Figure 4: remove_data_from_list

```
@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
    """

    bln_item_removed = False  # Declare a boolean flag
    for row in list_of_rows:  # Evaluate the list one row at a time
        str_task_compare, str_priority = dict(row).values()  # Extract the list data into variables
        if str_task_compare.lower().strip() == task.lower().strip():  # Compare user entered task to list data
            list_of_rows.remove(row)  # if the list data and user entered data match, remove the list row
            bln_item_removed = True
    return list_of_rows
```

Step 4.3

For step 4.3, I imported my code from assignment 05 and renamed the variables to fit the previously defined arguments. I then ran my code to test my implementation. After a not insignificant amount of time, I identified an issue in my save function to be improper indentation. After correcting the formatting, I ran the program again and confirmed everything worked as expected.

Figure 5: write_data_to_file

```
@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
    """
    file = open(file_name, "w")  # Open the txt file
    for line in list_of_rows:  # Evaluate the below action for each row
        file.write(line['Task'] + ' , ' + line['Priority'] + '\n')  # Write each row in the table to the file.
    file.close()  # Close the file
    return list_of_rows
```

Conclusion:
In this module, I edited an existing pycharm and worked with lists/dictionaries inside of functions. The files are saved on Github here: [zdevries123/IntroToProg-Python (github.com)]

Figure 6: Program Execution in PyCharm

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

Enter a Task: Take out the Trash
What is the priority? [high/low] high

******* The current tasks ToDo are: *******
Rake (High)
Water (HIGH)
Take out the Trash (HIGH)
*******************************************


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

Figure 7: Program Execution in Console

```
******* The current tasks ToDo are: *******
Rake (High)
Water (HIGH)
*****************************************


        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

Enter a task to remove: Rake

******* The current tasks ToDo are: *******
Water (HIGH)
*****************************************


        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 Saved!

******* The current tasks ToDo are: *******
Water (HIGH)
*****************************************
```