Meg E. Sumnicht
2022-FEB-22
IT FDN 110 A Wi 22: Foundations of Programming: Python
Assignment 06
https://github.com/RedHotMegma/IntroToProg-Python-Mod06

Functions

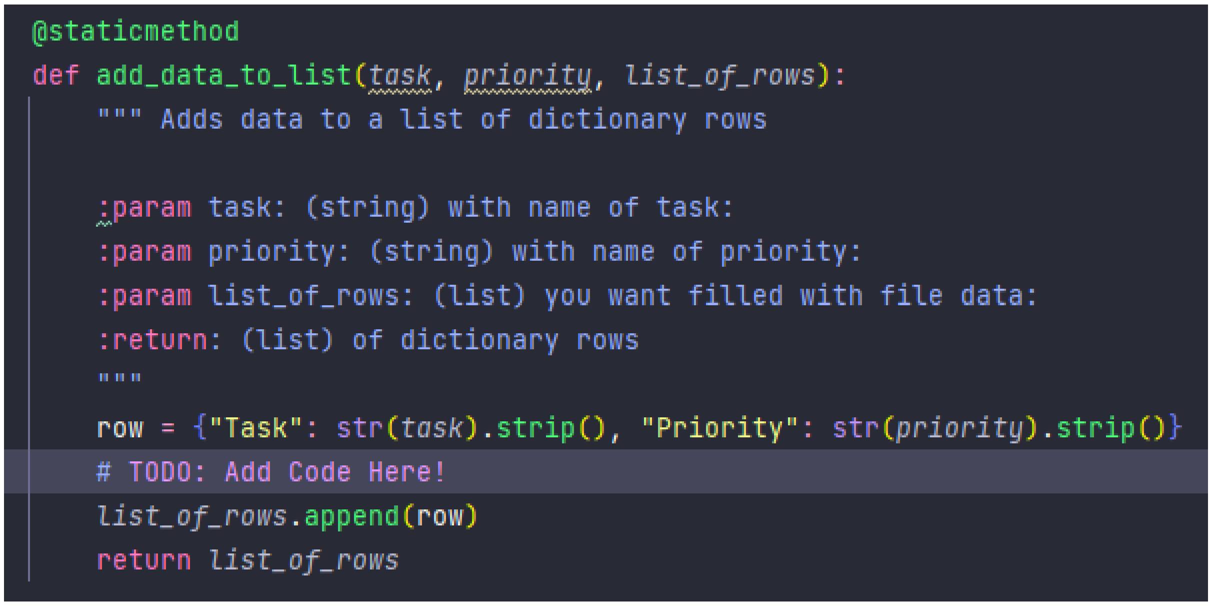
INTRODUCTION

The purpose of this exercise is to complete the creation of functions which helps to organize and simplify the code to generate and edit a to do list.

PROCESSING

add_data_to_list

The code for this function already included a variable *row* which defined a dictionary with a task and a priority. The inputs from the dictionary are entered parameters that will come from when the function is called in the main part of the code and will be pulled through the *input_new_task_and_priority* function. The code which needed to be added here was a simple list append function to append the dictionary *row* onto the *table_lst*.



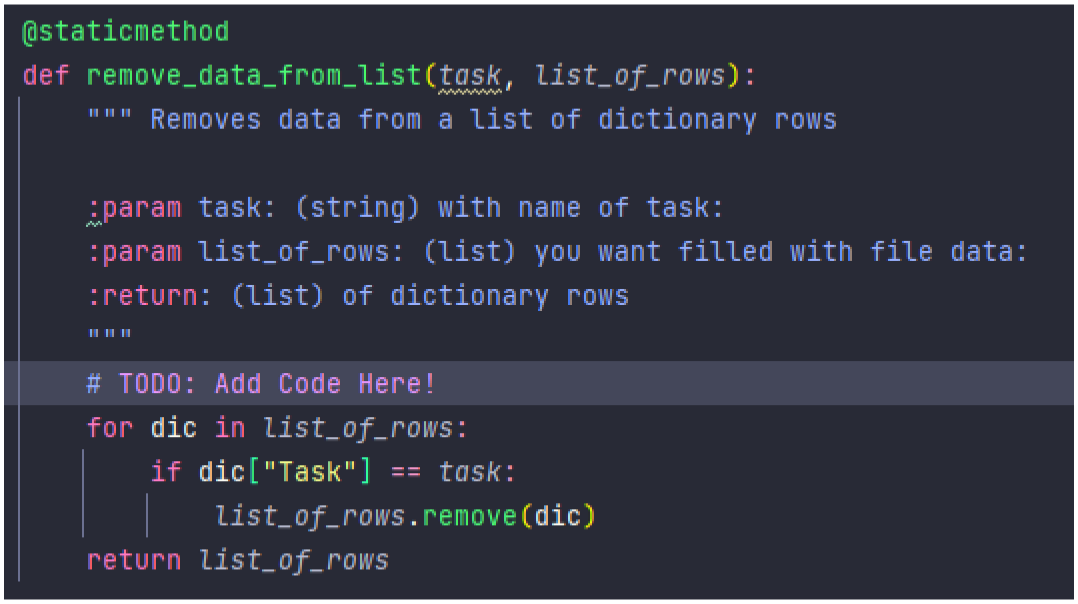
```
@staticmethod
def add_data_to_list(task, priority, list_of_rows):
    """ Adds data to a list of dictionary rows

    :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
    """
    row = {"Task": str(task).strip(), "Priority": str(priority).strip()}
    # TODO: Add Code Here!
    list_of_rows.append(row)
    return list_of_rows
```

FIGURE 1.1

remove_data_from_list

The purpose of this function is to find a task and its priority in the *table_lst* and remove it. This is accomplished through use of a for loop and a conditional. The function here works in tandem with another function *input_task_to_remove* which prompts the user to provide the task which they wish to remove.

The for loop is then steps through each dictionary line of the *list_of_rows* test to see if the "Task" equals the task which the user provided and if it does so then a list.remove is employed to remove that dictionary line from the *list_of_rows*. Once the loop has completed, the function then returns the amended *list_of_rows*.



```
@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
    """
    # TODO: Add Code Here!
    for dic in list_of_rows:
        if dic["Task"] == task:
            list_of_rows.remove(dic)
    return list_of_rows
```

FIGURE 1.2

write_data_to_file

This function opens the text file which is being used to store the To Do list. It then uses a for loop to step through each dictionary line of the list and write the task and its priority to a separate line. Once all of the lines have been written to the file, the loop ends and the file closes.

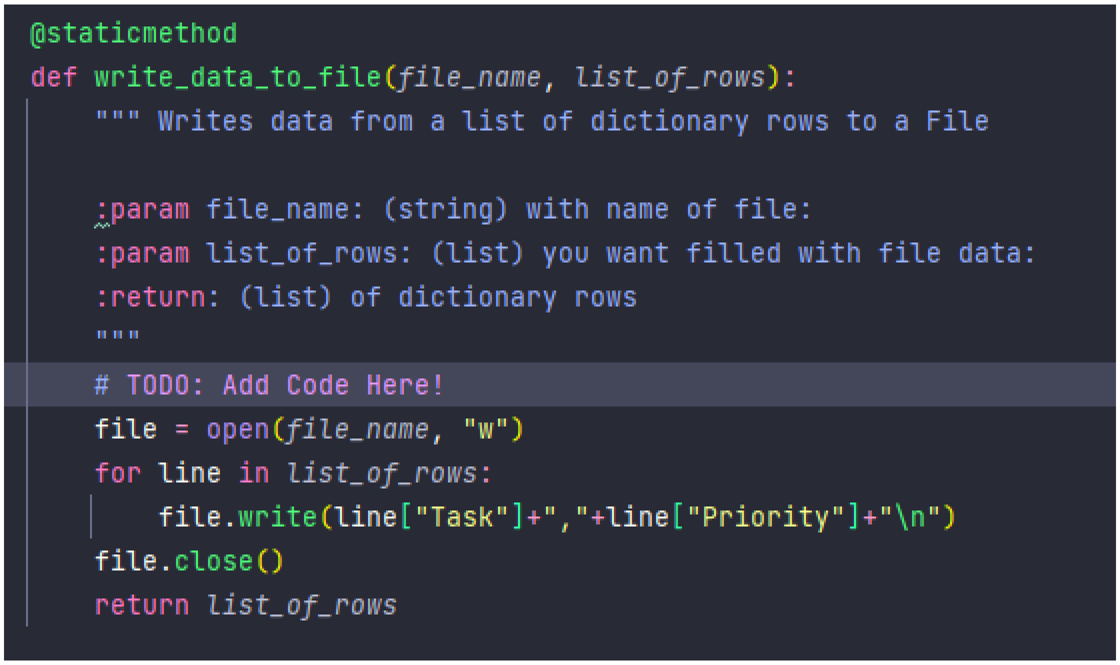

```python
@staticmethod
def write_data_to_file(file_name, list_of_rows):
    """ Writes data from a list of dictionary rows to a File

    :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 = open(file_name, "w")
    for line in list_of_rows:
        file.write(line["Task"]+","+line["Priority"]+"\n")
    file.close()
    return list_of_rows
```

FIGURE 1.3

PRESENTATION

input_new_task_and_priority

This is a fairly simple line of code which requests inputs from the user and then returns a tuple. This function is employed in tandem with the *add_data_to_list* function which takes the tuple which is returned and adds it to the list.

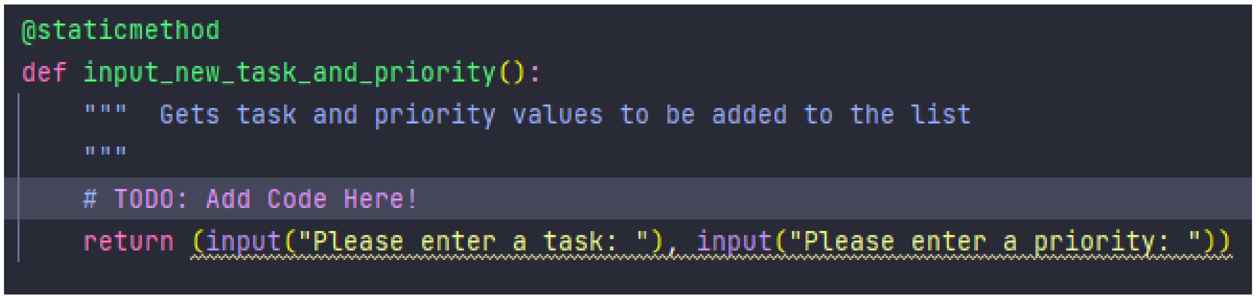
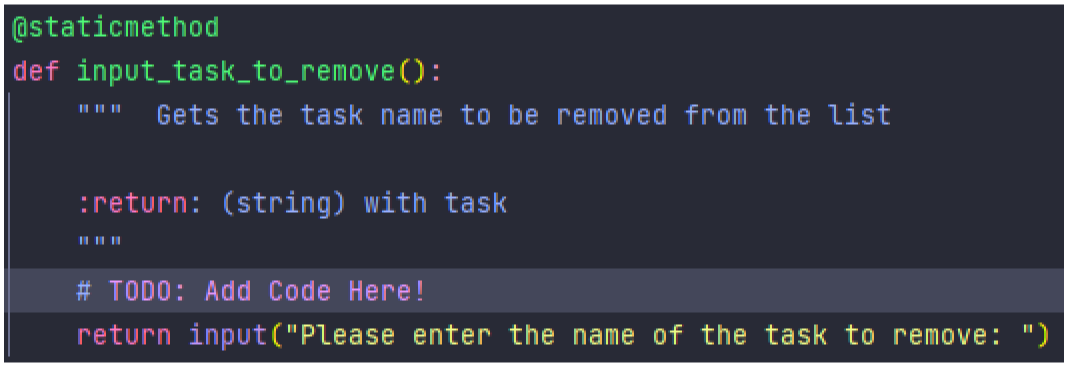

```python
@staticmethod
def input_new_task_and_priority():
    """  Gets task and priority values to be added to the list
    """
    # TODO: Add Code Here!
    return (input("Please enter a task: "), input("Please enter a priority: "))
```

FIGURE 2.1

input_task_to_remove

This function is very simple as it asks for a user to input the task from the list which they would like to remove and then returns it. This is used in tandem with the *remove_data_from_list* function which takes the string provided in this step and employs it to find the task which should be removed and then removes it.



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

    :return: (string) with task
    """
    # TODO: Add Code Here!
    return input("Please enter the name of the task to remove: ")
```

FIGURE 2.2

CONCLUSION

Functions provide a more streamlined and easier to read and edit code. By breaking the code into smaller chunks which then can be called upon in single lines, it is easier to identify problem areas when something goes wrong and it means that if a process is used in several places it only needs to be changed in one place and you can trust the code to flow into all the appropriate places.