Kathleen Wong

November 22, 2021

Intro to Programming

Assignment 06

https://kathleen101.github.io/IntroToProg-Python-Mod06/

Adding Functions to a Script

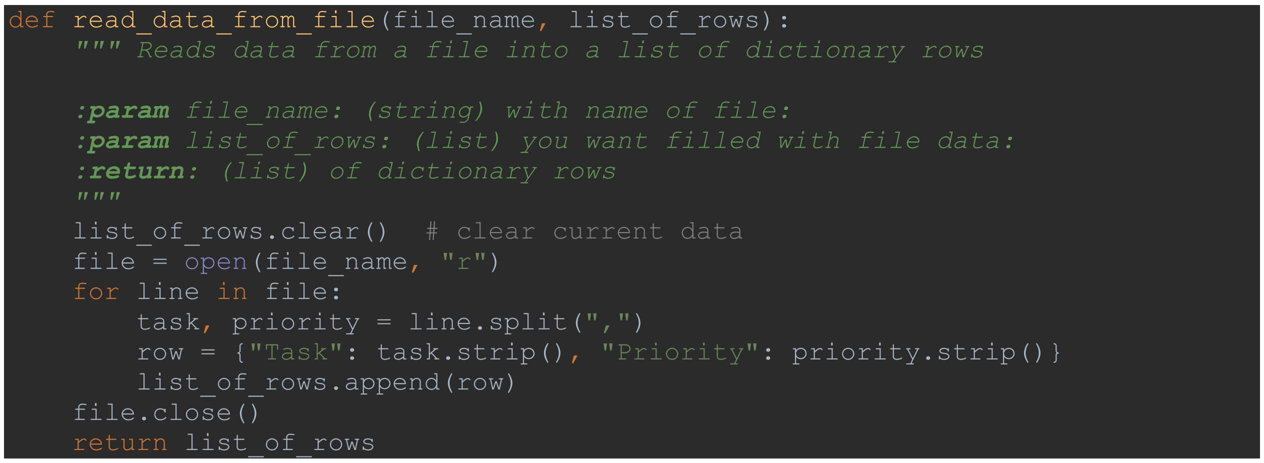
Introduction

This week, I learned about functions and how essential they are to programming. Functions are similar to simpler scripts. They perform a task and return control to my program (Python Programming Third Edition, External Site). However, rather than writing over and over again to do the same task in various portions of the script, functions allow programmers to simply write the task once as a function and when needed to execute the task, the programmer simply calls on the function again. In this paper, I will be discussing how I created my script, involving the functions requested by the class.

Assignment Background

Randall has provided an incomplete script, and it is my duty to update the script to complete the tasks requested. Thankfully, Randall has already provided the variables required to complete each task and the functions to edit. My job is to update the functions themselves and provide which functions are needed under each choice to execute the action. Randall provided the read_data_from_file function. Here the function clears any current data then opens the "ToDoFile.txt" file. Afterwards, the script splits each data in each row of data by a comma then sets the first data in the row as the task and the second data in the row as the priority. Then, the script appends the read data into the list_of_rows variable, which is simply a list filled with dictionary items.

Randall has also formatted the script to separate the functions by two classes: Processor and IO. This cleans up the script by each function's purpose on a higher level. So if I ever decide to call a function, I would need to format it as "Class"."Function." Please see this as an example under page 2, figure "Option Choice 1."



```python
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
```

Importing

I also decided to import the os library to pull in the directory of the text file. I do not seem to need this when testing within PyCharm as the script already knows the project and directory. However, when testing within Command Prompt, this does not apply. I need to specify which directory to allow them to know which text file exactly to read and write to.

```
import os
dir_path = os.path.dirname(os.path.realpath(__file__)) # pulling directory
file_name = os.path.join(dir_path, file_name) # connecting text file to
directory
```

Process User's Menu Choice

If the user inputs "1" as their choice, I need to add their task and priority to the list. Luckily with the list_of_rows provided by the read_data_from_file function, I do not need to create a list. Instead, I first update the input_new_task_and_priority function, which simply requests the task and priority from the user, both set as strings. Then, the function returns the task and priority, making the script capable of retrieving the results.

Next, I update the add_data_to_list function to set the user's data into a dictionary. The variable row_dic sets both "Task" and "Priority" as the keys while the user's input are set as the values and puts them both into a dictionary. It then appends the row_dic to list_of_rows. Afterwards, it prints out a note, saying the task has been added. The function then returns the list_of_rows.

Afterwards, I updated option 1 to set the task and priority to the input_new_task_and_priority function. Note that I formatted the function to IO.input_new_task_and_priority(). I did this because the function is under the IO class. I then set the table_lst variable to Processor.add_data_to_list(task, priority, table_lst).

The add_data_to_list function is underneath the Processor class, so I formatted the function as Processor.add_data_to_list. I then called for task, priority, and table_list into the function because the function calls for them as required information to continue. Both task and priority come from the input_new_task_and_priority function. The table_lst is created in the beginning of the script and is set to the results of the function.

Function input_new_task_and_priority

```
@staticmethod
def input_new_task_and_priority():
    """ Removes data from list of dictionary rows

    :param task: (string) data you want to add to file:
    :param priority: (string) you want to attach to task:
    :return: (string) of both datapoints
    """
    task = str(input("What task would you like to add? "))
    priority = str(input("What is the priority of this task? "))
    return task, priority
```

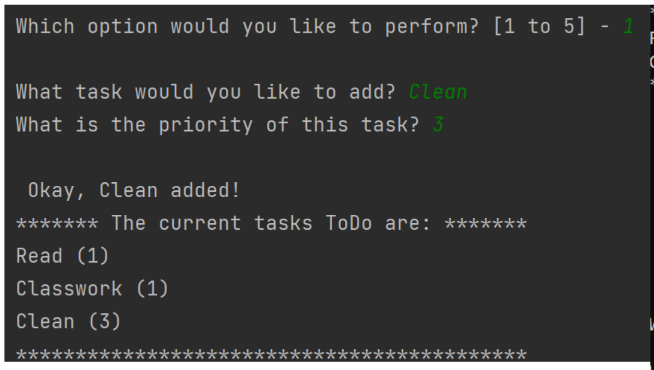
Function add_data_to_list

```
def add_data_to_list(task, priority, list_of_rows):
    """ Adds new data into list of dictionary rows

    :param task: (string) you want to add to list
    :param priority: (list) you want to attach to task
    :param list_of_rows: (list) you want filled with file data:
    :return: (list) of dictionary rows
    """
    row_dic = {"Task": task, "Priority": priority} # sets task and priority
into dictionary
    list_of_rows.append(row_dic) # appends task and priority to list_of_rows
    print("\n Okay,", task, "added!") # prints status
    return list_of_rows # returns list
```

Option Choice 1

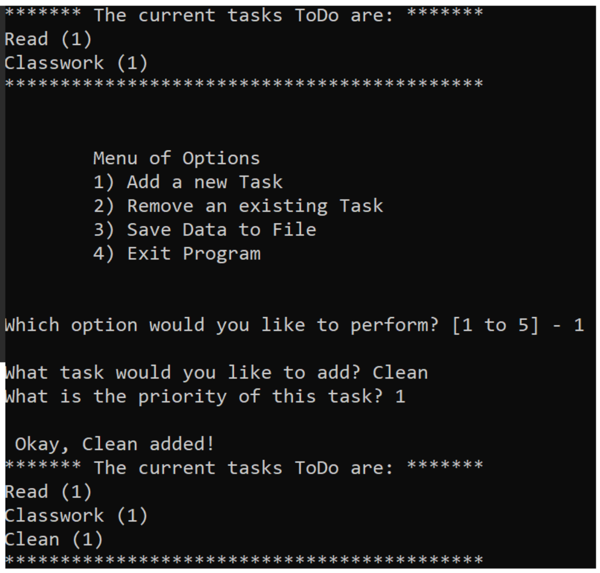
```
if choice_str.strip() == '1':  # Add a new Task
    print(table_lst)
    task, priority = IO.input_new_task_and_priority() # requests for data
from user
    table_lst = Processor.add_data_to_list(task, priority, table_lst) # adds
requested data into list
    continue  # to show the menu
```

Testing PyCharm



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

What task would you like to add? Clean
What is the priority of this task? 3

 Okay, Clean added!
******* The current tasks ToDo are: *******
Read (1)
Classwork (1)
Clean (3)
*******************************************
```

Testing Command Prompt



```
******* The current tasks ToDo are: *******
Read (1)
Classwork (1)
*******************************************


        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 5] - 1

What task would you like to add? Clean
What is the priority of this task? 1

 Okay, Clean added!
******* The current tasks ToDo are: *******
Read (1)
Classwork (1)
Clean (1)
*******************************************
```

Remove an Existing Task

If the user inputs "2" as their choice, I need to remove the task. Randall had set up two functions to complete these tasks: IO.input_task_to_remove() and Processor.remove_data_from_list().

The input_task_to_remove() function simply requests which task to remove. This function will return the task inputted. The input_task_to_remove() function actually removes the task. I first set the row_num variable to 0. I then pull out each task and priority from the dictionary's values and set them to the task_check and priority_check variables. Afterwards, I set up a loop, checking whether the requested task matches each row the dictionary and removing them row if it does match. After each check, the computer will add 1 to the row_num variable, so the script will move to the next row and remove the correct row if it does match.

Function input_task_to_remove

```
def input_task_to_remove():
    """ Removes data from list of dictionary rows

    :param task: (string) data you want to remove from file:
    :return: (string)
    """
    task = str(input("Which task would you like to remove? "))
    return task
```

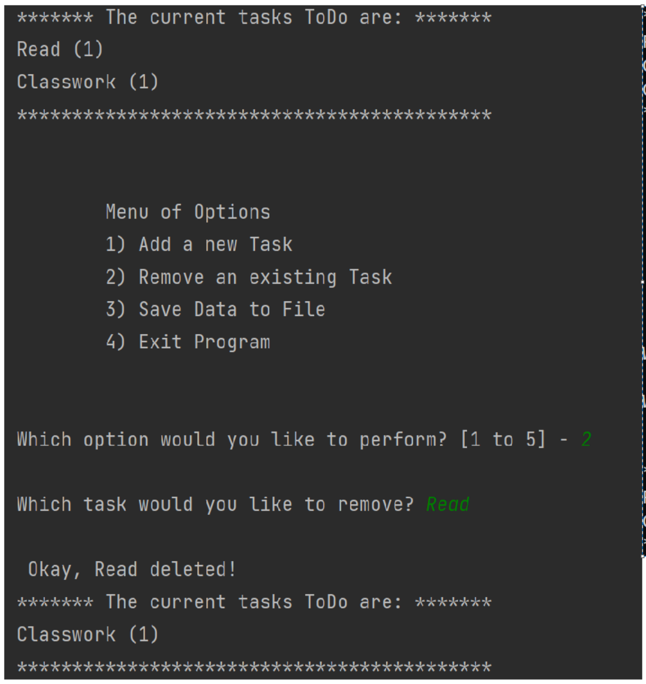
Function remove_data_from_list

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

    :param task: (string) you want to remove from list
    :param list_of_rows: (list) you want filled with file data:
    :return: (list) of dictionary rows
    """
    row_num = 0 # set row_num to 0
    for row in list_of_rows:
        task_check, priority_check = dict(row).values() # sets task_check and
priority_check to dictionary values
        if task.lower() == task_check.lower(): # checks task to each task in
dictionary and deletes row if equal
            del list_of_rows[row_num]
            row_num += 1
            print("\n Okay,", task, "deleted!")
        else:
            row_num += 1
    return list_of_rows
```

Option Choice 2

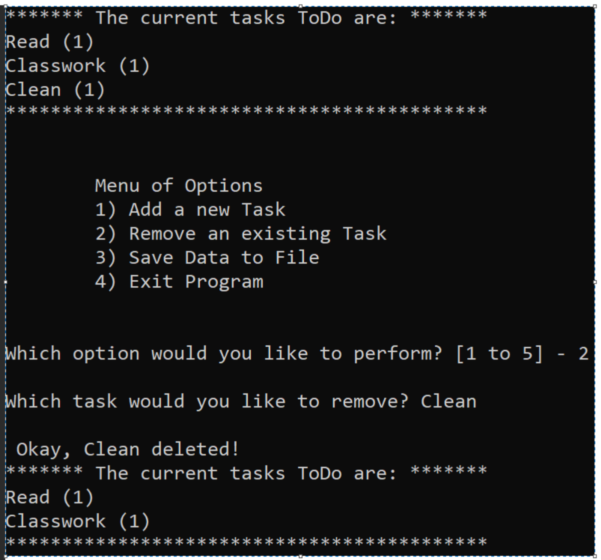
```
elif choice_str == '2':  # Remove an existing Task
    task = IO.input_task_to_remove() # requests for data from user
    table_lst = Processor.remove_data_from_list(task, table_lst) # removes
data from list if found
    continue  # to show the menu
```

Testing PyCharm



```
******* The current tasks ToDo are: *******
Read (1)
Classwork (1)
*******************************************


        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 5] - 2

Which task would you like to remove? Read

 Okay, Read deleted!
******* The current tasks ToDo are: *******
Classwork (1)
*******************************************
```

Testing Command Prompt



```
******* The current tasks ToDo are: *******
Read (1)
Classwork (1)
Clean (1)
*******************************************


        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 5] - 2

Which task would you like to remove? Clean

 Okay, Clean deleted!
******* The current tasks ToDo are: *******
Read (1)
Classwork (1)
*******************************************
```

Save Data to File

If the user inputs “3” as their choice, I need to save the results to the file.Randall had set up the Processor.write_data_to_file() function to complete this task. I updated the function to set file_obj to open the file and write into it. Afterwards, I set up a loop to go through each row in the list_of_rows and write each row of data as “Task,Priority.” Then, I close the file and return the list_of_rows.

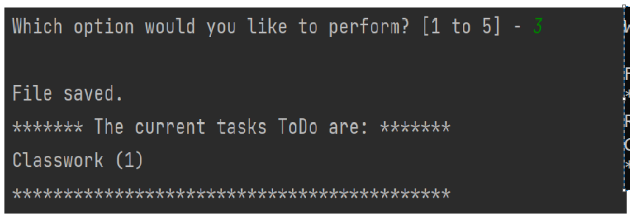
Function write_data_to_file

```
def write_data_to_file(file_name, list_of_rows):
    """ Removes data from 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
    """
    file_obj = open(file_name, "w") # opens file to write in
    for row_dic in list_of_rows:  # formats and writes data into file
        file_obj.write(row_dic["Task"] + "," + row_dic["Priority"] + "\n")
    print("File saved.") # prints status
    file_obj.close() #closes file
    return list_of_rows
```

Option Choice 3

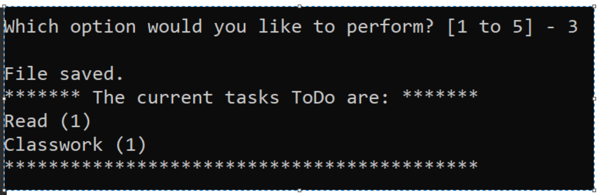
```
elif choice_str == '3':  # Save Data to File
    table_lst = Processor.write_data_to_file(file_name_str, table_lst) #
saves data to file
    continue  # to show the menu
```

Testing PyCharm



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


File saved.
******* The current tasks ToDo are: *******
Classwork (1)
*******************************************
```

Testing Command Prompt



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

File saved.
******* The current tasks ToDo are: *******
Read (1)
Classwork (1)
*******************************************
```

Conclusion

Functions allow programmers to simplify and clean up the script to avoid task repetition. After learning about functions, I updated Randall's script to use functions to execute the functions.