Felix Chen

2/21/23

IT FDN 110A

Assignment06

https://github.com/f3liz/IntroToProg-Python-Mod6/tree/main/docs

Creating a menu using functions

Functions

In this week's module we learned about functions which are a way to group multiple statements together. In order to use a function you first have to define it before you can use/call the function. Within functions we learned that you can pass through values into the function which are called parameters, which are officially called arguments but parameters work too.

```
fltV1 = None
fltV2 = None
fltSum = None
fltDif = None
fltProd = None
fltQuot = None

def math(fltValue_1, fltValue_2):
    fltSum = fltValue_1 + fltValue_2
    fltDif = fltValue_1 - fltValue_2
    fltProd = fltValue_1 * fltValue_2
    fltQuot = fltValue_1 / fltValue_2
    return fltSum, fltDif, fltProd, fltQuot

fltV1 = float(input("Number 1: "))
fltV2 = float(input("Number 2: "))

fltSum, fltDif, fltProd, fltQuot = math(fltV1, fltV2)

print('''
Sum is %.2f
Diff is %.2f
Prod is %.2f
Quot is %.2f''' % (fltSum, fltDif, fltProd, fltQuot))
```

Figure 1: Example of a function and arguments in PyCharm

Classes

We also were briefly introduced to the concept of classes this week. Classes are a way to group functions, kind of similar to how functions are to group statements.

```
class Processor:
    """  Performs Processing tasks """
```

```
class IO:
    """ Performs Input and Output tasks """
```

Figures 2 and 3: Example of classes in PyCharm

Steps I took

My first step for this assignment was to look over the starting script to see what I needed to add to it. Once I noted what parts of the script I had to add, I started by making a new function to write a new file if the user didn't have the file that was being loaded. This was similar to what I did last week.

```
    @staticmethod
    def create_new_file():
        objFile = open(file_name_str, "w")  # to create a new file if one isn't found
        objFile.close()
```

Figure 4: Function to create a new file

Now that I had created a function to make the file for the user if one isn't found for them, I added a try and except so the program would know if it needed to use this new function or not.

```
try:
    Processor.read_data_from_file(file_name=file_name_str, list_of_rows=table_lst)  # read file data
except:
    print('''
    ToDoList.exe file currently not found
    One has now been created for you
    ''')  # lets users know they don't have the file
    Processor.create_new_file()
```

Figure 5: My try and except code in PyCharm

Once I made my new function and added in my try and except in the beginning of the main body of my script I started to fill in the ToDos starting from the top. The first code I had to add was code to add data to the list and this was fairly simple. I used some of my old code from last week but had to change some things around to match the arguments.

```
    @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
        """
        # TODO: Add Code Here!
        dicRow = {"Task": str(task).strip(), "Priority": str(priority).strip()}
        list_of_rows += [dicRow] # will add the data to the list

        return list_of_rows
```

Figure 6: Add data to list code in PyCharm

The next code to write was code for removing data from a list. This was also fairly easy since I had written the code for last week's assignment and I had to just make it work for the function that I was going to put the code into.

```python
@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 row in list_of_rows:
        if row["Task"].lower() == task.lower():
            list_of_rows.remove(row)# removes the specified task
            print("Task has now been removed!")# only prints if the task is found and removed
            break
        else:
            print("Task not found.")# prints if task not found
    return list_of_rows
```

Figure 7: Remove data from list code in PyCharm

The next code to fill in was code to write the user data to the file. This was also almost the same as last week's code except there were just minor changes to make the code work for the function and its arguments.

```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!
    objFile = open(file_name, "w")
    for row in list_of_rows:
        objFile.write(str(row["Task"]) + "," + str(row["Priority"]) + "\n")# writes the new data to the file
    objFile.close()
    return list_of_rows
```

Figure 8: Write data to file code in PyCharm

Then there needed to be code added to the function that would store the users input for a task and its priority. I liked how I had written my code last week to prompt users to only input 1-5 so I used that code for this week's assignment as well because I felt like it was user friendly.

```
@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
    """
    print("Please enter a task and it's priority from 1 (high) to 5(low)")
    task = str(input("Name of task: "))
    while (True):
        priority = int(input("Priority of the task: "))
        if priority not in range(1, 6):  # if user input isn't 1-5 it'll ask them to choose 1-5
            print("Please choose from 1 being high priority to 5 being lowest priority")
        else:
            break
    priority = str(priority) # changes the int back to string
    return(task, priority)
    pass  # TODO: Add Code Here!
```

Figure 9: Example of function that stores user input in PyCharm

The last piece of code was to store the input of the user of what data they want removed. This function only stores which data the user wants to remove while the remove_data_from_list function is the one that actually removes it.

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

    :return: (string) with task
    """
    task = str(input("Name of the task you'd like removed: ")) # to store the name of task user wants to remove
    return(task)
    pass  # TODO: Add Code Here!
```

Figure 10: Example of function that stores user input on which task to remove in PyCharm

I also added an else statement at the end just to remind users to choose from options 1-4 if they had chosen an option outside of those.

```
else:
    print("Please choose from options 1-4!!!")
```

Figure 11: Else statement in PyCharm

```
    ToDoList.exe file currently not found
    One has now been created for you


******* The current tasks ToDo are: *******
*******************************************


        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 and it's priority from 1 (high) to 5(low)
Name of task: dishes
Priority of the task: 1

******* The current tasks ToDo are: *******
dishes (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 4] - 3

Data Saved!

******* The current tasks ToDo are: *******
dishes (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 4] - 4

Goodbye!
```

Figure 12: Script working in Terminal

```
******* The current tasks ToDo are: *******
dishes (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 4] - 3

Data Saved!

******* The current tasks ToDo are: *******
dishes (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 4] - 4

Goodbye!
```

Figure 13: Script working in PyCharm

```
dishes,1
```

Figure 14: Data saved to file from script

Summary

I enjoyed learning about classes and functions from this week's lecture as it helps me clean up the overall look of my script and keeps things neat and organized. It was a bit challenging at first dealing with functions and its arguments because I had to remember what was being passed through and how to pass those values through. I think it helped that the start script described what kind of code should be there and I just referenced my script from last week to input the right code for the right function.