Miguel Quintana
May 24, 2023
IT FDN 100 A
Assignment 06
https://github.com/miqo1385/ITFnd100-Mod006.git
https://miqo1385.github.io/ITFnd100-Mod006/

**Introduction:**

This week we will learn about functions and a little bit about classes. We will create a program similar to last week's assignment, but we will use functions instead of any other type of coding.

**Module 6 lecture (paper, book, week lecture)**

This week, we learned about Functions. By definition, functions are ways to group one or more statements. Functions sometimes use parameters, this is optional but a great tool. Parameters allow you to "pass" values into the function for processing, these parameters are also called "arguments", so arguments and parameters are basically the same, but in Python there is more use of the word parameters. It is important to mention that we can use variables as parameters.

Another important function tool is the "return" value. The return value makes the function work as an expression, which means that you can use the result of a function immediately without placing the result in a variable. Return values make the code easier to perform in three layers of concern: data, processing, and presentation. Another thing about return values is that you can return a single value or multiple values in one function. However, you need to bundle them into a collection and return them as a collection. Python packing and unpacking just work with tuples, this is a great tool for Python; other languages do not have this option.

Another concept related to functions is local and global variables. The variables that are declared inside the function are considered local and can not be accessed outside the function; instead, variables that are declared in the body are global and can be used anywhere in the script.

Above the function, we have classes. Classes are the way to group functions, variables, and constants. So we can say that functions group statements, and classes group functions, constants, and variables.

Functions help the concept of "separation of concern" because they organize a piece of code to perform specific tasks that are easy to find, and easy to fix and modify.

### Website review:

http://www.learnpython.org/en/Functions: This website goes through the definition of a function, how we write a function, and how we call a function. Is a short but very useful website.

### Video review:

https://youtu.be/qO4ZN5uZSVg: This video explains functions, explaining concepts like parameters, arguments, and global and local functions. As a resume, it explains the architecture of a function.

### Assignment 06:

The assignment for this week is about a menu of choices where the user can add a task, remove an existing task, save data to a file, and exit the program. As we can see, this is a very similar program to last week's assignment. The biggest difference is that this time we will use a pre-made, incomplete script that will use classes and functions to create the program. However, many functions just have a pseudocode that will give you a hint about what should be done with that particular function. The program will be divided into four parts: the data part, the processing, the presentation, and the body of the script.

On the presentation, the first function that is incomplete is add_data_to_list:

```
def add_data_to_list(task, priority, list_of_rows):

```

The only code given was was the structure of the dictionary

```
row = {"Task": str(task).strip(), "Priority": str(priority).strip()}

```

The part that we added to complete this function was to append the dictionary to a list of rows, and we returned the list_of_rows:

```
list_of_rows.append(row)

return list_of_rows
```

The next function that was missing code was def remove_data_from_list. What we did basically was to iterate over the list of rows with the "if" conditional statement with the established if row["Task"].lower() == task.lower(): We will remove that task from the list, and we will return the list of rows after we remove the task.

```
for row in list_of_rows:

   if row["Task"].lower() == task.lower():

       list_of_rows.remove(row)



return list_of_rows
```

Another function related to removing a task that I created was related to what happens if the task received from the user is not found on the list. If that is the case, it will show a message that says "Task not Found":

```
def task_not_found(task, list_of_rows):

   for row in list_of_rows:

       if row["Task"].lower() != task.lower():

           print("Task Not Found")

           return list_of_rows
```

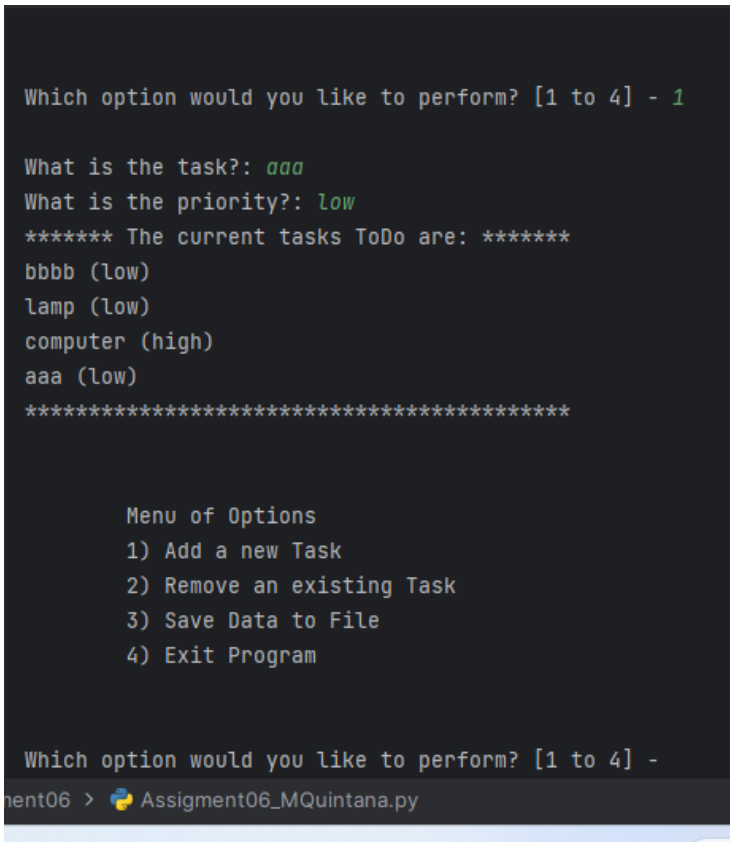
Finally, I complete the def write_data_to_file. We create a variable file that will hold the open function that will open the file and let you write on it:

```
file = open(file_name, "w")

for row in list_of_rows:

   file.write(row["Task"] + "," + row["Priority"] + "\n")

file.close()

return list_of_rows
```

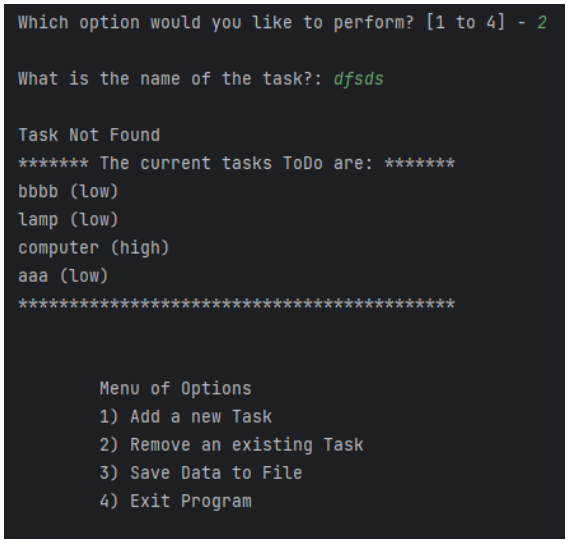
To get input from the user, we will use another function in a different class, This class is about presentation; it will ask the user for a task and priority, and the def write_data_to_file() will process these inputs and add them to the list of rows.

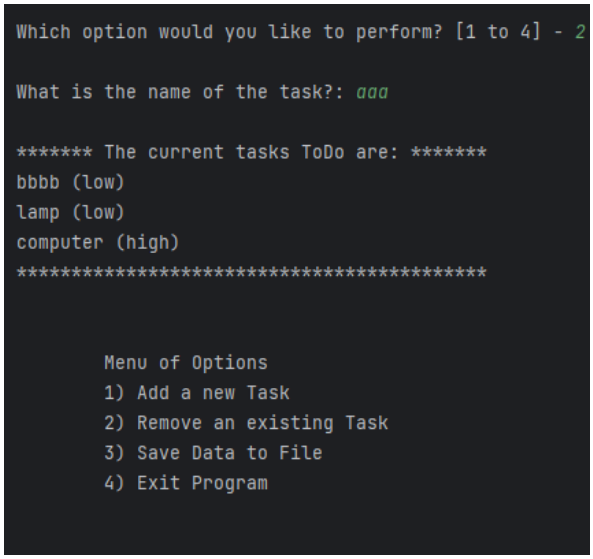
```
def input_new_task_and_priority():

task = str(input("What is the task?: ")).strip()

priority = str(input("What is the priority?: ")).strip()

return task, priority
```

Ok after making a few other modifications we tested the program. First we add new data to a file:

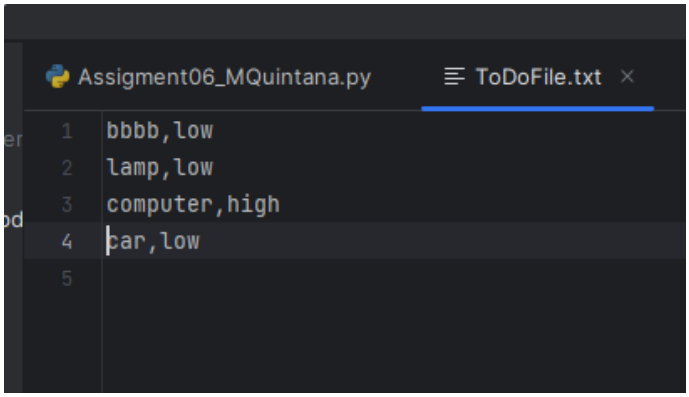


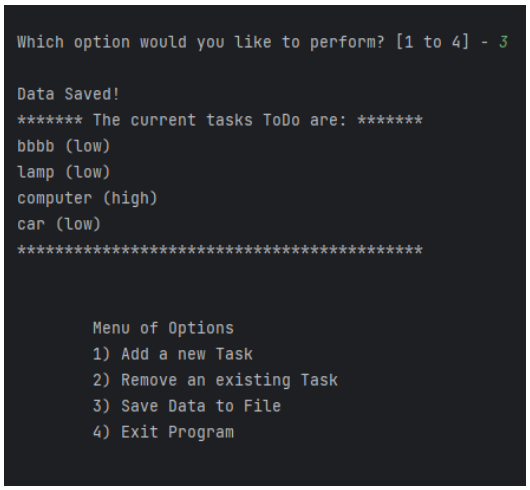
Then we removed a task from the file, but first we tried a random task that is not in the file to show the message that the task is not found on the list, followed by the task that is found on the list:




After that we tested option 3, we add a new task “car” and we saved:



We tested option 4: