

Luis F Lopez Penunuri

February 19, 2023

IT FDN 110 A

Assignment 06

<https://github.com/fernandoPenunuri34/IntroToProg-Python-Mod06.git>

Functions

Introduction

In this practice, we need to improve the system for Jose. Let's assume that we are going to sell him version 3.0 of the previous system. To achieve this, we will implement some new functions.

Practice

In this practice, we implemented some functions, and it was hard for me to understand. The main problem for me was the management of the dictionary. This is a new concept for me, but I believe I understand the idea now.

My Code:

Menu:

This code is almost the same as the previous assignment, but we can notice a difference in how the functions are implemented in the menu. In assignment number 5, the code for each choice was placed inside an 'if' statement for that option. In this code, the 'menu' acts like a 'main' function, providing structure for how the code should be run (Figure 6.1 and Figure 6.2).

```

while (True):
    # Step 3 Show current data
    I0.output_current_tasks_in_list(list_of_rows=table_lst) # Show current data in the list/table
    I0.output_menu_tasks() # Shows menu
    choice_str = I0.input_menu_choice() # Get menu option
    # Step 4 - Process user's menu choice
    if choice_str.strip() == '1': # Add a new Task
        try: #This try/catch helps to solve an issue I had
            task, priority = I0.input_new_task_and_priority()
        except TypeError:
            print("Please try again. \n")
        table_lst = Processor.add_data_to_list(task=task, priority=priority, list_of_rows=table_lst)
        continue # to show the menu
    elif choice_str == '2': # Remove an existing Task
        task = I0.input_task_to_remove()
        table_lst = Processor.remove_data_from_list(task=task, list_of_rows=table_lst)
        continue # to show the menu
    elif choice_str == '3': # Save Data to File
        table_lst = Processor.write_data_to_file(file_name=file_name_str, list_of_rows=table_lst)
        print("Data Saved! \n")
        continue # to show the menu
    elif choice_str == '4': # Exit Program
        print("Did you save last changes?")
        strExit = str(input("Are you sure you want to exit Y/N:"))
        if strExit in ["y", "Y"]:
            print("Goodbye!")
            break
        else:
            continue
    elif choice_str == '4':

```

Figure 6.1 Menu choice (Back to Menu)

```

class I0:
    """ Performs Input and Output tasks """
    @staticmethod
    def output_menu_tasks():
        """ Display a menu of choices to the user
        :return: nothing
        """
        print('
Menu of Options
1) Add a new Task
2) Remove an existing Task
3) Save Data to File
4) Exit Program
')
        print() # Add an extra line for looks
    @staticmethod

```

Figure 6.2 Menu (Back to Menu)

The implementation of functions can increase the complexity of a system, but it also enhances the flexibility of the code. Here's how I managed the value selected by the user. I added a validation to ensure that the user selects only a valid option (Figure 6.3).

```
def input_menu_choice():
    """ Gets the menu choice from a user
    :return: string
    """
    choice = str(input("Which option would you like to perform? [1 to 4] -")).strip()
    if choice >= '5': #validate menu options
        print("Not a valid option")
    print() # Add an extra line for looks
    return choice
```

Figure 6.3 Menu choice validation

Function “input_new_task_and_prioity”

This function creates an empty dictionary named objRow and assigns the user's input for "task" and "priority" as its key-value pairs. The script checks whether the "task" value is numeric and prints an error message if it is. Similarly, it checks whether the "priority" value is numeric and in the valid range of 1-3. If the input is valid, the function returns the task and priority. Otherwise, it prints an error message and returns nothing (Figure 6.4).

```
def input_new_task_and_priority():
    """ Gets task and priority values to be added to the list: return: (string, string) with
    objRow = {}
    objRow["task"] = input("Enter Tasks:")
    objRow["priority"] = input("Enter Priority (1-High, 2-Medium, 3-Low): ")

    # Checking if Task is numeric
    if objRow["task"].isnumeric() or objRow["task"].isdigit():
        print("Sorry", "" + objRow["task"] + "", "is not a valid.")

    elif objRow["priority"].isnumeric(): # Checking if Priority is numeric
        if objRow["priority"] not in ["1", "2", "3"]: #Validated value of priority
            print("Sorry", "" + objRow["priority"] + "", "is not a valid priority")
        else:
            # Return (string, string) with task and priority
            #strPriority = str(objRow["priority"])
            return objRow["task"], objRow["priority"]
    else:
        print("Sorry", "" + objRow["priority"] + "", "is not a valid priority")
```

Figure 6.4 Input new task

Function `remove_data_from_list`

Similar to the previous assignment, this function uses a flag to validate the input value and removes it from the list. Additionally, I added a note to inform the user that the data needs to be saved. Although it is not a validation, I believe it is helpful. I considered adding a function to save the data after removing the value, but since the user needs to click the save icon every time they use Word or Excel, I decided against it ([Figure 6.5](#)).

```
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
    """
    for row_dic in list_of_rows:
        bTaskFound = False
        if row_dic['Task'] == task:
            list_of_rows.remove(row_dic)
            bTaskFound = True
            print(f"Task {task} has been removed")
            print("Please click (Option 3) to save it \n")
            break

    if not bTaskFound:
        print("Task not found.\n")

    return list_of_rows
```

Figure 6.5 Remove data from the list

Running the code PyCharm

Option 1 Add a new Task (Figure 6.6)

```
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

Enter Tasks:bbb
Enter Priority (1-High, 2-Medium, 3-Low): 1
***** The current tasks ToDo are: *****
ppp (2)
aaa (2)
bbb (1)
```

Figure 6.6 Adding Value

Option 2 Remove an existing task, in this example we removed value “aaa” (Figure 6.7)

```
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 the task name to be removed:aaa
Task aaa has been removed
Please click (Option 3) to save it

***** The current tasks ToDo are: *****
ppp (2)
bbb (1)
```

Figure 6.7 Remove task

Option 3 Save data to File. ([Figure 6.8](#) and [Figure 6.9](#))

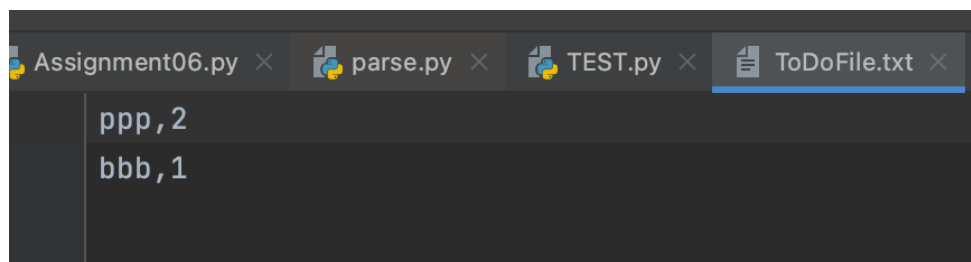
```
Menu of Options
1) Add a new Task
2) Remove an existing Task
3) Save Data to File
4) Exit Program

Which option would you like to perform? [1 to 4] -3

Data Saved!

***** The current tasks ToDo are: *****
ppp (2)
bbb (1)
```

Figure 6.8 Save data



The image shows a code editor window with four tabs: 'Assignment06.py', 'parse.py', 'TEST.py', and 'ToDoFile.txt'. The 'ToDoFile.txt' tab is selected and active. The editor displays the following text:

```
ppp,2
bbb,1
```

Figure 6.9 ToDoFile.txt document

Option 4 Exit the program (Figure 6.10), in this section I have a conversation with friend and he mentioned that we should have a validation since. I do believe it was a good idea and I added it (Figure 6.10).

```
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

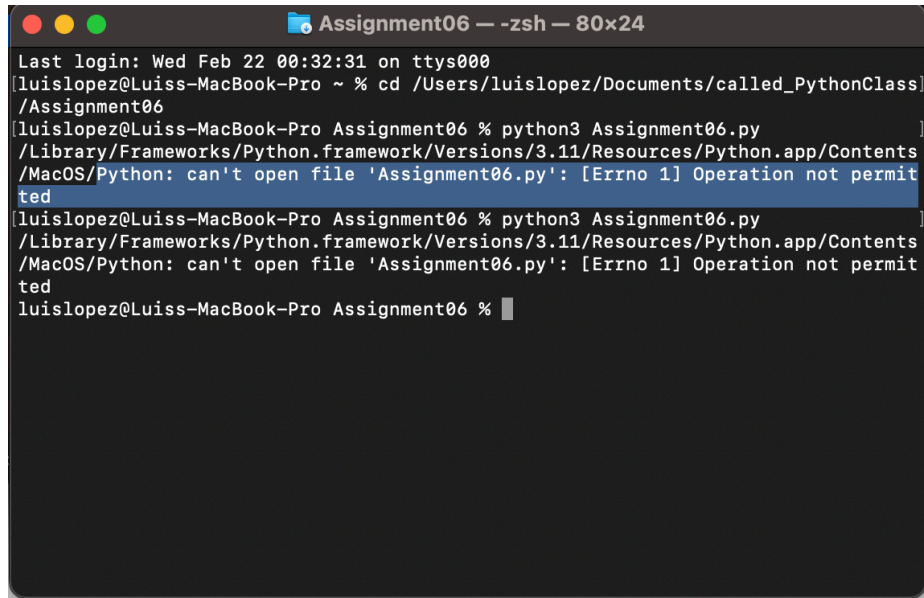
Did you save last changes?
Are you sure you want to exit Y/N:y
Goodbye!

Process finished with exit code 0
```

Figure 6.10 Exit

Running the code Terminal

Option 1 Add a new Task (Figure 6.11), I don't know why but I was not able to run this code in terminal. Still working on it. I need help!!

A screenshot of a macOS terminal window titled "Assignment06 - -zsh - 80x24". The terminal shows the following sequence of commands and output:

```
Last login: Wed Feb 22 00:32:31 on ttys000
[luisslopez@Luiss-MacBook-Pro ~ % cd /Users/luisslopez/Documents/called_PythonClass]
/Assignment06
[luisslopez@Luiss-MacBook-Pro Assignment06 % python3 Assignment06.py
/Library/Frameworks/Python.framework/Versions/3.11/Resources/Python.app/Contents
/MacOS/Python: can't open file 'Assignment06.py': [Errno 1] Operation not permit
ted
[luisslopez@Luiss-MacBook-Pro Assignment06 % python3 Assignment06.py
/Library/Frameworks/Python.framework/Versions/3.11/Resources/Python.app/Contents
/MacOS/Python: can't open file 'Assignment06.py': [Errno 1] Operation not permit
ted
luisslopez@Luiss-MacBook-Pro Assignment06 %
```

Figure 6.11 Terminal not working

Conclusion

As Assignment06 states, 'This project is like the last one, but different enough to be a challenge.' I can see the advantages of using functions, but I believe that understanding abstraction is one of the most difficult parts to grasp when you start coding.

I spoke with Jose, and he was very excited about the new system, but he said that if I don't get at least a 90, he won't buy it... It's up to you. :)"