11.2 Task Management System 📝

Read these guidelines in their entirety before implementing anything.

CSW8 Learning Goals

In this project, similar in structure to the Grades Management System, create a to-do list management system.

Just like you did before, incorporate functions from multiple labs to implement an interactive program, which you can use to collect and create data needed to store user tasks.

This project is intended to be implemented in an IDE, which will allow you to run your code interactively. You will need to create several files in order to implement and test your functions as well as to run your main program.

The “Getting Started” section shows you the steps to modify the Main Program (Template), so that within 15-20 minutes, you should be able to have a fully running skeleton of the system.

Table of Contents

Introduction

You will implement the following features for this system:

- Create an interface that allows the users to interact with the system (will use `while True` and `input()` to collect user data).

- Create a collection to store task information that the users can view and maintain by adding and updating entries.
- Allow the user to save the state of the system by saving the information to file and retrieving it from it.

You will need to use the code that you wrote in previous labs.

Back to top

Main Menu

IMPORTANT: Do NOT use any global variables.

In your main program, you need to define a dictionary `the_menu` that has the options shown below.

```
    "L" : "List"
    "A" : "Add"
    "U" : "Update"
    "D" : "Delete"
    "S" : "Save the data to file"
    "R" : "Restore data from file"
    "Q" : "Quit this program"
```

The menu should be printed by the `print_main_menu()` function when the user starts this system. You already implemented this function in Week 7 Interactive Menu lab.

- You can copy it directly into the **task_functions.py**.

Back to top

Main Program (Template)

Use the following starter code to implement the main loop for your program:

```
# TODO 0: add an import statement to load the functions

the_menu = {} # TODO 1: add the options from the instructions
opt = None

while True:
    # print_main_menu(...) # TODO 2: define the function,
uncomment, and call with the menu as an argument
    opt = input("::: Enter a menu option\n> ")
    opt = opt.upper() # to allow us to input lower- or upper-
case letters

    if ...: # TODO 3: check of the character stored in opt is
```

```
in the_menu dictionary
        print(f"WARNING: {opt} is an invalid menu option.\n")
        continue

    print(f"You selected option {opt} to > {the_menu[opt]}.")

    if opt == ...: # TODO 4: quit the program
        print("Goodbye!\n")
        break # exit the main `while` loop

    # Pause before going back to the main menu
    input("::: Press Enter to continue")

print("Have a nice day!")
```

Back to top

Getting Started

1. Create the requested Python files.
2. Copy the above template into your **main program file**.
3. Upload the files to Gradescope to ensure the names are correct. **You will need to submit *all* files at *the same* time to Gradescope (not in zyBooks).**
4. Address the TODO comments in the code (including adding the `print_main_menu()` to the **task_functions.py**)

Now, you are ready to create the `if/elif` branches to call the functions for the various menu options. In the rest of the instructions, you will get to implement the rest of the options, defining the needed functions, and calling them (remember to remove the `TODO` comments).

Back to top

Create the test file

For *each function that* ***returns*** *something* in the **task_functions.py** file, you should add the corresponding `assert` statements to the **task_tests.py**.

For now, since the next steps will rely on it:

- add the `get_written_date()` function implementation from LAB 7.18 to the **task_functions.py**
- add the `assert` statements to check the function correctness in your **test file**

If you see an `AssertionError`, check the `line` number that the error message pointed to. In IDLE, you can select the "**Show Line Numbers**" option from the top "Options" menu. (Other IDLE tips.)

Pro Tip: As you are reading the instructions, immediately begin adding the `assert` statements.

- test various conditions, especially, the edge cases
- copy into a comment the part of the instructions that the assert is supposed to test

Back to top

You should now have a basic structure for the system, and you are ready to begin implementing each option.

TODO: Once you assembled the above template and your main program and the testing code runs without any errors, **submit your files to Gradescope**.

We hope that you enjoy putting this project together!

Back to top

398092.2571084.qx3zqy7

LAB ACTIVITY 11.2.1: Task Management System 0 / 1

main.py

```
1
```

Develop mode Submit mode

Run your program as often as you'd like, before submitting for grading. Below, type any needed input values in the first box, then click **Run program** and observe the program's output in the second box.

Enter program input (optional)

If your code requires input values, provide them here.

Run program

Input (from above) → main.py (Your program)

Program output displayed here

Coding trail of your work What is this?

```
History of your effort will appear here once you begin
working on this zyLab.
```

Trouble with lab?