To-Do List Application

Aditi Dure

June 8, 2024

Abstract

This report details the development of a to-do list application using Python. The application allows users to manage tasks by adding, viewing, completing, and deleting tasks. The report discusses the design, implementation, and usage of the application.

Contents

Chapter 1

Introduction

1.1 Purpose

The purpose of this project is to create a simple and user-friendly to-do list application to help users manage their tasks efficiently.

1.2 Scope

The application provides functionalities to add new tasks, view the list of tasks, mark tasks as completed, and delete tasks. Completed tasks can also be viewed separately.

Chapter 2

Design and Implementation

2.1 Design Approach

The application is designed with simplicity and ease of use in mind. It employs a command-line interface where users can interact with the to-do list through various options.

2.2 Modules and Functions

2.2.1 Task Manager Module

The main functionality is encapsulated in a module named `task_manager.py`, which handles all task-related operations.

Data Structures

- `tasks`: A list to store current tasks.
- `completed_tasks`: A list to store completed tasks.

Functions

- `load_tasks_from_file(filename)`: Loads tasks from a specified file.
- `load_completed_tasks_from_file(filename)`: Loads completed tasks from a specified file.
- `save_tasks_to_file(filename)`: Saves current tasks to a specified file.

- `save_completed_tasks_to_file(filename)`: Saves completed tasks to a specified file.
- `add_task(task)`: Adds a new task to the list.
- `view_tasks()`: Displays all current tasks.
- `complete_task(task_index)`: Marks a task as completed.
- `delete_task(task_index)`: Deletes a task from the list.
- `view_completed_tasks()`: Displays all completed tasks.

2.3 Data Storage

The tasks and completed tasks are stored in separate text files:

- `to_do_list.txt`: This file stores the current tasks.
- `completed_tasks.txt`: This file stores the completed tasks.

2.4 Main Program

The main program provides a command-line interface for interacting with the to-do list. It handles user input and calls the appropriate functions from the `task_manager` module.

Chapter 3

Usage

3.1 Running the Application

To run the application, execute the main program script. The user will be presented with a menu of options to interact with the to-do list.

3.2 Features

- **Add Task**: Allows the user to add a new task to the list.
- **View Tasks**: Displays all current tasks.
- **Mark Task as Completed**: Marks a specified task as completed.
- **Delete Task**: Deletes a specified task from the list.
- **View Completed Tasks**: Displays all tasks that have been marked as completed.
- **Exit**: Exits the application.

3.3 Result Screenshots

```
aditi@aditi-HP-ENVY-Laptop-13-balxxx:~/iith/year-break-2024/python_course/project/final3$ python3 main.py
No previous 'tasks' found.
No 'completed tasks' found.

------------------------------------------------------------
To-Do List Options:
1. Add task
2. View tasks
3. Mark task as completed
4. Delete task
5. View completed tasks
6. Exit
Enter your choice (1/2/3/4/5/6): 1
Enter a task: sleep
Task 'sleep' added successfully!

------------------------------------------------------------
To-Do List Options:
1. Add task
2. View tasks
3. Mark task as completed
4. Delete task
5. View completed tasks
6. Exit
Enter your choice (1/2/3/4/5/6): 1
Enter a task: eat
Task 'eat' added successfully!

------------------------------------------------------------
To-Do List Options:
1. Add task
2. View tasks
3. Mark task as completed
4. Delete task
5. View completed tasks
6. Exit
Enter your choice (1/2/3/4/5/6): 1
Enter a task: exercise
Task 'exercise' added successfully!
```

Figure 3.1: terminal

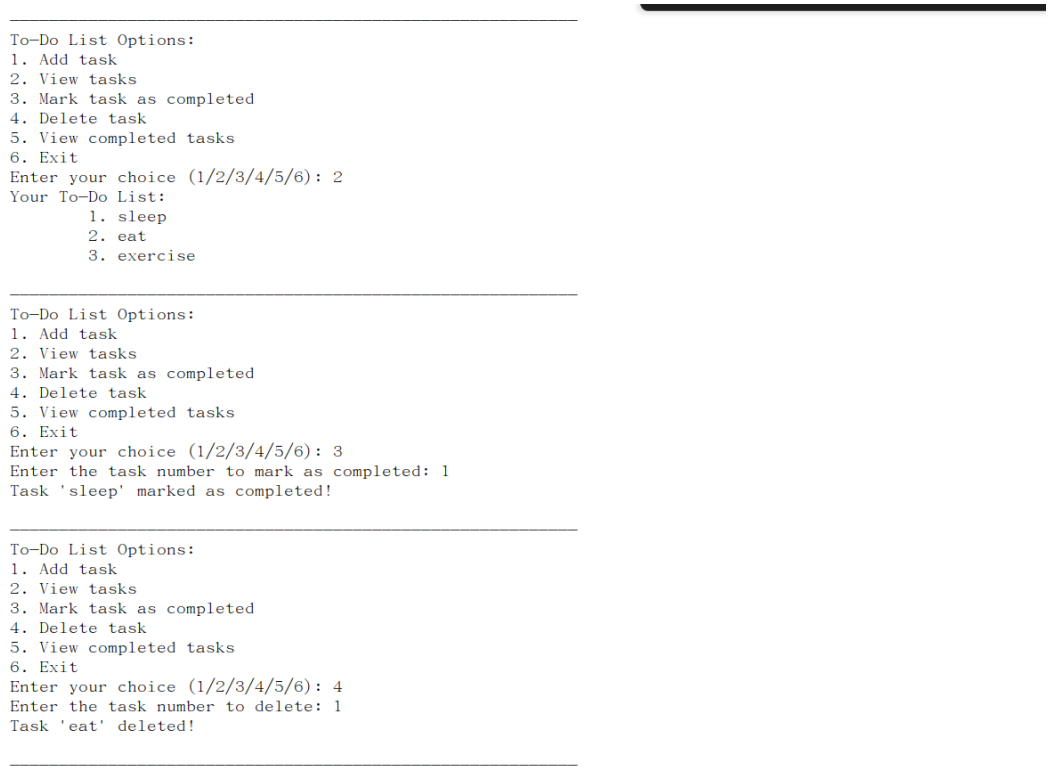

Figure 3.2: terminal

```
To-Do List Options:
1. Add task
2. View tasks
3. Mark task as completed
4. Delete task
5. View completed tasks
6. Exit
Enter your choice (1/2/3/4/5/6): 2
Your To-Do List:
        1. exercise

------------------------------------------------------------
To-Do List Options:
1. Add task
2. View tasks
3. Mark task as completed
4. Delete task
5. View completed tasks
6. Exit
Enter your choice (1/2/3/4/5/6): 5
Completed Tasks:
        1. sleep

------------------------------------------------------------
To-Do List Options:
1. Add task
2. View tasks
3. Mark task as completed
4. Delete task
5. View completed tasks
6. Exit
Enter your choice (1/2/3/4/5/6): 6
Are you sure you want to exit? (yes/no): yes
Do you want to erase data from the text files before exiting? (yes/no): no
Exiting To-Do List.
```

Figure 3.3: terminal

```
aditi@aditi-HP-ENVY-Laptop-13-balxxx:~/iith/year-break-2024/python_course/project/final3$ python3 main.py
'Tasks' already exist. Do you want to clear them? (yes/no): no
Existing 'tasks' kept.
'Completed tasks' already exist. Do you want to clear them? (yes/no): yes
Existing 'completed tasks' cleared.

____________________________________________________________
To-Do List Options:
1. Add task
2. View tasks
3. Mark task as completed
4. Delete task
5. View completed tasks
6. Exit
Enter your choice (1/2/3/4/5/6): 2
Your To-Do List:
        1. exercise

____________________________________________________________
To-Do List Options:
1. Add task
2. View tasks
3. Mark task as completed
4. Delete task
5. View completed tasks
6. Exit
Enter your choice (1/2/3/4/5/6): 5
No completed tasks yet.

____________________________________________________________
To-Do List Options:
1. Add task
2. View tasks
3. Mark task as completed
4. Delete task
5. View completed tasks
6. Exit
Enter your choice (1/2/3/4/5/6):
```

Figure 3.4: terminal

```
1 exercise
```

Figure 3.5: to_do_list.txt

```
1
```

Figure 3.6: completed_tasks.txt

Chapter 4

Conclusion

The to-do list application provides a simple and effective way for users to manage their tasks. Future improvements could include a graphical user interface and additional features such as task prioritization and reminders.

Appendix A

Code Listings

A.1 task_manager.py

```
tasks = []  # List operations
completed_tasks = []  # List operations

def load_tasks_from_file(filename):  # Function definition
    try:  # Exception handling
        with open(filename, 'r') as file:  # File handling
            lines = file.readlines()
            if not lines:  # Conditional statement
                print("No previous 'tasks' found.")  # Print
    statement
                return False  # Return statement
            for line in lines:  # Loop
                tasks.append(line.strip())  # List operations
        return True
    except FileNotFoundError:
        print("No previous 'tasks' found.")
        return False

def load_completed_tasks_from_file(filename):
    try:
        with open(filename, 'r') as file:
            lines = file.readlines()
            if not lines:
                print("No 'completed tasks' found.")
                return False
            for line in lines:
                completed_tasks.append(line.strip())
        return True
    except FileNotFoundError:
```

```
        print("No 'completed tasks' found.")
        return False

def save_tasks_to_file(filename):
    with open(filename, 'w') as file:
        for task in tasks:
            file.write(task + '\n')  # String operations

def save_completed_tasks_to_file(filename):
    with open(filename, 'w') as file:
        for task in completed_tasks:
            file.write(task + '\n')

def add_task(task):
    tasks.append(task)
    print(f"Task '{task}' added successfully!")  # Print
    statement
    save_tasks_to_file("to_do_list.txt")

def view_tasks():
    if tasks:
        print("Your To-Do List:")
        for i, task in enumerate(tasks, 1):  # Loop with
    enumeration
            print(f"\t{i}. {task}")
    else:
        print("Your To-Do List is empty.")

def complete_task(task_index):
    if 0 < task_index <= len(tasks):
        completed_task = tasks.pop(task_index - 1)  # List
    operations
        completed_tasks.append(completed_task)
        print(f"Task '{completed_task}' marked as completed!")
        save_tasks_to_file("to_do_list.txt")
        save_completed_tasks_to_file("completed_tasks.txt")
    else:
        print("Invalid task number.")

def delete_task(task_index):
    if 0 < task_index <= len(tasks):
        deleted_task = tasks.pop(task_index - 1)
        print(f"Task '{deleted_task}' deleted!")
        save_tasks_to_file("to_do_list.txt")
    else:
        print("Invalid task number.")

def view_completed_tasks():
```

```
    if completed_tasks:
        print("Completed Tasks:")
        for i, task in enumerate(completed_tasks, 1):
            print(f"\t{i}. {task}")
    else:
        print("No completed tasks yet.")
```

Listing A.1: Task Manager Module

A.2 main.py

```
import task_manager as tm  # Module import

tasks_loaded = tm.load_tasks_from_file("to_do_list.txt")  # File handling
completed_tasks_loaded = tm.load_completed_tasks_from_file("completed_tasks.txt")

if tasks_loaded:  # Conditional statement
    clear_existing_tasks = input("'Tasks' already exist. Do you want to clear them? (yes/no): ")  # User input
    if clear_existing_tasks.lower() == 'yes':
        tm.tasks = []  # List operations
        tm.save_tasks_to_file("to_do_list.txt")
        print("Existing 'tasks' cleared.")  # Print statement
    elif clear_existing_tasks.lower() == 'no':
        print("Existing 'tasks' kept.")
    else:
        print("Invalid choice. Please enter 'yes' or 'no'.")

if completed_tasks_loaded:
    clear_existing_completed_tasks = input("'Completed tasks' already exist. Do you want to clear them? (yes/no): ")
    if clear_existing_completed_tasks.lower() == 'yes':
        tm.completed_tasks = []
        tm.save_completed_tasks_to_file("completed_tasks.txt")
        print("Existing 'completed tasks' cleared.")
    else:
        print("Existing 'completed tasks' kept.")

while True:  # Loop
    print("\n------------------------------------------------------------\nTo-Do List Options:")
    print("1. Add task")
    print("2. View tasks")
```

```
    print("3. Mark task as completed")
    print("4. Delete task")
    print("5. View completed tasks")
    print("6. Exit")

    choice = input("Enter your choice (1/2/3/4/5/6): ")

    if choice == '1':
        task = input("Enter a task: ")
        tm.add_task(task)  # Function call
    elif choice == '2':
        tm.view_tasks()
    elif choice == '3':
        task_index = int(input("Enter the task number to mark as completed: "))  # Type conversion
        tm.complete_task(task_index)
    elif choice == '4':
        task_index = int(input("Enter the task number to delete: "))
        tm.delete_task(task_index)
    elif choice == '5':
        tm.view_completed_tasks()
    elif choice == '6':
        confirm_exit = input("Are you sure you want to exit? (yes/no): ")
        if confirm_exit.lower() == 'yes':
            erase_data = input("Do you want to erase data from the text files before exiting? (yes/no): ")
            if erase_data.lower() == 'yes':
                open("to_do_list.txt", 'w').close()
                open("completed_tasks.txt", 'w').close()
            print("Exiting To-Do List.")
            break  # Loop control
        elif confirm_exit.lower() == 'no':
            continue
        else:
            print("Invalid choice. Please enter 'yes' or 'no'.")
    else:
        print("Invalid choice. Please select a valid option.")
```

Listing A.2: Main Program

Appendix B

References

- Python Course: Techgyan.