To-Do List Application Report

Date: 22/11/2025

This report provides a concise overview of a simple Python command-line to-do list application, summarizing its purpose, design, and possible improvements.

Introduction

The application is a lightweight Python script that allows users to view, add, and remove tasks using a text-based menu. It uses basic Python data structures and is easy to run without extra dependencies.

Problem Statement

Users often need a quick and simple way to track tasks without complex tools. This script offers essential task management but lacks features such as data saving, editing, and prioritizing.

Functional Requirements

- View tasks in a numbered list
- Add new tasks
- Remove tasks using their list number
- Exit the program
- Handle invalid inputs gracefully

Non-Functional Requirements

- Easy-to-use interface
- Fast and lightweight execution
- Basic reliability through simple error handling
- Works on any system with Python installed

System Architecture

The program is procedural, using a single function that loops through menu options. Tasks are stored temporarily in a Python list, with no permanent storage.

Design Decisions

A list was chosen to store tasks for simplicity. A menu-driven interface was used to keep the program easy for beginners. Persistence was intentionally excluded to keep the script minimal.

Implementation Summary

The script initializes an empty list for tasks, repeatedly displays a menu, and performs actions based on user input. Tasks are added with append() and removed with pop() after validating indices.

Testing

Testing consisted of manually trying all menu options, checking edge cases such as empty task lists and invalid inputs.

Challenges

The program has no persistence, and input validation is limited. The command-line interface also restricts usability but fits the goal of simplicity.

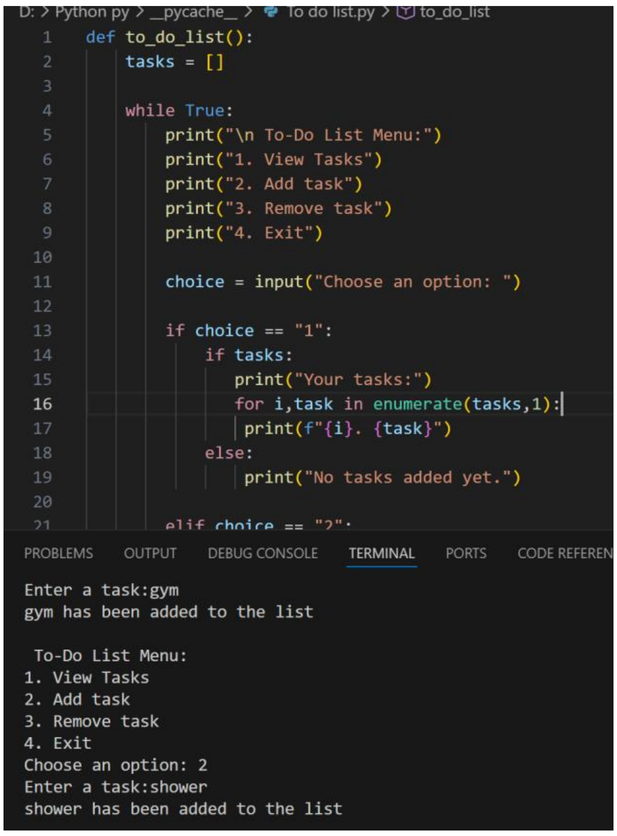
Key Learnings

The project demonstrates how simple tools can be effective. Clean loops, conditionals, and input handling are essential for interactive programs. The design allows room for future expansion.

Future Enhancements

- Save tasks to a file
- Add priorities or due dates
- Allow editing and searching
- Support multiple users
- Improve validation and modularity

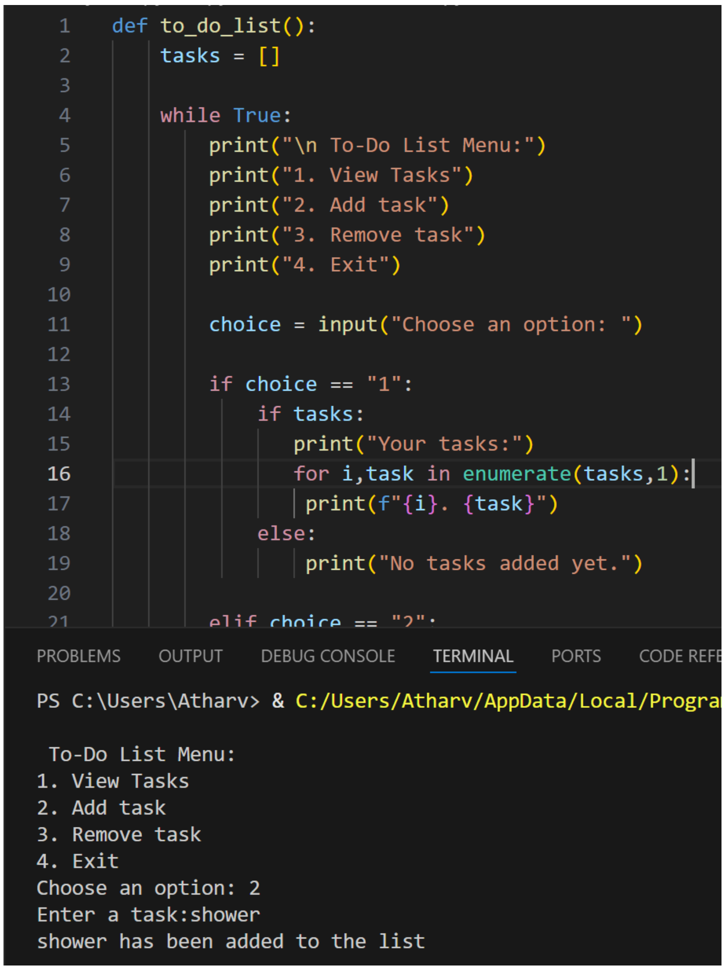
RESULT SCREENSHOTS



```
D: > Python py > __pycache__ > To do list.py > to_do_list
def to_do_list():
    tasks = []

    while True:
        print("\n To-Do List Menu:")
        print("1. View Tasks")
        print("2. Add task")
        print("3. Remove task")
        print("4. Exit")

        choice = input("Choose an option: ")

        if choice == "1":
            if tasks:
                print("Your tasks:")
                for i,task in enumerate(tasks,1):
                 print(f"{i}. {task}")
            else:
                print("No tasks added yet.")

        elif choice == "2":
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS CODE REFEREN

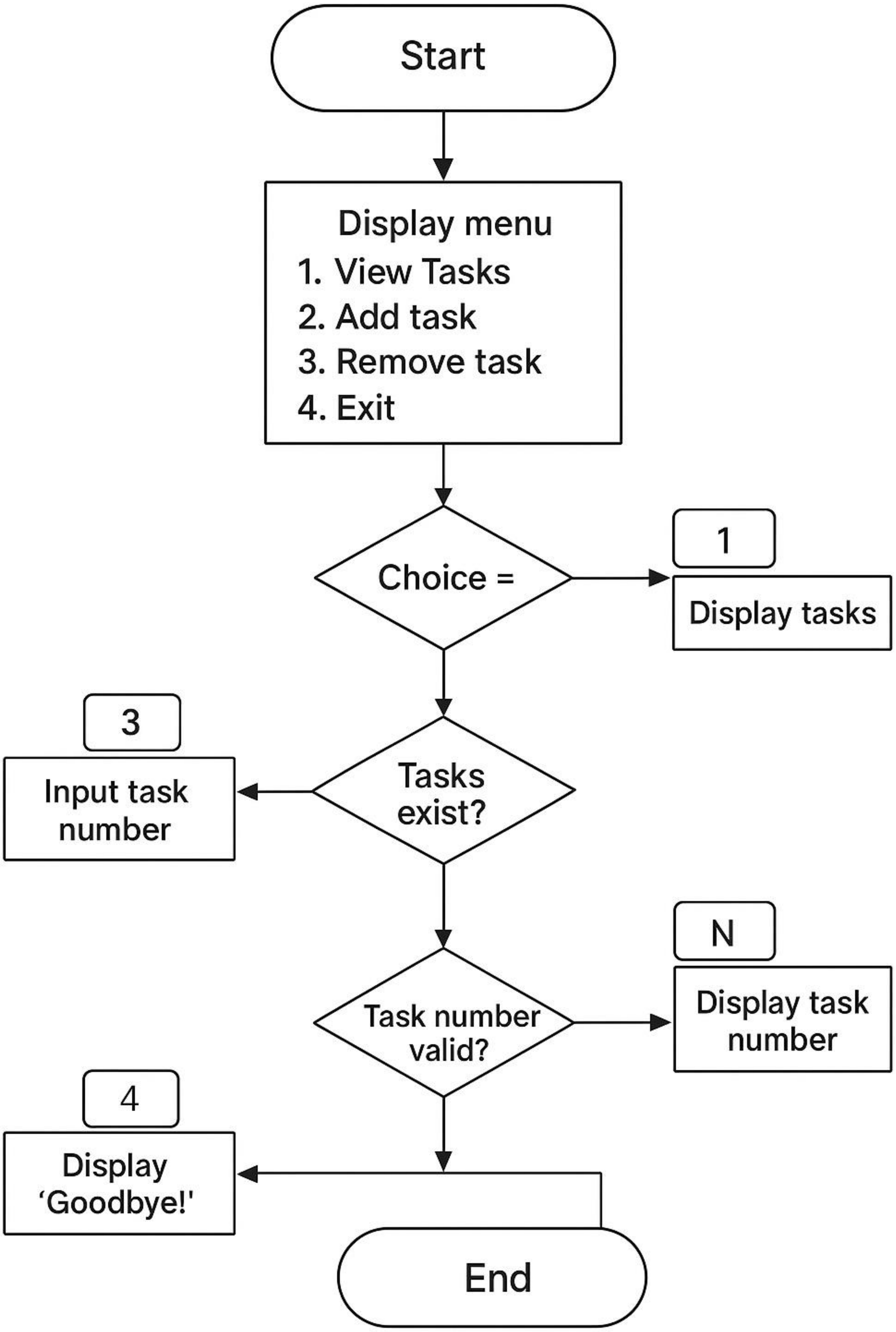
```
Enter a task:gym
gym has been added to the list

 To-Do List Menu:
1. View Tasks
2. Add task
3. Remove task
4. Exit
Choose an option: 2
Enter a task:shower
shower has been added to the list
```



```
def to_do_list():
    tasks = []

    while True:
        print("\n To-Do List Menu:")
        print("1. View Tasks")
        print("2. Add task")
        print("3. Remove task")
        print("4. Exit")

        choice = input("Choose an option: ")

        if choice == "1":
            if tasks:
                print("Your tasks:")
                for i,task in enumerate(tasks,1):
                 print(f"{i}. {task}")
            else:
                print("No tasks added yet.")

        elif choice == "2":
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS CODE REFE

```
PS C:\Users\Atharv> & C:/Users/Atharv/AppData/Local/Progra

 To-Do List Menu:
1. View Tasks
2. Add task
3. Remove task
4. Exit
Choose an option: 2
Enter a task:shower
shower has been added to the list
```



Flowchart of the program