

* **Name : Tahir Fareed**
* **Roll No: SU92-BSAIM-F24-036**
* **Section: BSAI (3A)**
* **Subject: AI Lab**
* **Task : 01**

# To-Do List Application in Python

## Introduction

The To-Do List Application is a simple console-based program developed in Python.   
It allows users to add, remove, and display their daily tasks efficiently.   
The application is built using Object-Oriented Programming (OOP) concepts to ensure modularity and reusability.

## Objectives

- To demonstrate the use of classes and objects in Python.  
- To manage tasks dynamically using Python lists.  
- To implement menu-driven console applications.  
- To provide a simple task management system for users.

## Tools & Technologies

- Programming Language: Python 3  
- Concepts Used: Object-Oriented Programming (Classes, Objects, Methods)  
- Data Structure: List  
- Environment: Console-based execution

## System Design

The system is designed around a single class `ToDoList` which manages all tasks.   
The class contains methods for adding, removing, and displaying tasks.   
The program uses a `main()` function that provides a menu-driven interface for user interaction.

## Implementation

class ToDoList:  
 def \_\_init\_\_(self):  
 self.tasks = []  
  
 def add\_task(self, task):  
 self.tasks.append(task)  
 print("Task added successfully!")  
  
 def remove\_task(self, task\_index):  
 if 1 <= task\_index <= len(self.tasks):  
 removed = self.tasks.pop(task\_index - 1)  
 print(f"Task '{removed}' removed successfully!")  
 else:  
 print("Invalid task index!")  
  
 def display\_tasks(self):  
 if self.tasks:  
 print("\nYour To-Do List:")  
 for index, task in enumerate(self.tasks, start=1):  
 print(f"{index}. {task}")  
 else:  
 print("Your To-Do List is empty.")  
  
  
def main():  
 todo\_list = ToDoList()  
  
 while True:  
 print("\n===== To-Do List App =====")  
 print("1. Add Task")  
 print("2. Remove Task")  
 print("3. Display Tasks")  
 print("4. Exit")  
  
 choice = input("Enter your choice: ")  
  
 if choice == "1":  
 task = input("Enter the task: ")  
 todo\_list.add\_task(task)  
 elif choice == "2":  
 try:  
 task\_index = int(input("Enter the task number to remove: "))  
 todo\_list.remove\_task(task\_index)  
 except ValueError:  
 print("Please enter a valid number.")  
 elif choice == "3":  
 todo\_list.display\_tasks()  
 elif choice == "4":  
 print("Exiting...")  
 break  
 else:  
 print("Invalid choice. Please try again.")  
  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 main()

## Working / Execution

1. The program starts and displays a menu with four options.  
2. If the user chooses option 1, they can add a new task.  
3. If the user chooses option 2, they can remove an existing task by number.  
4. If the user chooses option 3, the program displays all the current tasks.  
5. If the user chooses option 4, the program exits successfully.

## Example Execution

===== To-Do List App =====  
1. Add Task  
2. Remove Task  
3. Display Tasks  
4. Exit  
  
Enter your choice: 1  
Enter the task: Buy groceries  
Task added successfully!  
  
Enter your choice: 3  
Your To-Do List:  
1. Buy groceries

## Conclusion

The To-Do List Application demonstrates the effective use of Python programming and OOP principles.   
It shows how lists can be used to dynamically manage tasks and how a menu-driven interface can improve user interaction.   
This project provides a foundation for building more advanced task management systems in the future.