# Mini Project: Dynamic To-Do List in Python

This mini project implements a Dynamic To-Do List using Python. It allows users to add, view, remove, and reorder tasks dynamically. The project demonstrates the use of functions, input handling, lists, and string processing.

## Code Explanation

### 1. add\_task()

This function takes input from the user and appends the task to the list. It ignores empty tasks. It first asks the user to enter a task using input().The entered task is appended to the task\_list using the append() method. Finally, a confirmation message is displayed to tell the user that the task was added successfully.

### 2. view\_task()

This function prints all current tasks with their serial numbers. It checks if the list is empty before displaying tasks. First, it checks if the list is empty. If yes, it prints a message like *"No tasks in the list.* If the list has tasks, it loops through them using enumerate(), which gives both the index (serial number) and the task name. Each task is displayed in a numbered format so the user can easily identify them.

### 3. remove\_task()

This function removes a task based on its index number. It uses try-except to handle invalid input or out-of-range values. First, it calls view\_tasks() to display the tasks Then it asks the user to enter the task number they want to remove. The task is removed using the pop() method, which deletes the item at the given index. A try-except block is used to handle errors: ValueError if the user enters something other than a number. IndexError if the user enters a number outside the valid range.

### 4. reorder\_task()

This function allows the user to rearrange tasks by entering a new order of indices. It ensures that all task numbers are included exactly once, otherwise it shows an error. First, it displays the current tasks. The user is then asked to enter a new order of indices (e.g., 2 1 3).These numbers are converted into a list of integers using map (int, input().split()). It checks if the entered order contains all task numbers exactly once using sorted(new\_order). If valid, it reorders the list by creating a new order with list slicing. If not valid, it prints an error message.

### 5. Main Loop

The main loop repeatedly shows the menu and calls the corresponding function based on the user's choice. It uses an infinite while True loop. Each time, it displays the menu using show\_menu(). Based on the user’s choice (1, 2, 3, 4, or 5), it calls the appropriate function. If the user enters 5, the loop breaks and the program end. For any other invalid input, it prints "Invalid choice. Try again."

## Output Screenshots

Below are example outputs of the program when running in a terminal:

1. Adding tasks  
   2. Viewing tasks  
   3. Removing tasks  
   4. Reordering tasks  
   5. Exiting the program



