Assignment: Build a To-Do List Backend

Objective
Create a simple backend for managing a to-do list. This exercise focuses on **basic data storage, functions, and state updates**. You'll practice handling collections of data, creating and modifying entries, and returning meaningful results.

Part 1: Initialization

Function: `new_list()`

- **Task:** Return an empty to-do list data structure (array, list, dictionary, etc. depending on your language).

Part 2: Adding Tasks

Function: `add_task(todo_list, description)`

- **Tasks:**
    1. Insert a new task with a text description.
    2. Each task should include:
        - A unique ID (e.g., incrementing number)
        - A description string
        - A completion status (default = False).
    3. Return the updated list.

Part 3: Listing Tasks

Function: `list_tasks(todo_list)`

- **Tasks:**
    1. Return all tasks in the current list.
    2. Each task should display:
        - ID
        - Description
        - Completion status.

Part 4: Completing Tasks

Function: `complete_task(todo_list, task_id)`

- **Tasks:**
    1. Find the task with the given ID.
    2. Mark it as completed (status = True).
    3. If the task ID doesn't exist, return an error/False.

Part 5: (Optional) Deleting Tasks

Function: `delete_task(todo_list, task_id)`

- **Tasks:**
    1. Remove the task with the given ID.
    2. Return the updated list.

Part 6: (Optional Demo Flow)

Function: `demo()`

- **Tasks:**
    1. Start with a new empty list.
    2. Add a few tasks.
    3. Mark one complete.
    4. List current tasks.
    5. Optionally delete a task.

Stretch Goal: Sorting Tasks

Add functionality to organize tasks more effectively:

Function: `sort_tasks(todo_list, mode)`

- **Tasks:**
    1. Accept a `mode` parameter to decide the sort order.
        - `"by_status"` → show incomplete tasks first, then completed ones.
        - `"by_description"` → alphabetical order by task description.
        - `"by_id"` → default order of creation.
    2. Return a new list sorted accordingly.