# TASK- 2 TO DO LIST

This Python code creates a To-Do List application with a graphical user interface (GUI) using Tkinter and a SQLite database to store tasks. It includes features for adding, viewing, marking as done, deleting tasks, viewing task history, and setting reminders for due dates. Let's break down the code in detail:

1. Imports:
   - `sqlite3`: For interacting with the SQLite database.
   - `tkinter as tk`: For creating the GUI.
   - `from tkinter import messagebox`: For displaying message boxes (errors, confirmations, etc.).
   - `from tkcalendar import Calendar`: For a calendar widget to select due dates.
   - `import datetime`: For working with dates and times.
   - `import threading`: For running the due date checker in a separate thread.
2. `ToDoList` Class:
   - `__init__(self, db_name="tasks.db")`:
     - Initializes the database connection and cursor.
     - Calls `create_table()` to create the necessary database tables if they don't exist.
     - Calls `init_gui()` to set up the GUI.
     - Calls `start_due_date_checker()` to begin the background thread for due date reminders.
   - `create_table(self)`:
     - Creates the `tasks` table with columns for `id`, `title`, `description`, `category`, `priority`, `due_date`, and `status`.
     - Creates the `history` table to log task actions (added, completed, deleted) with a timestamp.
   - `add_task(self)`:
     - Retrieves task details from the GUI entries and calendar.
     - Validates that the title is not empty.
     - Inserts the new task into the `tasks` table.

    - Gets the last inserted row ID (task ID) and logs the "Added" action in the `history` table.
    - Calls `view_tasks()` to refresh the task list in the GUI.
- `view_tasks(self)`:
    - Clears the current task list in the GUI.
    - Retrieves tasks from the `tasks` table (ID, title, status).
    - Inserts each task into the listbox in the format "ID - Title (Status)".
- `mark_done(self)`:
    - Gets the selected task from the listbox.
    - If no task is selected, displays an error message.
    - Extracts the task ID and title from the selected item.
    - Updates the task status to "Completed" in the `tasks` table.
    - Logs the "Completed" action in the `history` table.
    - Calls `view_tasks()` to refresh the task list.
- `delete_task(self)`:
    - Similar to `mark_done()`, but deletes the selected task from the `tasks` table and logs the "Deleted" action.
- `log_history(self, task_id, title, action)`:
    - Inserts a record into the `history` table with the task ID, title, action, and current timestamp.
- `view_history(self)`:
    - Creates a new top-level window to display the task history.
    - Retrieves history entries from the `history` table.
    - Inserts each history entry into a listbox in the history window.
- `check_due_dates(self)`:
    - This function runs in a separate thread.
    - It continuously checks for tasks due today.
    - If any due tasks are found, it displays a warning message box.
    - Uses `threading.Event().wait(86400)` to pause the thread for 24 hours (86400 seconds) before checking again.
- `start_due_date_checker(self)`:

    - Creates a new thread and sets the `check_due_dates` function as its target.
    - Sets the thread as a daemon thread (so it will exit when the main program exits).
    - Starts the thread.
- `init_gui(self)`:
    - Creates the main window (`self.root`).
    - Sets up labels, entry fields, the calendar widget, and buttons for all the task operations.
    - Creates the listbox to display tasks.
    - Calls `view_tasks()` to initially populate the task list.

3. Main Execution Block:

- `if __name__ == "__main__":`: Ensures that the `ToDoList()` object is only created when the script is run directly (not imported as a module).
- `ToDoList()`: Creates an instance of the `ToDoList` class, which initializes the GUI and starts the application.

Key Improvements and Features:

- Database Integration: Uses SQLite to persistently store tasks.
- Task History: Logs all task actions (add, complete, delete) with timestamps.
- Due Date Reminders: Uses a background thread to check for due tasks and display reminders.
- Clearer GUI: Improved layout and organization of GUI elements.
- Error Handling: Includes basic error handling (e.g., checking for empty title, no selected task).
- Use of `tkcalendar`: Provides a user-friendly way to select due dates.
- Threading: Runs the due date check in a separate thread to prevent blocking the GUI.

This code provides a solid foundation for a To-Do List application. It could be further enhanced with features like editing tasks, setting priorities, searching/filtering tasks, and a more visually appealing GUI.