ToDoリストアプリ

```
import tkinter as tk
from tkinter import messagebox
import sqlite3
from datetime import datetime

conn = None
cursor = None

def connect_database():
    global conn, cursor
    conn = sqlite3.connect('todo.db')
    cursor = conn.cursor()

def close_database():
    global conn
    conn.close()

def create_todo_table(cursor):
    # ToDoテーブルを作成
    cursor.execute('''
        CREATE TABLE IF NOT EXISTS todos (
            id INTEGER PRIMARY KEY AUTOINCREMENT,
            task TEXT NOT NULL,
            completed BOOLEAN NOT NULL,
            due_date DATE,
            due_time TIME,
            end_time TIME
        )
    ''')

def submit_form():
    try:
        task = entry_task.get()
        duedate = entry_due_date.get()
        duetime = entry_due_time.get()
        endtime = entry_end_time.get()

        # フォーマットが正しいか確認
        datetime.strptime(duedate, '%Y-%m-%d')
        if duetime.strip():
            datetime.strptime(duetime, '%H:%M')
        if endtime.strip():
            datetime.strptime(endtime, '%H:%M')

        # ToDoをデータベースに追加
        connect_database()
        add_todo(task, duedate, duetime, endtime)
        close_database()

        messagebox.showinfo("フォームの内容", f"タスク: {task},
         日付: {duedate} が送信されました。")

        # フォーム送信後にデータを再取得して表示
        connect_database()
        display_todos()
        close_database()

```

```
    except ValueError as e:
        messagebox.showerror("エラー",
         f"日付または時間の形式が正しくありません。詳細: {str(e)}")

def add_todo(task, due_date=None, due_time=None, end_time=None):
    # 日付と時間を文字列からdatetimeオブジェクトに変換
    # due_date = datetime.strptime(f'{due_date} {due_time}',
     '%Y-%m-%d %H:%M:%S') if due_date else None
    due_date = datetime.strptime(due_date, '%Y-%m-%d') if due_date else None
    due_time = datetime.strptime(due_time, '%H:%M') if due_time else None
    end_time = datetime.strptime(end_time, '%H:%M') if end_time else None

    cursor.execute('INSERT INTO todos (task, completed, due_date, due_time, end_time)
     VALUES (?, ?, ?, ?, ?)',
     (task, False, due_date, due_time, end_time))
    conn.commit()

def delete_selected_item():
    global conn, cursor

    selected_index = listbox.curselection()

    if selected_index:
        # 選択されたアイテムのインデックスを取得
        selected_index = selected_index[0]

        # 選択されたアイテムを取得
        selected_item = listbox.get(selected_index)

        # |で分割して、各要素を取得
        parts = [part.strip() for part in selected_item.split('|')]
        print(parts)

        # "タスク: "の部分を取得して削除
        task_info = parts[0].split(': ')[-1]
        due_date_info = parts[1].split(': ')[-1]
        due_time_info = parts[2].split(': ')[-1]
        end_time_info = parts[3].split(': ')[-1]

        # データベースから対応する要素を削除
        connect_database()
        cursor.execute("DELETE FROM todos WHERE task = ? AND due_date LIKE ?
         AND due_time LIKE ? AND end_time LIKE ?",
        (task_info, f'%{due_date_info}%', f'%{due_time_info}%', f'%{end_time_info}%'))
        # データベース変更をコミット
        conn.commit()
        close_database()

        # 選択されたアイテムを削除
        listbox.delete(selected_index)

        # 新しい値でアイテムを挿入
        new_value = "削除"
        listbox.insert(selected_index, new_value)

        # # フォーム送信後にデータを再取得して表示
        # connect_database()
        # display_todos()
```

```
        # close_database()

def update_selected_item():
    global conn, cursor

    try:
        task = entry_task.get()
        duedate = entry_due_date.get()
        duetime = entry_due_time.get()
        endtime = entry_end_time.get()

        # フォーマットが正しいか確認
        datetime.strptime(duedate, '%Y-%m-%d')
        if duetime.strip():
            datetime.strptime(duetime, '%H:%M')
        if endtime.strip():
            datetime.strptime(endtime, '%H:%M')



        selected_index = listbox.curselection()

        if selected_index:
            # 選択されたアイテムのインデックスを取得
            selected_index = selected_index[0]

            # 選択されたアイテムを取得
            selected_item = listbox.get(selected_index)

            # |で分割して、各要素を取得
            parts = [part.strip() for part in selected_item.split('|')]

            # "タスク: "の部分を取得して削除
            task_info = parts[0].split(': ')[-1]
            due_date_info = parts[1].split(': ')[-1]
            due_time_info = parts[2].split(': ')[-1]
            end_time_info = parts[3].split(': ')[-1]

            # データベースから対応する要素を削除
            connect_database()
            cursor.execute("UPDATE todos SET task=?, due_date=?, due_time=?,
             end_time=? WHERE task=? AND due_date LIKE ? AND due_time LIKE ? AND end_time LIKE ?",
            (task, duedate, duetime, endtime, task_info, f'%{due_date_info}%',
             f'%{due_time_info}%', f'%{end_time_info}%'))
            # データベース変更をコミット
            conn.commit()
            close_database()

            # 選択されたアイテムを削除
            listbox.delete(selected_index)

            # 新しい値でアイテムを挿入
            new_value = f"タスク: {task:<20} | 日付: {duedate:<15} | 開始時間: {duetime:<10} |
             終了時間: {endtime:<10}"
            listbox.insert(selected_index, new_value)

    except ValueError as e:
        messagebox.showerror("エラー", f"日付または時間の形式が正しくありません。詳細: {str(e)}")
```

```
174
175 def display_todos():
176     # 日付と時間でソートしてToDoリストを取得
177     cursor.execute("SELECT task, strftime('%Y-%m-%d', due_date), strftime('%H:%M', due_time),
178      strftime('%H:%M', end_time) FROM todos ORDER BY due_date, due_time")
179     todos = cursor.fetchall()
180
181     # エントリーをクリア
182     entry_task.delete(0, tk.END)
183     entry_due_date.delete(0, tk.END)
184     entry_due_time.delete(0, tk.END)
185     entry_end_time.delete(0, tk.END)
186
187     # リストボックスをクリア
188     listbox.delete(0, tk.END)
189
190     # ToDoをリストボックスに表示
191     for todo in todos:
192         task = todo[0] if todo[0] is not None else "未設定"
193         due_date = todo[1] if todo[1] is not None else "未設定"
194         due_time = todo[2] if todo[2] is not None else "未設定"
195         end_time = todo[3] if todo[3] is not None else "未設定"
196
197         listbox.insert(tk.END, f"タスク: {task:<20} | 日付: {due_date:<15} |
198          開始時間: {due_time:<10} | 終了時間: {end_time:<10}")
199
200 # Tkinterウィンドウの作成
201 root = tk.Tk()
202 root.title("タスク入力画面")
203
204 # ラベルとエントリー(タスク)
205 label_task = tk.Label(root, text="タスク")
206 label_task.pack(padx=10)
207 entry_task = tk.Entry(root, width=30)   # 幅を20に設定
208 entry_task.pack(padx=10)
209
210 # ラベルとエントリ(日付)
211 label_due_date = tk.Label(root, text="年月日(yyyy-mm-dd)")
212 label_due_date.pack(padx=10)
213 entry_due_date = tk.Entry(root, width=30)   # 幅を20に設定
214 entry_due_date.pack(padx=10)
215
216 # ラベルとエントリー(開始時間)
217 label_due_time = tk.Label(root, text="開始時間(hh:mm)")
218 label_due_time.pack(padx=10)
219 entry_due_time = tk.Entry(root, width=30)   # 幅を20に設定
220 entry_due_time.pack(padx=10)
221
222 # ラベルとエントリー(終了時間)
223 label_end_time = tk.Label(root, text="終了時間(hh:mm)")
224 label_end_time.pack(padx=10)
225 entry_end_time = tk.Entry(root, width=30)   # 幅を20に設定
226 entry_end_time.pack(padx=10)
227
228
229 # 送信ボタン
230 submit_button = tk.Button(root, text="送信", command=submit_form)
231 submit_button.pack(pady=10, padx=10)
```

```
232
233 # リストボックスの作成
234 listbox = tk.Listbox(root, width=80)    # 幅を50に設定
235 listbox.pack(pady=10, padx=10)
236
237 # 削除ボタンの作成
238 delete_button = tk.Button(root, text="選択したアイテムを削除", command=delete_selected_item)
239 delete_button.pack(pady=10)
240
241 # 変更ボタンの作成
242 update_button = tk.Button(root, text="選択したアイテムを変更", command=update_selected_item)
243 update_button.pack(pady=10)
244
245 # フォーム起動時にデータを表示
246 connect_database()
247 display_todos()
248 close_database()
249
250 # Tkinterメインループ
251 root.mainloop()
```