

ToDoリストアプリ

```
1 import tkinter as tk
2 from tkinter import messagebox
3 import sqlite3
4 from datetime import datetime
5
6 conn = None
7 cursor = None
8
9 def connect_database():
10     global conn, cursor
11     conn = sqlite3.connect('todo.db')
12     cursor = conn.cursor()
13
14 def close_database():
15     global conn
16     conn.close()
17
18 def create_todo_table(cursor):
19     # ToDoテーブルを作成
20     cursor.execute('''
21         CREATE TABLE IF NOT EXISTS todos (
22             id INTEGER PRIMARY KEY AUTOINCREMENT,
23             task TEXT NOT NULL,
24             completed BOOLEAN NOT NULL,
25             due_date DATE,
26             due_time TIME,
27             end_time TIME
28         )
29     ''')
30
31 def submit_form():
32     try:
33         task = entry_task.get()
34         duedate = entry_due_date.get()
35         duetime = entry_due_time.get()
36         endtime = entry_end_time.get()
37
38         # フォーマットが正しいか確認
39         datetime.strptime(duedate, '%Y-%m-%d')
40         if duetime.strip():
41             datetime.strptime(duetime, '%H:%M')
42         if endtime.strip():
43             datetime.strptime(endtime, '%H:%M')
44
45         # ToDoをデータベースに追加
46         connect_database()
47         add_todo(task, duedate, duetime, endtime)
48         close_database()
49
50         messagebox.showinfo("フォームの内容", f"タスク: {task},
51             日付: {duedate} が送信されました。")
52
53         # フォーム送信後にデータを再取得して表示
54         connect_database()
55         display_todos()
56         close_database()
57
```

```

58     except ValueError as e:
59         messagebox.showerror("エラー",
60             f"日付または時間の形式が正しくありません。詳細: {str(e)}")
61
62 def add_todo(task, due_date=None, due_time=None, end_time=None):
63     # 日付と時間を文字列からdatetimeオブジェクトに変換
64     # due_date = datetime.strptime(f'{due_date} {due_time}',
65         '%Y-%m-%d %H:%M:%S') if due_date else None
66     due_date = datetime.strptime(due_date, '%Y-%m-%d') if due_date else None
67     due_time = datetime.strptime(due_time, '%H:%M') if due_time else None
68     end_time = datetime.strptime(end_time, '%H:%M') if end_time else None
69
70     cursor.execute('INSERT INTO todos (task, completed, due_date, due_time, end_time)
71         VALUES (?, ?, ?, ?, ?)',
72         (task, False, due_date, due_time, end_time))
73     conn.commit()
74
75 def delete_selected_item():
76     global conn, cursor
77
78     selected_index = listbox.curselection()
79
80     if selected_index:
81         # 選択されたアイテムのインデックスを取得
82         selected_index = selected_index[0]
83
84         # 選択されたアイテムを取得
85         selected_item = listbox.get(selected_index)
86
87         # |で分割して、各要素を取得
88         parts = [part.strip() for part in selected_item.split('|')]
89         print(parts)
90
91         # "タスク: "の部分を取得して削除
92         task_info = parts[0].split(': ')[-1]
93         due_date_info = parts[1].split(': ')[-1]
94         due_time_info = parts[2].split(': ')[-1]
95         end_time_info = parts[3].split(': ')[-1]
96
97         # データベースから対応する要素を削除
98         connect_database()
99         cursor.execute("DELETE FROM todos WHERE task = ? AND due_date LIKE ?
100             AND due_time LIKE ? AND end_time LIKE ?",
101             (task_info, f'%{due_date_info}%', f'%{due_time_info}%', f'%{end_time_info}%'))
102         # データベース変更をコミット
103         conn.commit()
104         close_database()
105
106         # 選択されたアイテムを削除
107         listbox.delete(selected_index)
108
109         # 新しい値でアイテムを挿入
110         new_value = "削除"
111         listbox.insert(selected_index, new_value)
112
113         # フォーム送信後にデータを再取得して表示
114         # connect_database()
115         # display_todos()

```

```

116         # close_database()
117
118     def update_selected_item():
119         global conn, cursor
120
121         try:
122             task = entry_task.get()
123             duedate = entry_due_date.get()
124             duetime = entry_due_time.get()
125             endtime = entry_end_time.get()
126
127             # フォーマットが正しいか確認
128             datetime.strptime(duedate, '%Y-%m-%d')
129             if duetime.strip():
130                 datetime.strptime(duetime, '%H:%M')
131             if endtime.strip():
132                 datetime.strptime(endtime, '%H:%M')
133
134
135             selected_index = listbox.curselection()
136
137             if selected_index:
138                 # 選択されたアイテムのインデックスを取得
139                 selected_index = selected_index[0]
140
141                 # 選択されたアイテムを取得
142                 selected_item = listbox.get(selected_index)
143
144                 # | で分割して、各要素を取得
145                 parts = [part.strip() for part in selected_item.split('|')]
146
147                 # "タスク: "の部分を取得して削除
148                 task_info = parts[0].split(': ')[-1]
149                 due_date_info = parts[1].split(': ')[-1]
150                 due_time_info = parts[2].split(': ')[-1]
151                 end_time_info = parts[3].split(': ')[-1]
152
153                 # データベースから対応する要素を削除
154                 connect_database()
155                 cursor.execute("UPDATE todos SET task=?, due_date=?, due_time=?,
156                                end_time=? WHERE task=? AND due_date LIKE ? AND due_time LIKE ? AND end_time LIKE ?",
157                                (task, duedate, duetime, endtime, task_info, f'%{due_date_info}%',
158                                f'%{due_time_info}%', f'%{end_time_info}%'))
159                 # データベース変更をコミット
160                 conn.commit()
161                 close_database()
162
163                 # 選択されたアイテムを削除
164                 listbox.delete(selected_index)
165
166                 # 新しい値でアイテムを挿入
167                 new_value = f"タスク: {task:<20} | 日付: {duedate:<15} | 開始時間: {duetime:<10} |
168                             終了時間: {endtime:<10}"
169                 listbox.insert(selected_index, new_value)
170
171         except ValueError as e:
172             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)}")
173

```

```

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()
```