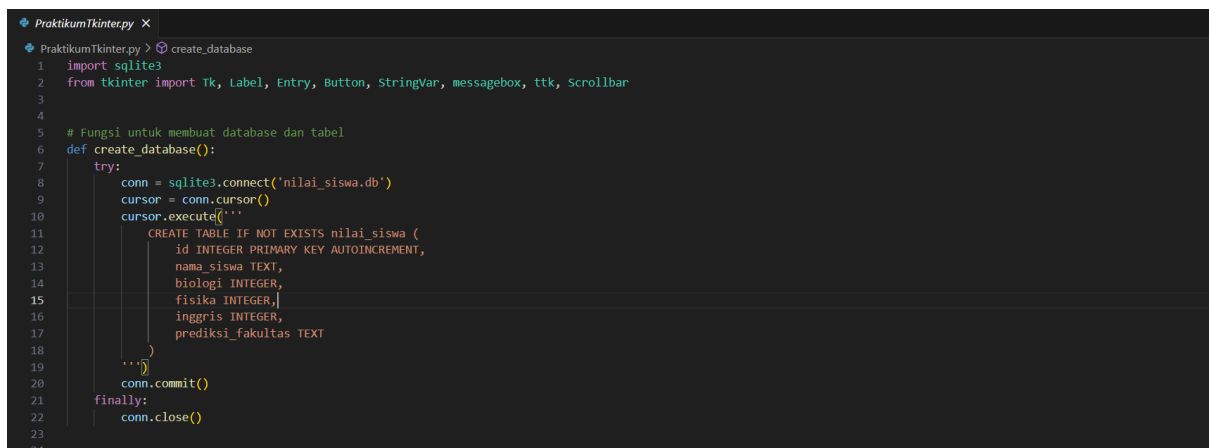


Nama : Rafli Ramadhani Oktavianto Khufi

Kelas : TI C

NIM : 20230140127

Link Repository : [https://github.com/DhaniCuPliZ/TkinterCRUD\\_127.git](https://github.com/DhaniCuPliZ/TkinterCRUD_127.git)



```
PraktikumTkinter.py > create_database
1 import sqlite3
2 from tkinter import Tk, Label, Entry, Button, StringVar, messagebox, ttk, Scrollbar
3
4
5 # Fungsi untuk membuat database dan tabel
6 def create_database():
7     try:
8         conn = sqlite3.connect('nilai_siswa.db')
9         cursor = conn.cursor()
10        cursor.execute('''
11            CREATE TABLE IF NOT EXISTS nilai_siswa (
12                id INTEGER PRIMARY KEY AUTOINCREMENT,
13                nama_siswa TEXT,
14                biologi INTEGER,
15                fisika INTEGER,
16                inggris INTEGER,
17                prediksi_fakultas TEXT
18            )
19        ''')
20        conn.commit()
21    finally:
22        conn.close()
```

Def create\_database() yang digunakan untuk membuat database dan tabel jika belum ada.

Memulai blok try, yang akan menangani kemungkinan terjadinya kesalahan (exception) saat membuat atau membuka database.

conn = sqlite3.connect('nilai\_siswa.db') : Membuka koneksi ke database SQLite bernama nilai\_siswa.db. Jika file database belum ada, SQLite akan membuatnya.

cursor = conn.cursor() : Membuat objek cursor yang digunakan untuk mengeksekusi perintah SQL pada database.

cursor.execute(" CREATE TABLE IF NOT EXISTS nilai\_siswa : Menjalankan perintah SQL untuk membuat tabel nilai\_siswa jika tabel tersebut belum ada

id INTEGER PRIMARY KEY AUTOINCREMENT : kolom integer yang berfungsi sebagai kunci utama dan auto-increment.

nama\_siswa TEXT : kolom untuk menyimpan nama siswa dalam bentuk teks.

biologi INTEGER, fisika INTEGER, inggris INTEGER, : kolom untuk menyimpan nilai-nilai dalam bentuk integer.

prediksi\_fakultas TEXT : kolom untuk menyimpan hasil prediksi fakultas dalam bentuk teks.

conn.commit() : Menyimpan perubahan yang telah dilakukan pada database (commit). Tanpa ini, perubahan tidak akan diterapkan

finally: memastikan bahwa blok finally akan dijalankan, baik terjadi kesalahan maupun tidak, untuk menutup koneksi.

conn.close() : Menutup koneksi ke database, memastikan bahwa sumber daya dibebaskan setelah penggunaan selesai

```
24
25 # Fungsi untuk mengambil semua data dari database
26 def fetch_data():
27     try:
28         conn = sqlite3.connect('nilai_siswa.db')
29         cursor = conn.cursor()
30         cursor.execute("SELECT * FROM nilai_siswa")
31         rows = cursor.fetchall()
32     finally:
33         conn.close()
34     return rows
35
```

def fetch\_data(): Mendefinisikan fungsi fetch\_data() yang digunakan untuk mengambil data dari tabel nilai\_siswa di database.

conn = sqlite3.connect('nilai\_siswa.db') : Membuka koneksi ke database SQLite yang bernama nilai\_siswa.db.

cursor = conn.cursor() : Membuat objek cursor, yang digunakan untuk menjalankan perintah SQL pada database.

cursor.execute("SELECT \* FROM nilai\_siswa") : Menjalankan perintah SQL SELECT \* FROM nilai\_siswa untuk mengambil semua data yang ada dalam tabel nilai\_siswa.

rows = cursor.fetchall() : Mengambil semua baris hasil dari perintah SQL di atas dan menyimpannya dalam variabel rows sebagai daftar (list) yang berisi tuple-tuple data.

finally: memastikan bahwa blok finally akan dijalankan, baik terjadi kesalahan maupun tidak, untuk menutup koneksi.

conn.close() : Menutup koneksi ke database, memastikan bahwa sumber daya dibebaskan setelah penggunaan selesai

return rows : Mengembalikan hasil yang disimpan dalam variabel rows, yaitu daftar semua baris yang diambil dari tabel nilai\_siswa.

```
37 # Fungsi untuk menyimpan data baru ke database
38 def save_to_database(nama, biologi, fisika, inggris, prediksi):
39     try:
40         conn = sqlite3.connect('nilai_siswa.db')
41         cursor = conn.cursor()
42         cursor.execute("""
43             INSERT INTO nilai_siswa (nama_siswa, biologi, fisika, inggris, prediksi_fakultas)
44             VALUES (?, ?, ?, ?, ?)
45             """, (nama, biologi, fisika, inggris, prediksi))
46         conn.commit()
47     finally:
48         conn.close()
49
```

def save\_to\_database(nama, biologi, fisika, inggris, prediksi):  
Mendefinisikan fungsi save\_to\_database() yang digunakan untuk menyimpan data baru ke dalam tabel nilai\_siswa di database. Fungsi ini menerima parameter nama, biologi, fisika, inggris, dan prediksi yang akan disimpan.

Try : Memulai blok try untuk menangani kemungkinan kesalahan (exception) yang bisa terjadi saat melakukan operasi database, seperti kegagalan koneksi atau kesalahan SQL.

conn = sqlite3.connect('nilai\_siswa.db') : Membuka koneksi ke database SQLite yang bernama nilai\_siswa.db.

cursor.execute("""

INSERT INTO nilai\_siswa (nama\_siswa, biologi, fisika, inggris, prediksi\_fakultas)

VALUES (?, ?, ?, ?, ?)

""", (nama, biologi, fisika, inggris, prediksi)) : Menjalankan perintah SQL INSERT INTO untuk menyisipkan data baru ke dalam tabel nilai\_siswa. Tanda tanya (?) digunakan sebagai placeholder untuk mencegah SQL injection, dan nilai-nilai dari parameter fungsi (nama, biologi, fisika, inggris, prediksi) dimasukkan ke dalam tempat-tempat tersebut.

conn.commit() : Menyimpan perubahan yang telah dilakukan pada database dengan memanggil commit(). Ini memastikan bahwa data yang baru disisipkan akan disimpan secara permanen di database.

finally: memastikan bahwa blok finally akan dijalankan, baik terjadi kesalahan maupun tidak, untuk menutup koneksi.

Conn.close() : Menutup koneksi ke database untuk membebaskan sumber daya setelah operasi selesai.

```
51 # Fungsi untuk memperbarui data di database
52 def update_database(record_id, nama, biologi, fisika, inggris, prediksi):
53     try:
54         conn = sqlite3.connect('nilai_siswa.db')
55         cursor = conn.cursor()
56         cursor.execute('''
57             UPDATE nilai_siswa
58             SET nama_siswa = ?, biologi = ?, fisika = ?, inggris = ?, prediksi_fakultas = ?
59             WHERE id = ?
60             ''', (nama, biologi, fisika, inggris, prediksi, record_id))
61         conn.commit()
62     finally:
63         conn.close()
```

def update\_database(record\_id, nama, biologi, fisika, inggris, prediksi):  
Mendefinisikan fungsi update\_database() yang digunakan untuk memperbarui data pada tabel nilai\_siswa berdasarkan record\_id. Fungsi ini menerima parameter record\_id, nama, biologi, fisika, inggris, dan prediksi, yang akan digunakan untuk memperbarui data.

try: Memulai blok try untuk menangani kemungkinan kesalahan (exception) yang dapat terjadi selama operasi, seperti kegagalan koneksi atau kesalahan dalam perintah SQL.

conn = sqlite3.connect('nilai\_siswa.db') : Membuka koneksi ke database SQLite yang bernama nilai\_siswa.db.

cursor = conn.cursor() : Membuat objek cursor, yang digunakan untuk mengeksekusi perintah SQL pada database.

cursor.execute("""

UPDATE nilai\_siswa  
SET nama\_siswa = ?, biologi = ?, fisika = ?, inggris = ?,  
prediksi\_fakultas = ?  
WHERE id = ?

""', (nama, biologi, fisika, inggris, prediksi, record\_id)) : Menjalankan perintah SQL UPDATE untuk memperbarui data pada tabel nilai\_siswa berdasarkan record\_id.

Tanda tanya (?) digunakan sebagai placeholder untuk nilai-nilai yang akan disisipkan (parameter fungsi: nama, biologi, fisika, inggris, prediksi, dan record\_id). Perintah ini akan memperbarui kolom nama\_siswa, biologi, fisika, inggris, dan prediksi\_fakultas untuk baris yang memiliki id yang sama dengan record\_id.

conn.commit() : Menyimpan perubahan yang telah dilakukan pada database dengan memanggil commit(). Ini memastikan bahwa pembaruan data akan disimpan secara permanen di database.

Finally : Memulai blok finally, yang akan selalu dieksekusi setelah blok try, baik ada kesalahan ataupun tidak. Biasanya digunakan untuk membersihkan atau menutup sumber daya yang telah digunakan.

conn.close() : Menutup koneksi ke database untuk membebaskan sumber daya setelah operasi selesai.

```
67 def delete_database(record_id):
68     try:
69         conn = sqlite3.connect('nilai_siswa.db')
70         cursor = conn.cursor()
71         cursor.execute('DELETE FROM nilai_siswa WHERE id = ?', (record_id,))
72         conn.commit()
73     finally:
74         conn.close()
```

def delete\_database(record\_id): Mendefinisikan fungsi delete\_database(), yang digunakan untuk menghapus data dari tabel nilai\_siswa berdasarkan record\_id. Fungsi ini menerima parameter record\_id, yang menentukan baris mana yang akan dihapus.

try: Memulai blok try untuk menangani kemungkinan kesalahan (exception) yang dapat terjadi selama operasi, seperti kegagalan koneksi atau kesalahan dalam perintah SQL.

conn = sqlite3.connect('nilai\_siswa.db') : Membuka koneksi ke database SQLite yang bernama nilai\_siswa.db.

cursor.execute('DELETE FROM nilai\_siswa WHERE id = ?', (record\_id,)) : Menjalankan perintah SQL DELETE untuk menghapus data dari tabel nilai\_siswa, dengan kondisi WHERE id = ?. Tanda tanya (?) digunakan sebagai placeholder untuk nilai yang akan disisipkan. Di sini, nilai yang disisipkan adalah record\_id yang diberikan saat fungsi dipanggil. Perintah

ini akan menghapus baris yang memiliki nilai id yang sama dengan record\_id.

conn.commit() : Menyimpan perubahan yang dilakukan pada database dengan commit(). Ini memastikan bahwa penghapusan data akan diterapkan secara permanen di database.

Finally : Memulai blok finally, yang akan selalu dieksekusi setelah blok try, baik ada kesalahan ataupun tidak. Biasanya digunakan untuk membersihkan atau menutup sumber daya yang telah digunakan.

conn.close() : Menutup koneksi ke database untuk membebaskan sumber daya setelah operasi selesai.

```
78 def calculate_prediction(biologi, fisika, inggris):
79     if biologi > fisika and biologi > inggris:
80         return "Kedokteran"
81     elif fisika > biologi and fisika > inggris:
82         return "Teknik"
83     elif inggris > biologi and inggris > fisika:
84         return "Bahasa"
85     else:
86         return "Tidak Diketahui"
```

def calculate\_prediction(biologi, fisika, inggris): Mendefinisikan fungsi calculate\_prediction() yang menerima tiga parameter: biologi, fisika, dan inggris. Fungsi ini akan mengembalikan prediksi fakultas berdasarkan nilai tertinggi dari ketiga mata pelajaran tersebut.

if biologi > fisika and biologi > inggris: Kondisi pertama untuk memeriksa apakah nilai Biologi lebih besar daripada nilai Fisika dan Inggris. Jika kondisi ini benar (nilai Biologi lebih besar dari kedua mata pelajaran lainnya), maka program akan mengembalikan hasil "Kedokteran", yang diprediksi sebagai fakultas yang sesuai berdasarkan nilai tertinggi di Biologi.

return "Kedokteran" : Jika kondisi sebelumnya benar, maka fungsi akan mengembalikan string "Kedokteran", yang menunjukkan bahwa berdasarkan nilai, prediksi fakultas yang sesuai adalah Kedokteran.

elif fisika > biologi and fisika > inggris: Kondisi berikutnya untuk memeriksa apakah nilai Fisika lebih besar dari Biologi dan Inggris. Jika kondisi ini benar (nilai Fisika lebih tinggi daripada kedua mata pelajaran

lainnya), maka program akan mengembalikan "Teknik", yang diprediksi sebagai fakultas yang sesuai berdasarkan nilai tertinggi di Fisika.

return "Teknik" : Jika kondisi sebelumnya benar, maka fungsi akan mengembalikan string "Teknik", yang menunjukkan bahwa berdasarkan nilai, prediksi fakultas yang sesuai adalah Teknik.

elif inggris > biologi and inggris > fisika: Kondisi berikutnya untuk memeriksa apakah nilai Inggris lebih besar dari nilai Biologi dan Fisika. Jika kondisi ini benar (nilai Inggris lebih tinggi dari kedua mata pelajaran lainnya), maka program akan mengembalikan "Bahasa", yang diprediksi sebagai fakultas yang sesuai berdasarkan nilai tertinggi di Inggris.

return "Bahasa" : Jika kondisi sebelumnya benar, maka fungsi akan mengembalikan string "Bahasa", yang menunjukkan bahwa berdasarkan nilai, prediksi fakultas yang sesuai adalah Bahasa.

else: Kondisi ini menangani kasus jika tidak ada satu mata pelajaran pun yang memiliki nilai yang lebih besar dari dua mata pelajaran lainnya. Ini dapat terjadi jika dua atau lebih nilai memiliki angka yang sama tertinggi, atau semua nilai memiliki angka yang sama.

return "Tidak Diketahui" : Jika tidak ada mata pelajaran yang jelas lebih unggul daripada yang lain, maka fungsi akan mengembalikan string "Tidak Diketahui", yang berarti prediksi fakultas tidak dapat ditentukan berdasarkan nilai-nilai yang diberikan.

```
90 def submit():
91     try:
92         nama = nama_var.get().strip()
93         biologi = int(biologi_var.get())
94         fisika = int(fisika_var.get())
95         inggris = int(inggris_var.get())
96
97         if not nama:
98             raise ValueError("Nama siswa tidak boleh kosong.")
99
100        prediksi = calculate_prediction(biologi, fisika, inggris)
101        save_to_database(nama, biologi, fisika, inggris, prediksi)
102
103        messagebox.showinfo("Sukses", f"Data berhasil disimpan!\nPrediksi Fakultas: {prediksi}")
104        clear_input()
105        populate_table()
106    except ValueError as e:
107        messagebox.showerror("Error", f"Kesalahan input: {e}")
108    except Exception as e:
109        messagebox.showerror("Error", f"Kesalahan: {e}")
```

□ def submit(): Mendefinisikan fungsi submit yang akan dijalankan saat pengguna mengirimkan data.

nama = nama\_var.get().strip() Mengambil nilai dari input nama (nama\_var) dan menghapus spasi di awal dan akhir string menggunakan strip().

biologi = int(biologi\_var.get()) Mengambil nilai input untuk mata pelajaran Biologi (biologi\_var) dan mengonversinya menjadi integer.

fisika = int(fisika\_var.get()) : Mengambil nilai input untuk mata pelajaran Fisika (fisika\_var) dan mengonversinya menjadi integer.

inggris = int(inggris\_var.get()) : Mengambil nilai input untuk mata pelajaran Bahasa Inggris (inggris\_var) dan mengonversinya menjadi integer.

if not nama: Memeriksa apakah input nama kosong (string kosong).

raise ValueError("Nama siswa tidak boleh kosong.") : Jika nama kosong, akan memicu exception ValueError dengan pesan "Nama siswa tidak boleh kosong."

prediksi = calculate\_prediction(biologi, fisika, inggris) :Menghitung prediksi fakultas berdasarkan nilai mata pelajaran menggunakan fungsi calculate\_prediction.

save\_to\_database(nama, biologi, fisika, inggris, prediksi) Menyimpan data yang telah diinput, termasuk prediksi, ke dalam database menggunakan fungsi save\_to\_database.

messagebox.showinfo("Sukses", f'Data berhasil disimpan!\nPrediksi Fakultas: {prediksi}') : Menampilkan pesan sukses dengan informasi bahwa data berhasil disimpan dan menunjukkan prediksi fakultas.

clear\_input() :Memanggil fungsi clear\_input untuk membersihkan form input setelah data berhasil disimpan.

populate\_table()

Memanggil fungsi populate\_table untuk memperbarui atau menampilkan :data yang telah disimpan pada tabel.

except ValueError as e: :Menangkap exception ValueError yang terjadi jika ada kesalahan dalam input data, seperti jika nilai tidak bisa dikonversi ke integer.

messagebox.showerror("Error", f'Kesalahan input: {e}') : Menampilkan pesan kesalahan jika terjadi error saat input dengan pesan yang dihasilkan dari exception.



except Exception as e: Menangkap exception lainnya (error umum) yang mungkin terjadi selama proses submit.

messagebox.showerror("Error", f"Kesalahan: {e}") : Menampilkan pesan kesalahan jika terjadi error umum dalam proses submit.

```
113 def update():
114     try:
115         if not selected_record_id.get():
116             raise Exception("Pilih data dari tabel untuk diupdate!")
117
118         record_id = int(selected_record_id.get())
119         nama = nama_var.get().strip()
120         biologi = int(biologi_var.get())
121         fisika = int(fisika_var.get())
122         inggris = int(inggris_var.get())
123
124         if not nama:
125             raise ValueError("Nama siswa tidak boleh kosong.")
126
127         prediksi = calculate_prediction(biologi, fisika, inggris)
128         update_database(record_id, nama, biologi, fisika, inggris, prediksi)
129
130         messagebox.showinfo("Sukses", "Data berhasil diperbarui.")
131         clear_input()
132         populate_table()
133     except ValueError as e:
134         messagebox.showerror("Error", f"Kesalahan: {e}")
135     except Exception as e:
136         messagebox.showerror("Error", f"Kesalahan: {e}")
137
```

def update(): Mendefinisikan fungsi update() yang akan dijalankan ketika pengguna ingin memperbarui data.

if not selected\_record\_id.get(): Memeriksa apakah pengguna telah memilih data dari tabel yang akan diupdate. selected\_record\_id adalah ID dari data yang dipilih.

raise Exception("Pilih data dari tabel untuk diupdate!") Jika tidak ada ID yang dipilih (kosong), maka akan memicu exception dengan pesan bahwa pengguna harus memilih data terlebih dahulu.

record\_id = int(selected\_record\_id.get()) Mengambil ID dari data yang dipilih dan mengonversinya menjadi integer untuk digunakan dalam pembaruan data.

`nama = nama_var.get().strip()` Mengambil nilai dari input nama (`nama_var`) dan menghapus spasi di awal dan akhir string menggunakan `strip()`.

`biologi = int(biologi_var.get())` Mengambil nilai input untuk mata pelajaran Biologi (`biologi_var`) dan mengonversinya menjadi integer.

`fisika = int(fisika_var.get())` Mengambil nilai input untuk mata pelajaran Fisika (`fisika_var`) dan mengonversinya menjadi integer.

`inggris = int(inggris_var.get())` Mengambil nilai input untuk mata pelajaran Bahasa Inggris (`inggris_var`) dan mengonversinya menjadi integer.

`if not nama:` Memeriksa apakah nama kosong (string kosong). Nama siswa wajib diisi.

`raise ValueError("Nama siswa tidak boleh kosong.")` Jika nama kosong, akan memicu exception `ValueError` dengan pesan bahwa nama siswa tidak boleh kosong.

`prediksi = calculate_prediction(biologi, fisika, inggris)` Menghitung prediksi fakultas berdasarkan nilai mata pelajaran menggunakan fungsi `calculate_prediction`.

`update_database(record_id, nama, biologi, fisika, inggris, prediksi)` Memperbarui data pada database menggunakan fungsi `update_database` dengan ID yang telah dipilih, beserta data baru yang dimasukkan.

`messagebox.showinfo("Sukses", "Data berhasil diperbarui.")` Menampilkan pesan sukses yang memberi tahu pengguna bahwa data berhasil diperbarui.

`clear_input()` Memanggil fungsi `clear_input` untuk membersihkan form input setelah pembaruan berhasil.

`populate_table()` Memanggil fungsi `populate_table` untuk memperbarui atau menampilkan data terbaru dalam tabel.

`except ValueError as e:` Menangkap exception `ValueError` yang terjadi jika ada kesalahan dalam input data, seperti jika nilai tidak bisa dikonversi ke integer.

`messagebox.showerror("Error", f'Kesalahan: {e}')` Menampilkan pesan kesalahan jika terjadi error terkait input (misalnya format yang salah) dengan pesan dari exception.

except Exception as e: Menangkap exception lainnya (error umum) yang mungkin terjadi selama proses update().

messagebox.showerror("Error", f"Kesalahan: {e}") Menampilkan pesan kesalahan jika terjadi error umum selama proses pembaruan data.

```
140 def delete():
141     try:
142         if not selected_record_id.get():
143             raise Exception("Pilih data dari tabel untuk dihapus!")
144
145         record_id = int(selected_record_id.get())
146         delete_database(record_id)
147         messagebox.showinfo("Sukses", "Data berhasil dihapus.")
148         clear_input()
149         populate_table()
150     except Exception as e:
151         messagebox.showerror("Error", f"Kesalahan: {e}")
152
```

def delete(): Mendefinisikan fungsi delete() yang akan dijalankan ketika pengguna ingin menghapus data.

if not selected\_record\_id.get(): Memeriksa apakah pengguna telah memilih data yang akan dihapus. selected\_record\_id berisi ID dari data yang dipilih.

raise Exception("Pilih data dari tabel untuk dihapus!") : Jika tidak ada ID yang dipilih (kosong), maka akan memicu exception dengan pesan bahwa pengguna harus memilih data terlebih dahulu.

record\_id = int(selected\_record\_id.get()) : Mengambil ID dari data yang dipilih dan mengonversinya menjadi integer agar bisa digunakan dalam penghapusan data.

delete\_database(record\_id) : Memanggil fungsi delete\_database untuk menghapus data dari database berdasarkan record\_id yang dipilih.

messagebox.showinfo("Sukses", "Data berhasil dihapus.") : Menampilkan pesan sukses yang memberi tahu pengguna bahwa data telah berhasil dihapus.

clear\_input() : Memanggil fungsi clear\_input untuk membersihkan form input setelah penghapusan data.

`populate_table()` : Memanggil fungsi `populate_table` untuk memperbarui atau menampilkan tabel dengan data terbaru setelah penghapusan.

`except Exception as e:` Menangkap exception umum jika terjadi error dalam proses penghapusan data.

`messagebox.showerror("Error", f'Kesalahan: {e}')` : Menampilkan pesan kesalahan dengan informasi dari exception jika terjadi error selama proses penghapusan data.

```
154 # Fungsi untuk mengisi tabel dengan data dari database
155 def populate_table():
156     for row in tree.get_children():
157         tree.delete(row)
158     for row in fetch_data():
159         tree.insert('', 'end', values=row)
160
```

`def populate_table():` Mendefinisikan fungsi `populate_table` untuk mengisi ulang data pada tabel (treeview).

`for row in tree.get_children():` Mengambil semua elemen atau baris (children) yang saat ini ada di dalam treeview (tree) untuk proses penghapusan.

`tree.delete(row)` : Menghapus setiap baris (node) yang ada dalam treeview. Hal ini dilakukan agar tabel bersih sebelum data baru dimasukkan.

`for row in fetch_data():` Mengiterasi setiap baris data yang diperoleh dari fungsi `fetch_data()`. Fungsi `fetch_data()` biasanya digunakan untuk mengambil data terbaru dari database.

`tree.insert("", 'end', values=row)` : Memasukkan setiap baris data (row) ke dalam treeview.

Parameter `"` menunjukkan bahwa baris baru tidak memiliki parent node (berada di tingkat root).

`'end'` berarti menambahkan baris di bagian akhir tabel.

`values=row` adalah data yang akan ditampilkan di baris tersebut.

```

162 # Fungsi untuk mengisi input dari tabel
163 def fill_inputs_from_table(event):
164     try:
165         selected_item = tree.selection()[0]
166         selected_row = tree.item(selected_item)['values']
167
168         selected_record_id.set(selected_row[0])
169         nama_var.set(selected_row[1])
170         biologi_var.set(selected_row[2])
171         fisika_var.set(selected_row[3])
172         inggris_var.set(selected_row[4])
173     except IndexError:
174         pass
175

```

`def fill_inputs_from_table(event)`: Fungsi ini digunakan untuk mengisi form input berdasarkan data yang dipilih dari tabel (treeview). Fungsi ini dipicu oleh suatu event, seperti klik pada baris tabel.

`try`: Memulai blok try untuk menangani kemungkinan error, seperti ketika tidak ada baris yang dipilih.

`selected_item = tree.selection()[0]` : `tree.selection()`: Mengembalikan daftar ID dari item yang dipilih di treeview.

`[0]`: Mengambil ID pertama dari daftar tersebut (asumsi hanya satu baris yang dipilih).

`selected_row = tree.item(selected_item)['values']` : `tree.item(selected_item)`: Mengambil informasi lengkap dari item (baris) yang dipilih berdasarkan ID-nya.

`['values']`: Ekstrak nilai-nilai (data) yang ditampilkan di kolom tabel untuk baris tersebut.

`selected_record_id.set(selected_row[0])` :Mengatur variabel `selected_record_id` (tipe `StringVar` atau `IntVar`) dengan ID data dari baris yang dipilih (kolom pertama).

`nama_var.set(selected_row[1])` : Mengisi variabel input `nama_var` (tipe `StringVar`) dengan nama siswa dari kolom kedua.

`biologi_var.set(selected_row[2])` : Mengisi variabel input `biologi_var` (tipe `IntVar` atau `StringVar`) dengan nilai Biologi dari kolom ketiga.

`fisika_var.set(selected_row[3])` : Mengisi variabel input `fisika_var` dengan nilai Fisika dari kolom keempat.

`inggris_var.set(selected_row[4])` :Mengisi variabel input `inggris_var` dengan nilai Bahasa Inggris dari kolom kelima.

`except IndexError`: Menangkap error `IndexError` yang terjadi jika pengguna tidak memilih baris mana pun (misalnya, klik area kosong di tabel).

`Pass` : Melewatkan error tanpa melakukan apa pun untuk memastikan aplikasi tidak crash.

```
177 # Fungsi untuk membersihkan input
178 def clear_input():
179     nama_var.set("")
180     biologi_var.set("")
181     fisika_var.set("")
182     inggris_var.set("")
183     selected_record_id.set("")
184
```

`def clear_input()`: Fungsi ini digunakan untuk mengosongkan semua input form dan variabel terkait, termasuk ID record yang dipilih.

`nama_var.set("")` mengatur nilai variabel `nama_var` (biasanya tipe `StringVar`) menjadi string kosong, membersihkan input nama siswa. `biologi_var.set("")`, `fisika_var.set("")`, dan `inggris_var.set("")` masing-masing mengosongkan input nilai untuk mata pelajaran Biologi, Fisika, dan Bahasa Inggris. Terakhir, `selected_record_id.set("")` mengosongkan informasi tentang ID record yang sebelumnya dipilih dari tabel. Fungsi ini memastikan semua input kembali ke kondisi awal, memudahkan pengguna untuk memasukkan data baru atau menghindari kesalahan input.

```

186 # Inisialisasi database
187 create_database()
188
189 # Membuat GUI dengan Tkinter
190 root = Tk()
191 root.title("Prediksi Fakultas Siswa")
192
193 # Variabel tkinter
194 nama_var = StringVar()
195 biologi_var = StringVar()
196 fisika_var = StringVar()
197 inggris_var = StringVar()
198 selected_record_id = StringVar()

```

`create_database()` : Fungsi ini (diasumsikan) digunakan untuk membuat database atau tabel yang diperlukan untuk menyimpan data siswa, jika belum ada.

`root = Tk()` : Membuat objek utama Tkinter bernama `root`, yang berfungsi sebagai jendela utama aplikasi GUI.

`root.title("Prediksi Fakultas Siswa")` : Mengatur judul jendela utama menjadi "Prediksi Fakultas Siswa".

`nama_var = StringVar()` : Membuat variabel Tkinter `nama_var` bertipe `StringVar`. Variabel ini digunakan untuk mengelola data input nama siswa.

`biologi_var = StringVar()` : Membuat variabel Tkinter `biologi_var` bertipe `StringVar`. Digunakan untuk mengelola data input nilai Biologi siswa.

`fisika_var = StringVar()` : Membuat variabel Tkinter `fisika_var` bertipe `StringVar`. Digunakan untuk mengelola data input nilai Fisika siswa.

`inggris_var = StringVar()` : Membuat variabel Tkinter `inggris_var` bertipe `StringVar`. Digunakan untuk mengelola data input nilai Bahasa Inggris siswa.

`selected_record_id = StringVar()` : Membuat variabel Tkinter `selected_record_id` bertipe `StringVar`. Digunakan untuk menyimpan ID data yang dipilih dari tabel (treeview) untuk keperluan pembaruan atau penghapusan data.

```

200 # Elemen GUI
201 Label(root, text="Nama Siswa").grid(row=0, column=0, padx=10, pady=5)
202 Entry(root, textvariable=nama_var).grid(row=0, column=1, padx=10, pady=5)
203
204 Label(root, text="Nilai Biologi").grid(row=1, column=0, padx=10, pady=5)
205 Entry(root, textvariable=biologi_var).grid(row=1, column=1, padx=10, pady=5)
206
207 Label(root, text="Nilai Fisika").grid(row=2, column=0, padx=10, pady=5)
208 Entry(root, textvariable=fisika_var).grid(row=2, column=1, padx=10, pady=5)
209
210 Label(root, text="Nilai Inggris").grid(row=3, column=0, padx=10, pady=5)
211 Entry(root, textvariable=inggris_var).grid(row=3, column=1, padx=10, pady=5)
212
213 Button(root, text="Add", command=submit).grid(row=4, column=0, pady=10)
214 Button(root, text="Update", command=update).grid(row=4, column=1, pady=10)
215 Button(root, text="Delete", command=delete).grid(row=4, column=2, pady=10)
216

```

Labels dan Entries: Membuat label dan kotak input (entry) untuk memasukkan Nama Siswa, Nilai Biologi, Nilai Fisika, dan Nilai Inggris.

Menggunakan grid layout untuk menata posisi label di kolom 0 dan entry di kolom 1.

Tombol Aksi:

Add: Menjalankan fungsi submit() untuk menambah data.

Update: Menjalankan fungsi update() untuk memperbarui data.

Delete: Menjalankan fungsi delete() untuk menghapus data.

Tombol ditata di baris ke-4 dengan masing-masing tombol di kolom terpisah.

```

217 # Tabel untuk menampilkan data
218 columns = ("id", "nama_siswa", "biologi", "fisika", "inggris", "prediksi_fakultas")
219 tree = ttk.Treeview(root, columns=columns, show='headings')
220

```

columns: Membuat tuple yang berisi nama kolom:

("id", "nama\_siswa", "biologi", "fisika", "inggris", "prediksi\_fakultas").

Kolom ini digunakan untuk mendefinisikan struktur data di Treeview.

tree = ttk.Treeview(root, columns=columns, show='headings'): Membuat widget Treeview dari modul ttk di Tkinter untuk menampilkan data dalam bentuk tabel.



columns=columns: Menentukan kolom-kolom yang akan ditampilkan di tabel.

show='headings': Menampilkan hanya judul kolom tanpa kolom tambahan (seperti kolom root).

```
221 # Menambahkan scrollbar
222 scrollbar = Scrollbar(root, orient='vertical', command=tree.yview)
223 tree.configure(yscroll=scrollbar.set)
224 scrollbar.grid(row=5, column=3, sticky='ns', pady=10)
225
226 for col in columns:
227     tree.heading(col, text=col.capitalize())
228     tree.column(col, anchor='center')
229
230 tree.grid(row=5, column=0, columnspan=3, padx=10, pady=10)
231
```

scrollbar = Scrollbar(root, orient='vertical', command=tree.yview): Membuat Scrollbar vertikal yang akan digunakan untuk menggulir data di Treeview.

orient='vertical' menentukan scrollbar berada di vertikal.

command=tree.yview menghubungkan scrollbar dengan Treeview agar saat digulir, Treeview juga ikut bergulir.

tree.configure(yscroll=scrollbar.set): Menghubungkan scrollbar dengan Treeview, sehingga saat scrollbar digulir, posisi tampilan Treeview ikut berubah.

scrollbar.grid(row=5, column=3, sticky='ns', pady=10): Menempatkan scrollbar pada baris ke-5 dan kolom ke-3 dalam grid layout.

sticky='ns' memastikan scrollbar vertikal mengisi ruang secara vertikal (north-south). pady=10 memberikan padding vertikal.

for col in columns: Melakukan iterasi untuk setiap kolom yang ada di columns dan mengatur heading dan pengaturan kolom Treeview.

tree.heading(col, text=col.capitalize()): Mengatur teks di heading (judul kolom) untuk setiap kolom menjadi kapitalisasi dari nama kolom (misalnya, id menjadi Id).

`tree.column(col, anchor='center')`: Menyelaraskan konten kolom agar berada di tengah.

`tree.grid(row=5, column=0, columnspan=3, padx=10, pady=10)`: Menempatkan Treeview di baris ke-5, kolom ke-0, dan membentang di 3 kolom. `padx=10, pady=10` memberi padding horizontal dan vertikal pada Treeview.

```
232 # Event untuk memilih data dari tabel
233 tree.bind('<ButtonRelease-1>', fill_inputs_from_table)
234
235 # Mengisi tabel dengan data
236 populate_table()
237
238 # Menjalankan Aplikasi
239 root.mainloop()
240 |
```

`tree.bind('<ButtonRelease-1>', fill_inputs_from_table)`: Menambahkan event binding pada Treeview (`tree`).

`<ButtonRelease-1>` mengikatkan event klik kiri mouse yang dilepas pada Treeview.

Ketika event ini terjadi, fungsi `fill_inputs_from_table` dipanggil, yang akan mengisi form input dengan data yang dipilih dari baris tabel.

`populate_table()`: Memanggil fungsi `populate_table()` untuk mengisi Treeview dengan data yang diambil dari database atau sumber data lainnya.

`root.mainloop()`: Menjalankan loop utama aplikasi Tkinter.

Fungsi ini memastikan aplikasi GUI tetap berjalan, menunggu interaksi dari pengguna dan memperbarui antarmuka pengguna sesuai dengan aksi yang dilakukan.

