

**LAPORAN PRAKTIKUM
PEMROGRAMAN BERORIENTASI OBJEK (PBO) – [TUGAS 5]**



Disusun Oleh

Andryano Shevchenko Limbong 123140205

**PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS TEKNOLOGI INFORMASI
INSTITUT TEKNOLOGI SUMATERA
2025**

Soal nomor 1

a. input soal

Challenge Praktikum: Membuat Aplikasi

Catatan Harian dengan Tkinter

Latar Belakang

Dalam praktikum ini, Anda akan ditantang untuk membuat Aplikasi Catatan Harian sederhana menggunakan Tkinter. Aplikasi ini memungkinkan pengguna untuk menulis, menyimpan, dan mengelola catatan harian dengan antarmuka grafis yang intuitif. Tantangan ini dirancang untuk menguji pemahaman Anda tentang widget Tkinter, manajemen tata letak, penanganan event, dan interaksi dasar dengan file.

Deskripsi Tantangan

Buatlah aplikasi GUI dengan Tkinter yang memiliki fitur berikut:

1. Input Catatan:

- Pengguna dapat memasukkan judul catatan di Entry .
- Pengguna dapat menulis isi catatan di Text (area teks multibaris).
- Tombol "Tambah Catatan" untuk menyimpan catatan ke daftar.

2. Tampilan Daftar Catatan:

- Tampilkan judul catatan dalam Listbox .
- Ketika pengguna mengklik judul di Listbox , isi catatan ditampilkan di area Text (read-only mode).
- Sertakan Scrollbar untuk Listbox agar dapat menampung banyak catatan.

3. Hapus Catatan:

- Tombol "Hapus Catatan" untuk menghapus catatan yang dipilih di Listbox .
- Tampilkan konfirmasi menggunakan messagebox sebelum menghapus.

4. Validasi Input:

- Pastikan judul dan isi catatan tidak kosong sebelum menyimpan.
- Tampilkan pesan error menggunakan messagebox jika input tidak valid.

5. Menu Bar:

- Tambahkan Menu dengan opsi:
 - File: Keluar (menutup aplikasi).
 - Bantuan: Tentang (tampilkan dialog dengan nama dan versi aplikasi).

Persyaratan Teknis

- Gunakan Python 3 dan Tkinter.
- Gunakan Grid untuk tata letak utama agar rapi, atau Pack untuk bagian sederhana.
- Simpan catatan dalam struktur data sederhana seperti daftar (list) atau kamus (dict) di memori (tidak perlu penyimpanan file untuk menyederhanakan).
- Gunakan messagebox untuk konfirmasi dan pesan error.
- Aplikasi harus memiliki antarmuka yang jelas dan mudah digunakan.
- Kode harus terstruktur dengan fungsi untuk modularitas (misalnya, fungsi untuk menambah, menghapus, dan menampilkan catatan).
- Tambahkan komentar untuk menjelaskan logika utama.

Bonus (Opsional)

1. Penyimpanan File:

- Simpan catatan ke file teks atau JSON saat aplikasi ditutup dan muat kembali saat dibuka.

2. Waktu Catatan:

- Tambahkan tanggal/waktu pembuatan catatan (gunakan modul datetime) dan tampilkan di samping judul di Listbox .

3. Edit Catatan:

- Tambahkan tombol "Edit Catatan" untuk memperbarui judul atau isi catatan yang dipilih.

Contoh Struktur Antarmuka

Berikut gambaran tata letak antarmuka yang disarankan:

File Bantuan	

Judul: [_____]	
[Tambah Catatan] [Hapus Catatan]	

Daftar Catatan (Listbox) Isi Catatan (Text)	
- Catatan 1 [Teks catatan...]	
- Catatan 2	
[Scrollbar]	

Panduan Implementasi

1. Inisialisasi Jendela:

- Buat jendela utama dengan Tk() dan atur judul (misalnya, "Catatan Harian").
- Tambahkan Menu dengan opsi File dan Bantuan.

2. Widget Utama:

- Gunakan Entry untuk input judul.
- Gunakan Text untuk input isi catatan.
- Gunakan Listbox dengan Scrollbar untuk daftar judul.
- Tambahkan dua Button untuk "Tambah Catatan" dan "Hapus Catatan".

3. Fungsi Inti:

- Tambah Catatan: Validasi input, simpan judul dan isi ke daftar/kamus, tambahkan judul ke Listbox .
- Tampilkan Catatan: Saat judul di Listbox diklik (gunakan event), tampilkan isi di Text (set state="disabled" untuk read-only).
- Hapus Catatan: Hapus catatan yang dipilih dari daftar dan Listbox setelah konfirmasi.

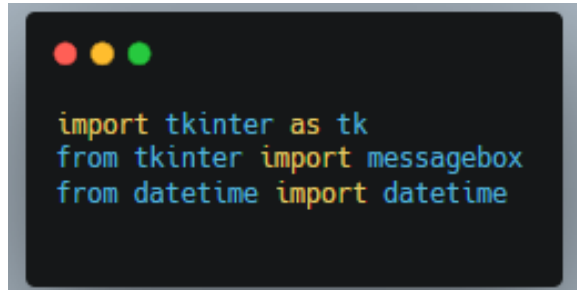
4. Validasi:

- Gunakan `messagebox.showerror` untuk input kosong.
- Gunakan `messagebox.askyesno` untuk konfirmasi penghapusan.

b. Penjelasan

Berikut adalah penjelasan dari kode tersebut:

1. Library tkinter

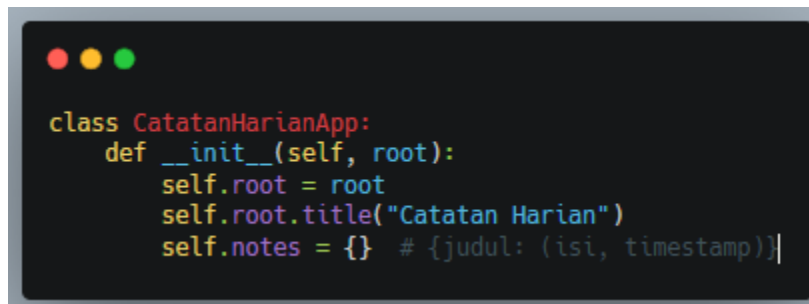


```
import tkinter as tk
from tkinter import messagebox
from datetime import datetime
```

`import tkinter as tk` digunakan sebagai library yang dapat membantu pengerjaan program GUI, yaitu tugas Challenge Praktikum: Membuat Aplikasi Catatan Harian dengan Tkinter.

Dengan library tkinter kita yaitu, `from tkinter import message box` dan `from date time import datetime` sehingga programmer dapat mengakses banyak fitur tambahan GUI.

2. Membuat Class



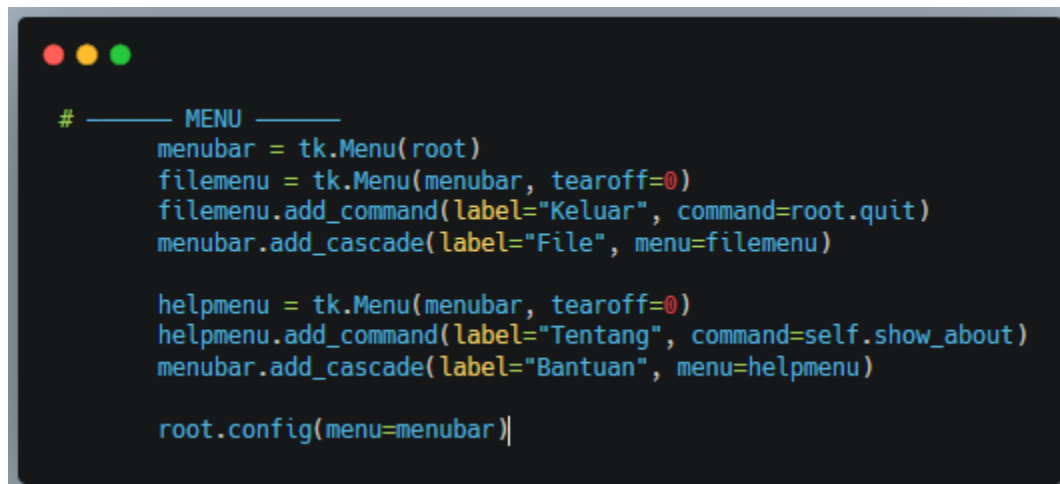
```
class CatatanHarianApp:
    def __init__(self, root):
        self.root = root
        self.root.title("Catatan Harian")
        self.notes = {} # {judul: (isi, timestamp)}
```

Membuat Class `CatatanHarianApp` untuk enkapsulasi fungsi dan elemen dari aplikasi.

- `self.root = root;`
Fungsi: Menyimpan objek `Tk()` yang dikirim saat inisialisasi ke dalam atribut `self.root`.

- `self.root.title("Catatan Harian")`
Fungsi: Mengatur judul (title bar) dari jendela utama aplikasi menjadi "Catatan Harian".
- `self.notes = {}` # Menyimpan catatan sebagai dictionary {judul: isi}
Fungsi: Membuat dictionary kosong untuk menyimpan catatan.

3. Membuat Menu



```
# ----- MENU -----
menubar = tk.Menu(root)
filemenu = tk.Menu(menubar, tearoff=0)
filemenu.add_command(label="Keluar", command=root.quit)
menubar.add_cascade(label="File", menu=filemenu)

helpmenu = tk.Menu(menubar, tearoff=0)
helpmenu.add_command(label="Tentang", command=self.show_about)
menubar.add_cascade(label="Bantuan", menu=helpmenu)

root.config(menu=menubar)
```

- A. Membuat Menu Bar Utama (menubar)
 - Menu bar adalah baris menu horizontal di bagian atas jendela GUI.
 - Objek ini akan menampung berbagai menu seperti "File", "Edit", "Help", dll.
- B. Menambahkan Submenu "File" (filemenu)
 - Menu ini akan muncul sebagai dropdown saat pengguna mengklik "File" pada menu bar.
 - `tearoff=0` mencegah submenu bisa dipisah jadi window sendiri (default Tkinter feature).
- C. Menambahkan Item Menu "Keluar"
 - Item ini adalah opsi di dalam menu "File".
 - Ketika diklik, menjalankan perintah `root.quit`, yaitu menutup aplikasi.

D. Menambahkan "File" ke Menu Bar

- Menggabungkan filemenu ke dalam menubar dengan label "File".
- Artinya, saat pengguna klik "File", akan muncul dropdown berisi "Keluar".

E. Membuat Submenu "Bantuan" (helpmenu)

- Ini adalah menu dropdown yang akan muncul ketika pengguna mengklik menu "Bantuan" di bar atas.
- `tearoff=0` digunakan agar submenu tidak bisa dipisahkan ke window baru.

F. Menambahkan Item Menu "Tentang"

- Menambahkan pilihan dalam menu "Bantuan" bernama "Tentang".
- Jika diklik, akan menjalankan fungsi `self.show_about`, yaitu menampilkan kotak informasi tentang aplikasi (biasanya pakai `messagebox`).

G. Menambahkan "Bantuan" ke Menu Bar

- Menu "Bantuan" (berserta item "Tentang" di dalamnya) ditambahkan ke menubar.

H. Mengaktifkan Menu Bar di Window Utama

- `root.config(menu=menubar)` menetapkan bahwa menu bar menubar digunakan oleh jendela utama (`root`).
- Ini menampilkan menu bar yang sudah dibuat dan dikonfigurasi (termasuk menu "File" dan "Bantuan").

4. Membuat Widget

```
# ----- WIDGETS -----
# Judul
tk.Label(root, text="Judul:").grid(row=0, column=0, sticky="w", padx=5, pady=5)
self.entry_judul = tk.Entry(root)
self.entry_judul.grid(row=0, column=1, columnspan=3, sticky="we", padx=5)

# Tombol
btn_add = tk.Button(root, text="Tambah Catatan", command=self.tambah_catatan)
btn_add.grid(row=0, column=4, padx=5)
btn_delete = tk.Button(root, text="Hapus Catatan", command=self.hapus_catatan)
btn_delete.grid(row=0, column=5, padx=5)
```

A. Menampilkan Label “Judul:”

- Memberikan teks penjas di sebelah kiri untuk inputan judul catatan.

B. Membuat Kolom Input Judul (Entry)

- Tempat user mengetik judul catatan yang ingin ditambahkan.

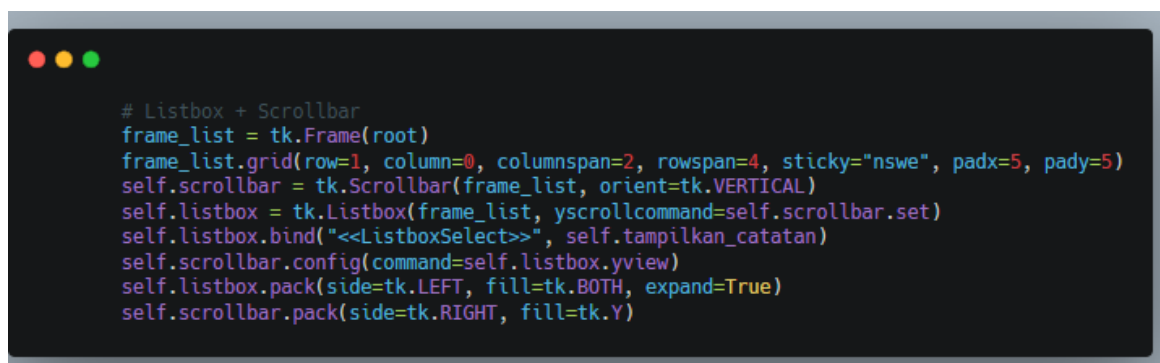
C. Tombol “Tambah Catatan”

- Ketika diklik, akan menjalankan fungsi `self.tambah_catatan` untuk menyimpan catatan yang sedang ditulis.

D. Tombol “Hapus Catatan”

- Ketika diklik, akan menjalankan fungsi `self.hapus_catatan` untuk menghapus catatan yang dipilih di daftar.

5. Membuat Listbox + Scrollbar



```
# Listbox + Scrollbar
frame_list = tk.Frame(root)
frame_list.grid(row=1, column=0, columnspan=2, rowspan=4, sticky="nswe", padx=5, pady=5)
self.scrollbar = tk.Scrollbar(frame_list, orient=tk.VERTICAL)
self.listbox = tk.Listbox(frame_list, yscrollcommand=self.scrollbar.set)
self.listbox.bind("<<ListboxSelect>>", self.tampilkan_catatan)
self.scrollbar.config(command=self.listbox.yview)
self.listbox.pack(side=tk.LEFT, fill=tk.BOTH, expand=True)
self.scrollbar.pack(side=tk.RIGHT, fill=tk.Y)
```

A. Membuat Frame sebagai wadah listbox dan scrollbar

- Digunakan untuk menyatukan tampilan Listbox dan Scrollbar agar mudah diatur dalam satu area.

B. Menampilkan Listbox

- Menampilkan daftar catatan berdasarkan judul + timestamp.
- User bisa memilih salah satu catatan dari daftar ini.

C. Menambahkan Scrollbar (vertikal)

- Memungkinkan user menggulir jika jumlah catatan yang tampil melebihi tinggi listbox.

D. Menghubungkan Listbox dengan Scrollbar

- `yscrollcommand=self.scrollbar.set` dan `command=self.listbox.yview` membuat scrollbar dan listbox saling sinkron.

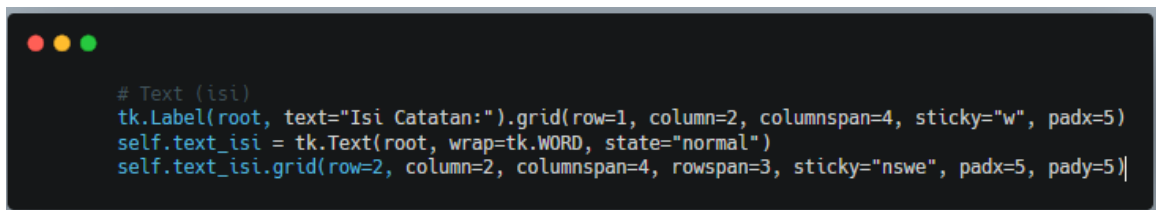
E. Event Binding pada Listbox

- Ketika item dipilih di Listbox, fungsi `self.tampilkan_catatan` akan dijalankan untuk menampilkan isi catatan tersebut.

F. Layouting dengan `.pack()`

- Listbox ditempatkan di kiri dan memenuhi ruang (`fill=tk.BOTH, expand=True`).
- Scrollbar ditempatkan di kanan dan memanjang ke bawah (`fill=tk.Y`).

6. Membuat Text (isi)

A screenshot of a code editor with a dark background and light-colored text. The code is in Python and defines a text widget. It includes comments and Tkinter widget creation and grid placement code.

```
# Text (isi)
tk.Label(root, text="Isi Catatan:").grid(row=1, column=2, columnspan=4, sticky="w", padx=5)
self.text_isi = tk.Text(root, wrap=tk.WORD, state="normal")
self.text_isi.grid(row=2, column=2, columnspan=4, rowspan=3, sticky="nswe", padx=5, pady=5)
```

A. Label "Isi Catatan:"

- Memberikan penanda atau judul di atas area teks, agar user tahu bahwa bagian ini digunakan untuk menulis isi catatan.

B. Widget Text (area teks)

- Tempat user menuliskan isi dari catatan harian.
- Mendukung teks panjang, baris baru, dan pengaturan pembungkus kata (`wrap=tk.WORD`) agar teks otomatis pindah baris per kata, bukan per karakter.

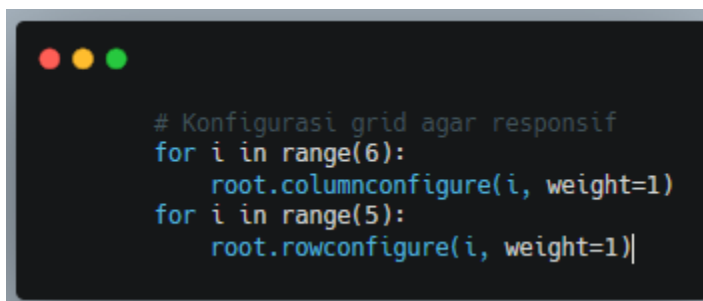
C. Properti state="normal"

- Menandakan area teks bisa diketik secara default.
- Nanti saat catatan ditampilkan ulang (misalnya setelah dipilih dari listbox), state bisa diubah ke "disabled" agar tidak bisa diedit.

D. Layout Grid

- Ditempatkan mulai dari baris ke-2 hingga baris ke-4 (rowspan=3), dan kolom ke-2 sampai ke-5 (columnspan=4) agar area teks cukup luas.
- sticky="nswe" agar area teks mengikuti ukuran window (responsif).

7. Konfigurasi grid agar responsif



```
# Konfigurasi grid agar responsif
for i in range(6):
    root.columnconfigure(i, weight=1)
for i in range(5):
    root.rowconfigure(i, weight=1)
```

A. root.columnconfigure(i, weight=1) untuk 6 kolom:

- Memberikan "berat" (weight) pada setiap kolom dalam grid layout.
- Artinya, semua 6 kolom (indeks 0–5) ikut melebar saat ukuran window diperluas secara horizontal.

B. root.rowconfigure(i, weight=1) untuk 5 baris:

- Memberikan "berat" pada setiap baris (indeks 0–4) agar juga ikut menyesuaikan saat tinggi window berubah.
- Membuat tampilan lebih adaptif terhadap perubahan ukuran jendela secara vertikal.

8. Tombol About

```
def show_about(self):  
    messagebox.showinfo("Tentang", "Catatan Harian v1.0\noleh [Andryano Shevchenko Limbong]")
```

Fungsi ini menampilkan **kotak pesan (message box)** berisi informasi tentang aplikasi.

9. Fungsi tombol Tambah Catatan

```
def tambah_catatan(self):  
    judul = self.entry_judul.get().strip()  
    isi = self.text_isi.get("1.0", tk.END).strip()  
    if not judul or not isi:  
        messagebox.showerror("Error", "Judul dan isi catatan tidak boleh kosong.")  
        return  
    timestamp = datetime.now().strftime("%Y-%m-%d %H:%M")  
    display_judul = f"{timestamp} - {judul}"  
    # Simpan dalam dict  
    self.notes[display_judul] = isi  
    # Tambah ke Listbox  
    self.listbox.insert(tk.END, display_judul)  
    # Reset input  
    self.entry_judul.delete(0, tk.END)  
    self.text_isi.delete("1.0", tk.END)
```

A. Mengambil Input dari Pengguna

- Mengambil judul dari Entry dan isi catatan dari Text.
- `.strip()` digunakan untuk menghapus spasi kosong di awal/akhir.

B. Validasi Input

- Mengecek apakah judul atau isi kosong.
- Jika kosong, tampilkan popup error dan hentikan proses (`return`).

C. Membuat Timestamp

- Membuat cap waktu (timestamp) dalam format YYYY-MM-DD HH:MM menggunakan `datetime.now()` agar catatan punya waktu tercatat.

D. Membuat Judul Tampilan

- Menggabungkan timestamp dan judul menjadi satu string seperti:
"2025-04-23 14:30 — Judul Catatan"

E. Menyimpan Catatan

- Catatan disimpan ke dalam dictionary self.notes dengan key = display_judul, dan value = isi.

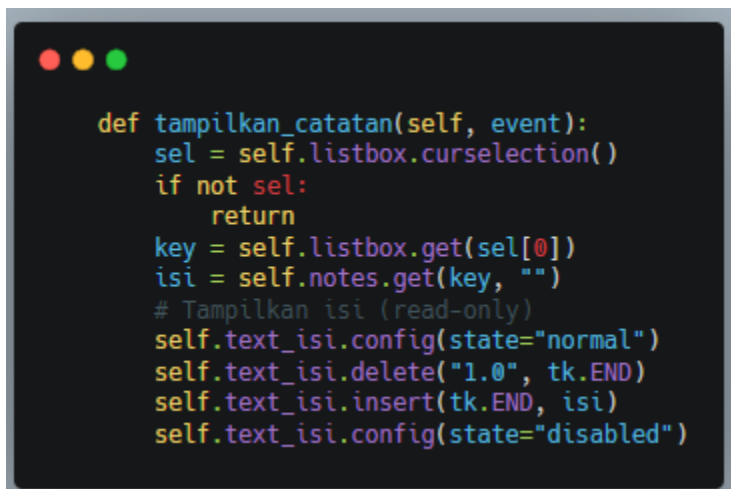
F. Menambahkan ke Listbox

- Menambahkan display_judul ke Listbox agar tampil di daftar catatan yang bisa dipilih.

G. Mengosongkan Inputan

- Setelah ditambahkan, Entry judul dan Text isi dikosongkan kembali agar siap untuk input baru.

10. Fungsi tombol Tampilkan Catatan

A screenshot of a code editor with a dark background and light-colored text. The code is a Python function named 'tampilkan_catatan' that takes 'self' and 'event' as arguments. It uses 'self.listbox.curselection()' to get the selected item index. If no item is selected, it returns. Otherwise, it gets the key from the listbox and the corresponding value from 'self.notes'. It then configures a text widget 'self.text_isi' to be read-only, deletes its current content, and inserts the retrieved note text. Finally, it configures the text widget to be disabled.

```
def tampilkan_catatan(self, event):  
    sel = self.listbox.curselection()  
    if not sel:  
        return  
    key = self.listbox.get(sel[0])  
    isi = self.notes.get(key, "")  
    # Tampilkan isi (read-only)  
    self.text_isi.config(state="normal")  
    self.text_isi.delete("1.0", tk.END)  
    self.text_isi.insert(tk.END, isi)  
    self.text_isi.config(state="disabled")
```

A. Mendapatkan Pilihan yang Dipilih di Listbox

- sel = self.listbox.curselection() mendapatkan indeks item yang dipilih di Listbox.
- Jika tidak ada item yang dipilih (if not sel), fungsi ini akan berhenti dan tidak melanjutkan.

B. Mengambil Key dari Listbox

- `key = self.listbox.get(sel[0])` mengambil judul catatan yang dipilih dari Listbox berdasarkan indeks yang dipilih oleh user.

C. Mencari Isi Catatan dari Dictionary

- `isi = self.notes.get(key, "")` mencari isi catatan di dictionary `self.notes` dengan menggunakan key (judul catatan).
- Jika tidak ditemukan, akan mengembalikan string kosong (`""`).

D. Menampilkan Isi Catatan dalam Text

- `self.text_isi.config(state="normal")` mengubah state area teks menjadi normal agar bisa diedit (walaupun pada akhirnya di-set menjadi read-only).
- `self.text_isi.delete("1.0", tk.END)` menghapus isi teks lama (jika ada).
- `self.text_isi.insert(tk.END, isi)` menambahkan isi catatan yang ditemukan ke dalam area teks.

E. Menonaktifkan Edit Area Teks (Read-Only)

- Setelah isi catatan ditampilkan, area teks diubah kembali ke state "disabled" agar tidak bisa diubah oleh user secara langsung.

11. Fungsi Tombol Hapus Catatan

```
def hapus_catatan(self):
    sel = self.listbox.curselection()
    if not sel:
        messagebox.showerror("Error", "Pilih catatan yang akan dihapus.")
        return
    key = self.listbox.get(sel[0])
    if messagebox.askyesno("Konfirmasi", f"Hapus catatan:\n{key}?"):
        # Hapus dari data dan UI
        del self.notes[key]
        self.listbox.delete(sel[0])
        # Bersihkan area teks
        self.text_isi.config(state="normal")
        self.text_isi.delete("1.0", tk.END)
        self.text_isi.config(state="normal")
```

A. Mengecek Item yang Dipilih di Listbox

- `sel = self.listbox.curselection()` mendapatkan indeks item yang dipilih di Listbox.
- Jika tidak ada item yang dipilih (if not sel), maka muncul pesan error agar user tahu bahwa mereka harus memilih catatan untuk dihapus.

B. Menampilkan Konfirmasi Penghapusan

- `key = self.listbox.get(sel[0])` mendapatkan judul catatan yang dipilih.
- `messagebox.askyesno("Konfirmasi", f"Hapus catatan:\n{key}?")` menampilkan dialog konfirmasi dengan pertanyaan "Apakah Anda yakin ingin menghapus catatan ini?".
- Jika user memilih "Yes", proses penghapusan dilanjutkan.

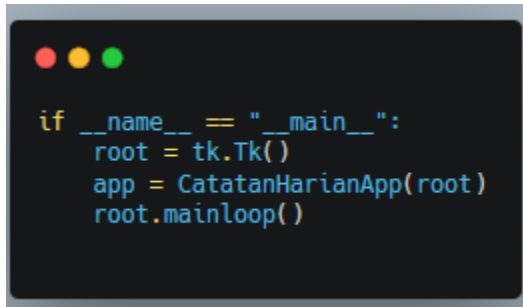
C. Menghapus Catatan dari Dictionary dan Listbox

- `del self.notes[key]` menghapus catatan yang dipilih dari dictionary `self.notes`.
- `self.listbox.delete(sel[0])` menghapus judul catatan yang dipilih dari tampilan Listbox.

D. Mengosongkan Area Teks

- Setelah catatan dihapus, area teks yang menampilkan isi catatan dikosongkan dengan:
 - `self.text_isi.config(state="normal")` mengubah state area teks menjadi dapat diedit sementara.
 - `self.text_isi.delete("1.0", tk.END)` menghapus semua teks yang ada di area teks.
 - `self.text_isi.config(state="disabled")` mengembalikan area teks ke read-only agar tidak bisa diubah.

12. Menjalankan Aplikasi GUI



A. Menjalankan Aplikasi saat Langsung Dijalankan:

- Baris `if __name__ == '__main__':` digunakan untuk memastikan bahwa kode di bawahnya hanya dijalankan ketika skrip ini dijalankan langsung (bukan diimpor sebagai modul dalam skrip lain).
- Ini adalah praktik umum di Python untuk memastikan program utama hanya dijalankan sekali.

B. Membuat Objek Tkinter (root):

- `root = tk.Tk()` membuat window utama untuk aplikasi GUI dengan Tkinter. Ini adalah jendela kosong yang digunakan sebagai wadah untuk semua widget aplikasi (seperti tombol, label, dll).

C. Membuat dan Menjalankan Aplikasi CatatanHarianApp:

- `app = CatatanHarianApp(root)` membuat instansi dari kelas `CatatanHarianApp` yang telah dibuat sebelumnya, dan menghubungkannya dengan window Tkinter (`root`).
- Ini memungkinkan semua elemen GUI, logika, dan interaktivitas yang sudah ditentukan di dalam kelas `CatatanHarianApp` untuk diinisialisasi dan dijalankan.

D. Memulai Mainloop Tkinter:

- `root.mainloop()` memulai main event loop Tkinter. Ini adalah loop tak terbatas yang menunggu interaksi pengguna, seperti klik tombol atau input, dan memastikan aplikasi tetap berjalan hingga window ditutup.
- Tanpa `mainloop()`, aplikasi tidak akan dapat merespon event atau tampil dengan benar.

c. Source Code

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

class CatatanHarianApp:
    def __init__(self, root):
        self.root = root
        self.root.title("Catatan Harian")
        self.notes = {} # {judul: (isi, timestamp)}

        # ----- MENU -----
        menubar = tk.Menu(root)
        filemenu = tk.Menu(menubar, tearoff=0)
        filemenu.add_command(label="Keluar", command=root.quit)
        menubar.add_cascade(label="File", menu=filemenu)

        helpmenu = tk.Menu(menubar, tearoff=0)
        helpmenu.add_command(label="Tentang", command=self.show_about)
        menubar.add_cascade(label="Bantuan", menu=helpmenu)

        root.config(menu=menubar)

        # ----- WIDGETS -----
        # Judul
        tk.Label(root, text="Judul:").grid(row=0, column=0, sticky="w", padx=5, pady=5)
        self.entry_judul = tk.Entry(root)
        self.entry_judul.grid(row=0, column=1, columnspan=3, sticky="we", padx=5)

        # Tombol
        btn_add = tk.Button(root, text="Tambah Catatan", command=self.tambah_catatan)
        btn_add.grid(row=0, column=4, padx=5)
        btn_delete = tk.Button(root, text="Hapus Catatan", command=self.hapus_catatan)
        btn_delete.grid(row=0, column=5, padx=5)

        # Listbox + Scrollbar
        frame_list = tk.Frame(root)
        frame_list.grid(row=1, column=0, columnspan=2, rowspan=4, sticky="nswe", padx=5, pady=5)
        self.scrollbar = tk.Scrollbar(frame_list, orient=tk.VERTICAL)
        self.listbox = tk.Listbox(frame_list, yscrollcommand=self.scrollbar.set)
        self.listbox.bind("<<ListboxSelect>>", self.tampilkan_catatan)
        self.scrollbar.config(command=self.listbox.yview)
        self.listbox.pack(side=tk.LEFT, fill=tk.BOTH, expand=True)
        self.scrollbar.pack(side=tk.RIGHT, fill=tk.Y)

        # Text (isi)
        tk.Label(root, text="Isi Catatan:").grid(row=1, column=2, columnspan=4, sticky="w",
        padx=5) self.text_isi = tk.Text(root, wrap=tk.WORD, state="normal")
        self.text_isi.grid(row=2, column=2, columnspan=4, rowspan=3, sticky="nswe", padx=5,
        pady=5)

        # Konfigurasi grid agar responsif
        for i in range(6):
            root.columnconfigure(i, weight=1)
        for i in range(5):
            root.rowconfigure(i, weight=1)

    def show_about(self):
        messagebox.showinfo("Tentang", "Catatan Harian v1.0\noleh [NAndryano Shevchenko Limbong]")

    def tambah_catatan(self):
        judul = self.entry_judul.get().strip()
        isi = self.text_isi.get("1.0", tk.END).strip()
        if not judul or not isi:
            messagebox.showerror("Error", "Judul dan isi catatan tidak boleh kosong.")
            return
        timestamp = datetime.now().strftime("%Y-%m-%d %H:%M")
        display_judul = f"{timestamp} - {judul}"
        # Simpan dalam dict
        self.notes[display_judul] = isi
        # Tambah ke Listbox
        self.listbox.insert(tk.END, display_judul)
        # Reset input
        self.entry_judul.delete(0, tk.END)
        self.text_isi.delete("1.0", tk.END)
```



```

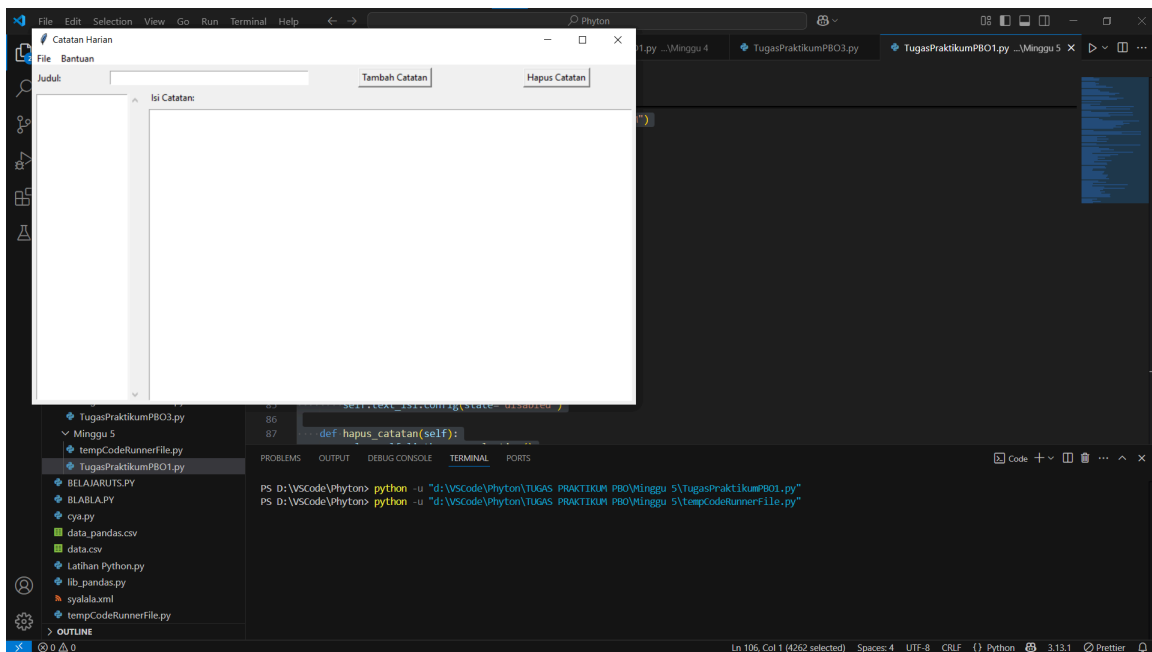
def tampilkan_catatan(self, event):
    sel = self.listbox.curselection()
    if not sel:
        return
    key = self.listbox.get(sel[0])
    isi = self.notes.get(key, "")
    # Tampilkan isi (read-only)
    self.text_isi.config(state="normal")
    self.text_isi.delete("1.0", tk.END)
    self.text_isi.insert(tk.END, isi)
    self.text_isi.config(state="disabled")

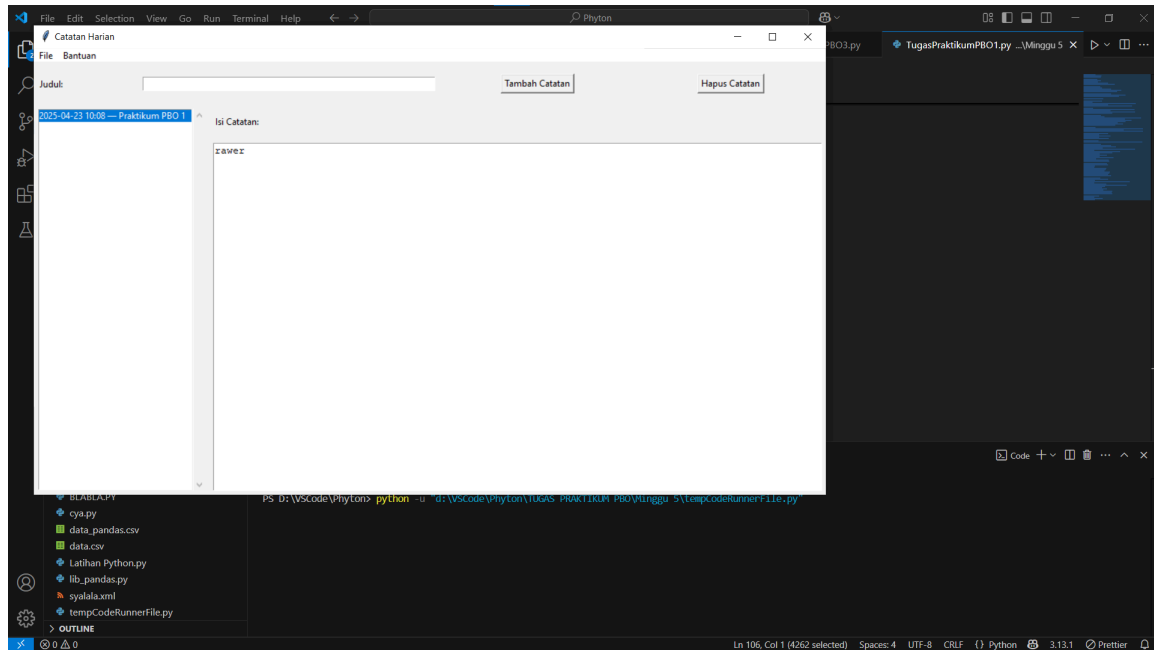
def hapus_catatan(self):
    sel = self.listbox.curselection()
    if not sel:
        messagebox.showerror("Error", "Pilih catatan yang akan dihapus.")
        return
    key = self.listbox.get(sel[0])
    if messagebox.askyesno("Konfirmasi", f"Hapus catatan:\n{key}?"):
        # Hapus dari data dan UI
        del self.notes[key]
        self.listbox.delete(sel[0])
        # Bersihkan area teks
        self.text_isi.config(state="normal")
        self.text_isi.delete("1.0", tk.END)
        self.text_isi.config(state="normal")

if __name__ == "__main__":
    root = tk.Tk()
    app = CatatanHarianApp(root)
    root.mainloop()

```

d. Output Hasil





Lampiran

1. [Link Percakapan LLM \(ChatGPT\):](https://chatgpt.com/share/68086783-11a4-8012-bcff-7b9c9ca7686a)
<https://chatgpt.com/share/68086783-11a4-8012-bcff-7b9c9ca7686a>

(Jika menggunakan LLM atau Referensi website dalam pembuatan laporan, baik untuk generate code ataupun penulisan text silahkan sertakan dokumentasinya bisa berupa link ataupun screenshot percakapan. Jika tidak melampirkan dan ketahuan menggunakan nilai = 0)

Contoh :

1. [Link Percakapan LLM](#)
2. [Web Referensi - DuniaIlkom](#)