

DAY 7

GRAPHICAL USER INTERFACE (GUI)

BY ALFADJRI DWI FADHILAH



Timeline

- 1 TEORI DASAR GUI
- 2 MENGENAL WIDGET DASAR
- 3 MANAJEMEN LAYOUT
- 4 EVENT HANDLING
- 5 DASAR-DASAR ERROR HANDLING

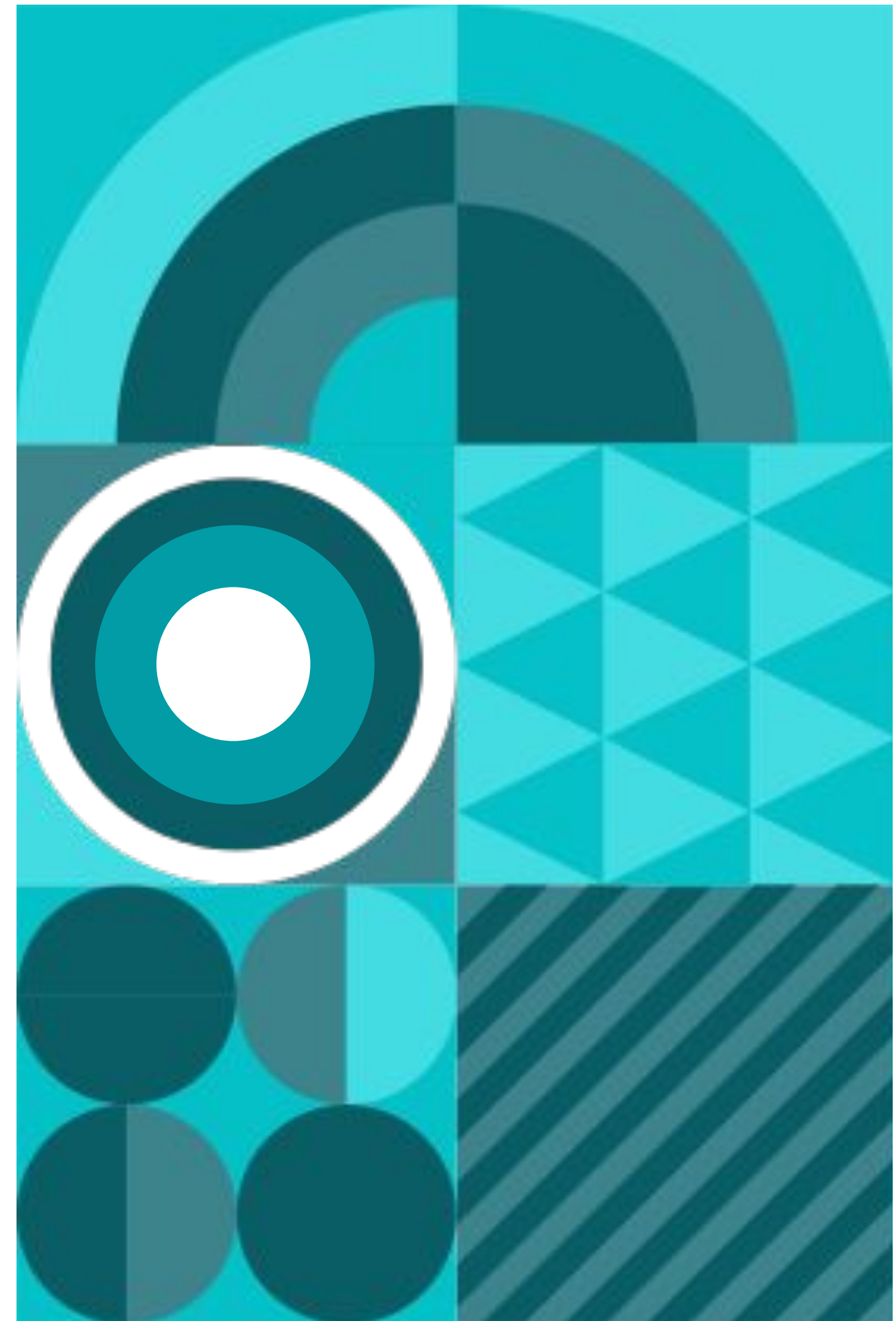


TEORI DASAR GUI




GRAPHICAL USER INTERFACE

Graphical User Interface (GUI) adalah sebuah sistem antarmuka yang memungkinkan pengguna berinteraksi dengan aplikasi melalui elemen-elemen grafis dan visual, seperti jendela , tombol, ikon, dan menu



CLI vs GUI





GUI	CLI
Visuals such as icons, buttons, and menus	Text-based commands
Intuitive and user-friendly	Memorization of commands and syntax
Slower for complex tasks	Commands can perform tasks faster
Limited flexibility	Highly flexible
Limited automation capabilities	Excellent for automation with batch processing
Mistakes are reversible	Mistakes can lead to critical errors
Provides visual feedback	Limited visual feedback
Limited control	High-level control of the system

CLI vs GUI

CLI

Kamu harus tau persis nama menu yang ingin dipesan

Kamu harus menyebutkan pesanan satu per satu secara lisan kepada operator

Interaksi bersifat sekuensial (satu arah) dan berbasis suara/teks (ingatan)

GUI

Kamu bisa melihat daftar menu lengkap dengan gambar-gambar

Kamu cukup menekan tombol “tambah” pada menu yang kamu inginkan

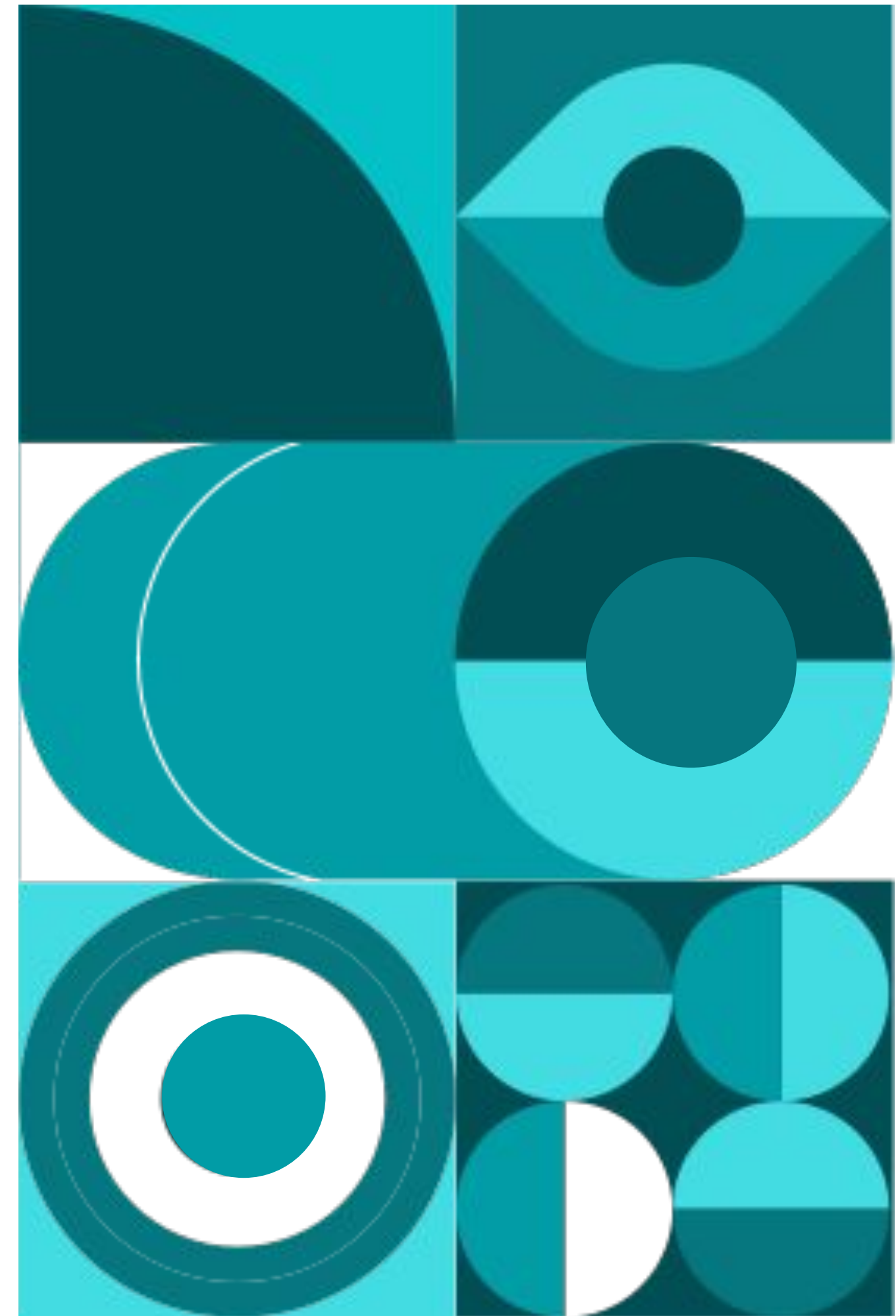
Keranjang belanja langsung ter-update secara visual, dan kamu bisa dengan mudah mengubah pesanan sebelum dikonfirmasi

Manfaat dan Tujuan Utama GUI

- **Intuitif dan mudah dipelajari:** Pengguna bisa langsung menggunakan aplikasi dengan sedikit atau tanpa pelatihan. Pola seperti icon simpan dan hapus sudah umum di mengerti
- **Efisien dan Cepat:** Untuk banyak tugas, interaksi visual seperti drag and drop file jauh lebih cepat daripada mengetik lokasi file secara manual di terminal
- **Memberikan Umpan Balik (feedback):** GUI memberikan response visual yang jelas saat tombol ditekan, ia terlihat saat proses berjalan dan progress bar selalu tahu apa yang sedang terjadi
- **Menjangkau semua pengguna:** Aplikasi dengan GUI yang baik dapat digunakan oleh siapa saja, terlepas dari tingkat keahlian teknis mereka

PRINSIP DASAR GUI

1. WIMP (Windows , Icon , Menus , Pointer)
 - a. Windows (Jendela) adalah area terpisah untuk setiap aplikasi
 - b. Icons (Ikon) adalah gambaran kecil yang mewakili file atau aksi
 - c. Menus (Menu) Daftar opsi atau perintah yang bisa dipilih
 - d. Pointer (Penunjuk) Kursor mouse untuk berinteraksi
2. Paradigma Event Driven Programming adalah konsep Aplikasi dimana pengguna akan “diam” atau menunggu sampai terjadi akses berikutnya atau Event



MENGENAL WIDGET DASAR



MENGENAL WIDGET

DASAR

Widget adalah objek atau komponen visual individual yang kita letakkan di dalam jendela aplikasi untuk membentuk antarmuka pengguna.

Setiap widget memiliki fungsi spesifik

Label


Button

Entry (Teks Box)



LABEL

Label adalah komponen paling dasar yang tujuannya hanya untuk menampilkan teks atau gambar statis. Teks pada label tidak bisa diedit oleh pengguna, sehingga fungsinya murni sebagai pemberi informasi

A screenshot of a 'Widget Demo' window. The window has a title bar with the text 'Widget Demo' and a close button. The main content area is light gray and contains a single line of text: 'This is a label'. The background of the slide features a teal and dark blue geometric pattern with overlapping circles and triangles.

Widget Demo

This is a label

KAPAN DIGUNAKAN

- Memberikan judul pada sebuah jendela atau form
- Menampilkan ikon atau gambar kecil
- Menuliskan instruksi untuk pengguna
- Menampilkan hasil dari sebuah proses atau kalkulasi



CONTOH

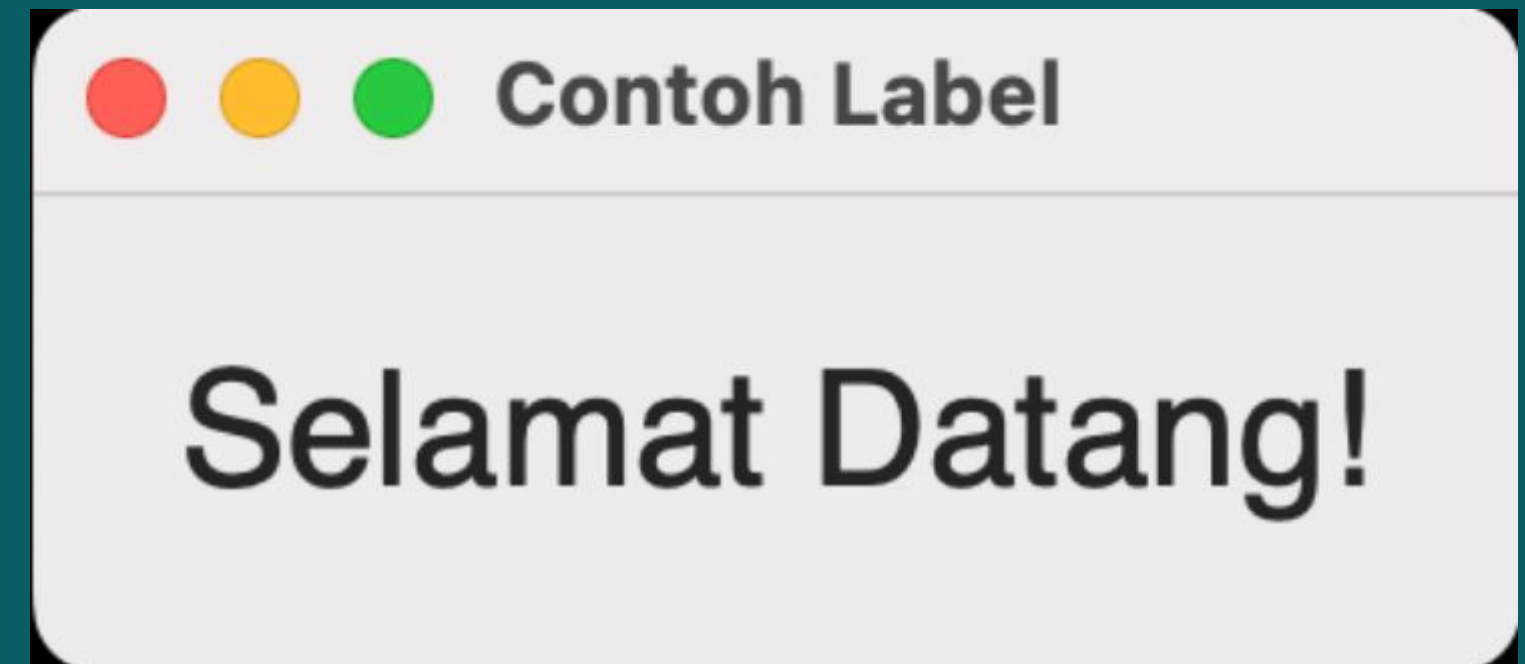
```
import tkinter as tk
from tkinter import ttk

window = tk.Tk()
window.title("Contoh Label")

# Membuat widget Label dengan kustomisasi font
label_salam = ttk.Label(
    window,
    text="Selamat Datang!",
    font=("Helvetica", 18)
)

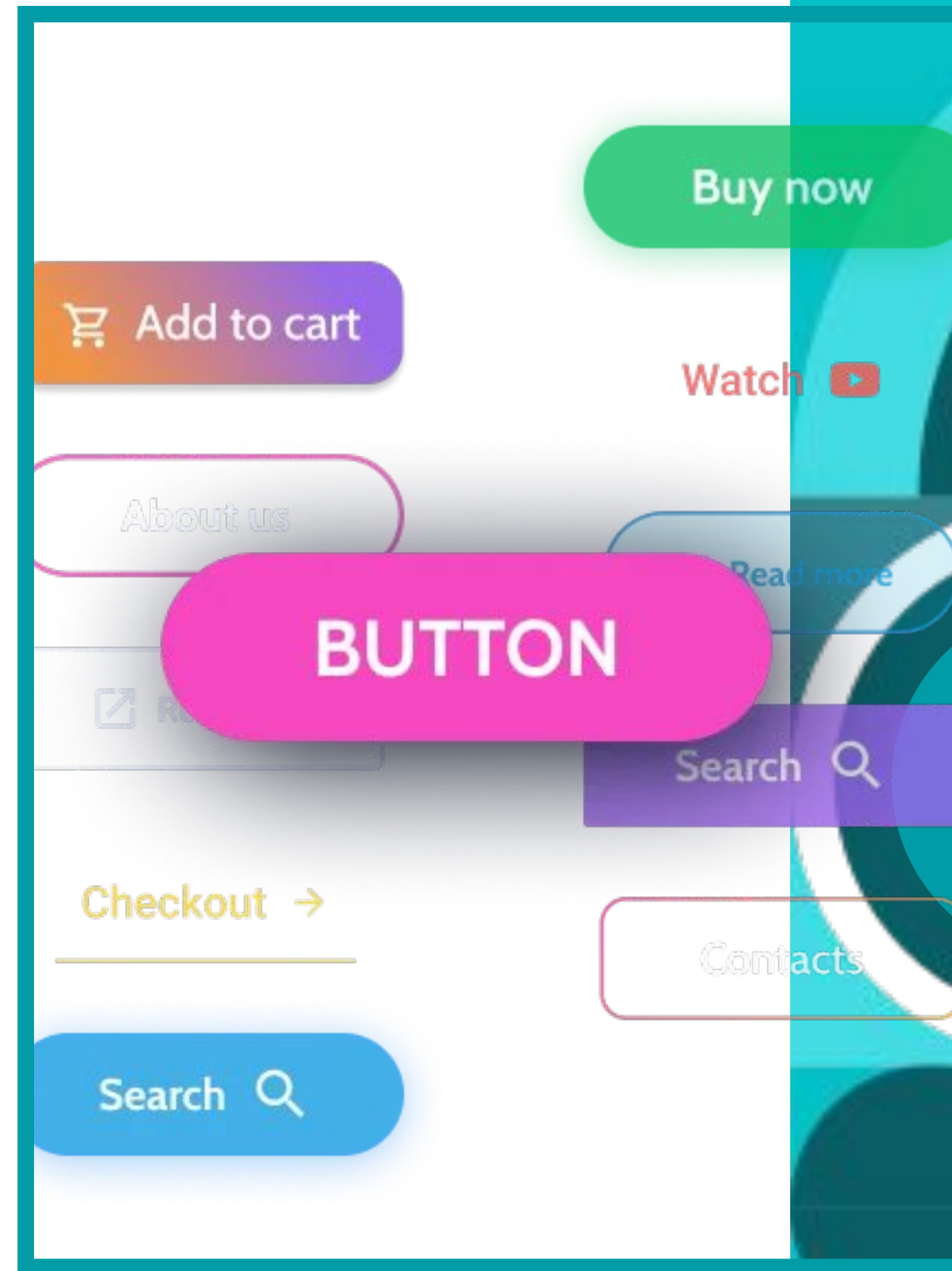
# Menampilkan widget di jendela
label_salam.pack(pady=20, padx=20)

window.mainloop()
```



BUTTON

Button adalah komponen interaktif utama yang memicu sebuah aksi (event) saat di click oleh pengguna.



KAPAN DIGUNAKAN

- Mengirimkan data dari sebuah form (misal: Tombol “Login”, “Simpan”)
- Menjalankan sebuah fungsi atau proses
- Membuka jendela baru atau menutup aplikasi



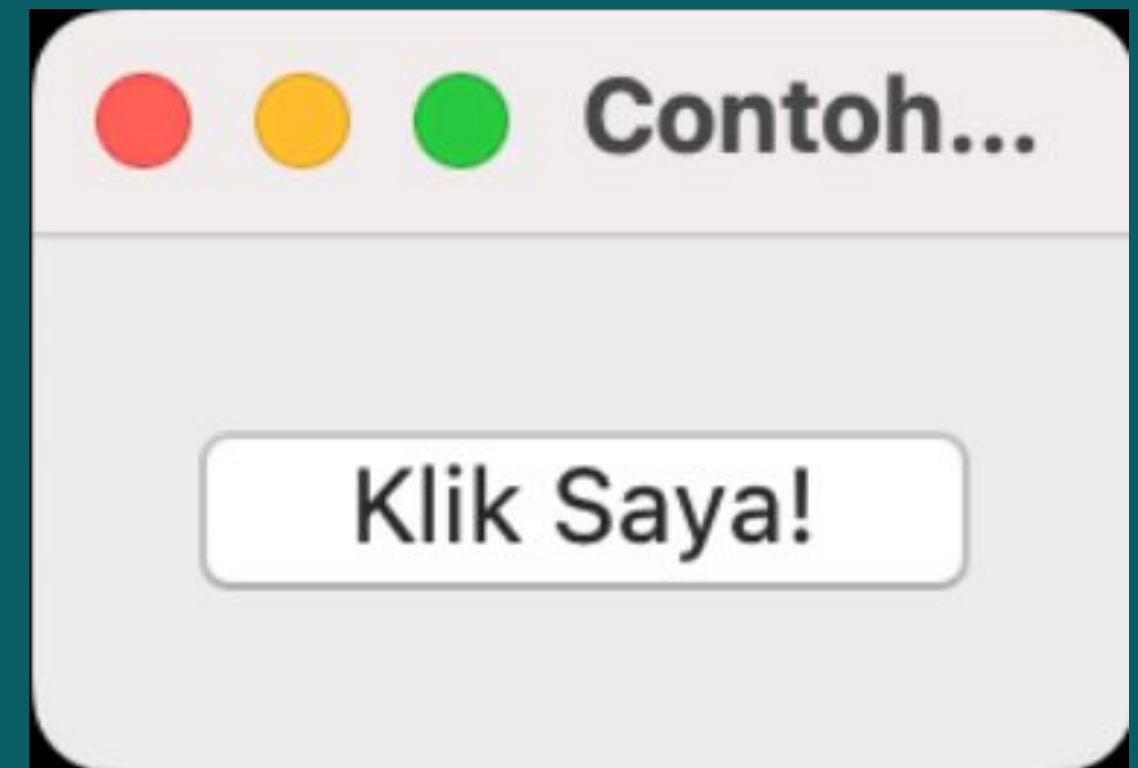
CONTOH

```
import tkinter as tk
from tkinter import ttk

# Fungsi ini akan dijalankan saat tombol diklik
def tampilkan_pesan():
    print("Tombol berhasil diklik! Sebuah aksi sedang berjalan...")
    window = tk.Tk()
    window.title("Contoh Button")

# Membuat widget Button yang terhubung ke sebuah fungsi
tombol_aksi = ttk.Button(
    window,
    text="Klik Saya!",
    command=tampilkan_pesan
)
tombol_aksi.pack(pady=20, padx=20)

window.mainloop()
```



ENTRY

Widget Entry menyediakan sebuah kotak input teks satu baris yang memungkinkan kita untuk mendapatkan data dari pengguna

Full Name

John |

KAPAN DIGUNAKAN

- Formulir login untuk memasukkan username dan password
- Formulir pendaftaran untuk data seperti nama , email , telepon
- Kotak pencarian



CONTOH

```
import tkinter as tk
from tkinter import ttk

def cetak_input():
    # Mengambil teks dari widget Entry dan menampilkannya di terminal
    data_pengguna = input_nama.get()
    print(f"Teks yang diinput: {data_pengguna}")

window = tk.Tk()
window.title("Contoh Entry")

# Widget Entry untuk input
input_nama = ttk.Entry(window, width=30)
input_nama.pack(pady=20, padx=20)

# Tombol untuk memicu pengambilan data
tombol_cetak = ttk.Button(window, text="Cetak Input",
                           command=cetak_input)
tombol_cetak.pack()

window.mainloop()
```



MANAJEMEN LAYOUT



MANAGEMENT LAYOUT

Management Layout adalah proses menyusun dan mengatur posisi semua widget di dalam jendela aplikasi agar rapi dan fungsional.

Terdapat 3 metode :

Pack

Grid

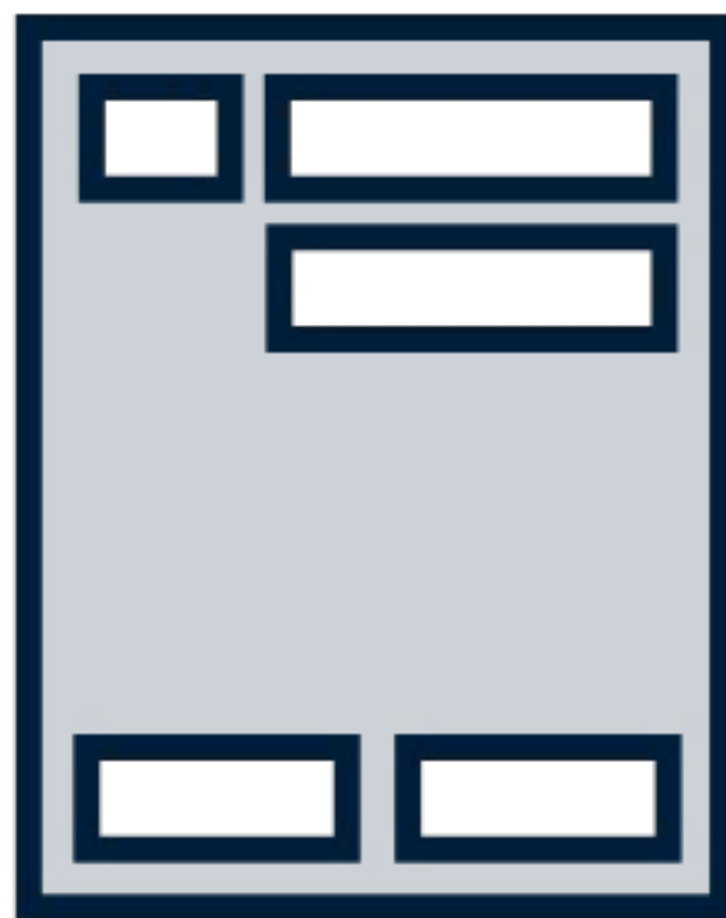
Place





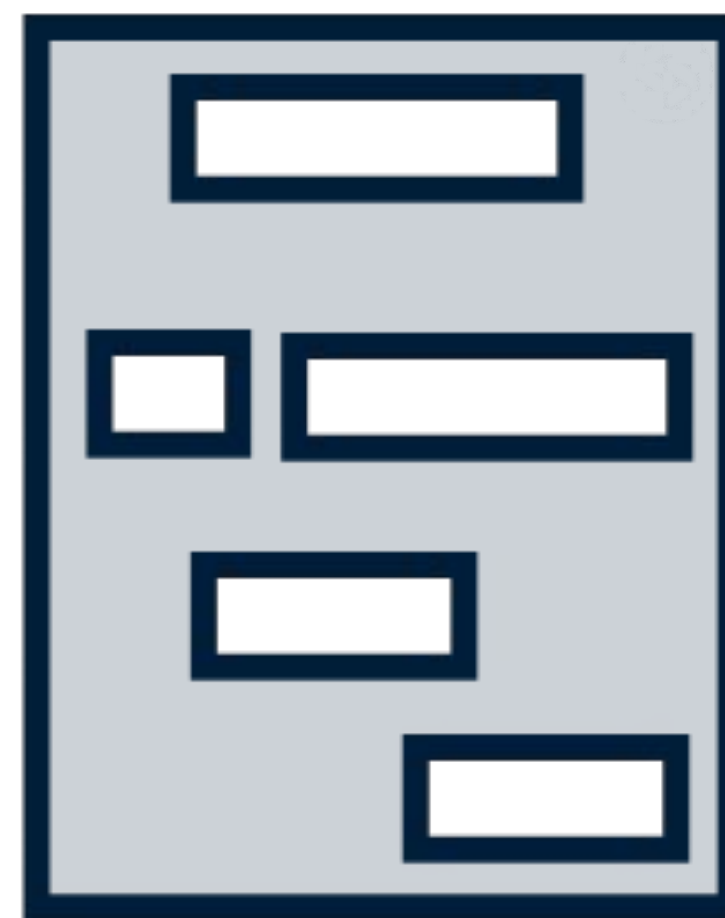
Pack

One after another,
vertically or horizontally



Grid

Assigned to grid cells,
using rows and columns



Place

Put stuff anywhere,
using coordinates



.pack() atau menumpuk

.pack adalah metode paling sederhana yang bekerja dengan menumpuk widget satu per [satu](#). di gunakan saat tata letak yang sangat simpel contohnya beberapa tombol yang tersusun di tengah atau sebuah toolbar di bagian atas/bawah jendela

CONTOH



```
import tkinter as tk
from tkinter import ttk
```

```
window = tk.Tk()
window.title('Contoh Layout')
```

```
ttk.Label(window, text="Label Pertama")
ttk.Button(window, text="Tombol ke dua")
ttk.Label(window, text="Label ke tiga").pack(pady=50 , padx=
50)
window.mainloop()
```

 Conto...

Label Pertama

Tombol Kedua

Label Ketiga



Grid

Grid adalah model paling populer dan flexibel yang menerapkan widget dalam sebuah sistem baris (row) dan kolom (column). Ideal digunakan saat membuat formulir atau tata letak yang kompleks dan butuh perataan presisi

CONTOH



```
import tkinter as tk
from tkinter import ttk
```

```
window = tk.Tk()
window.title("Contoh .grid()")
```

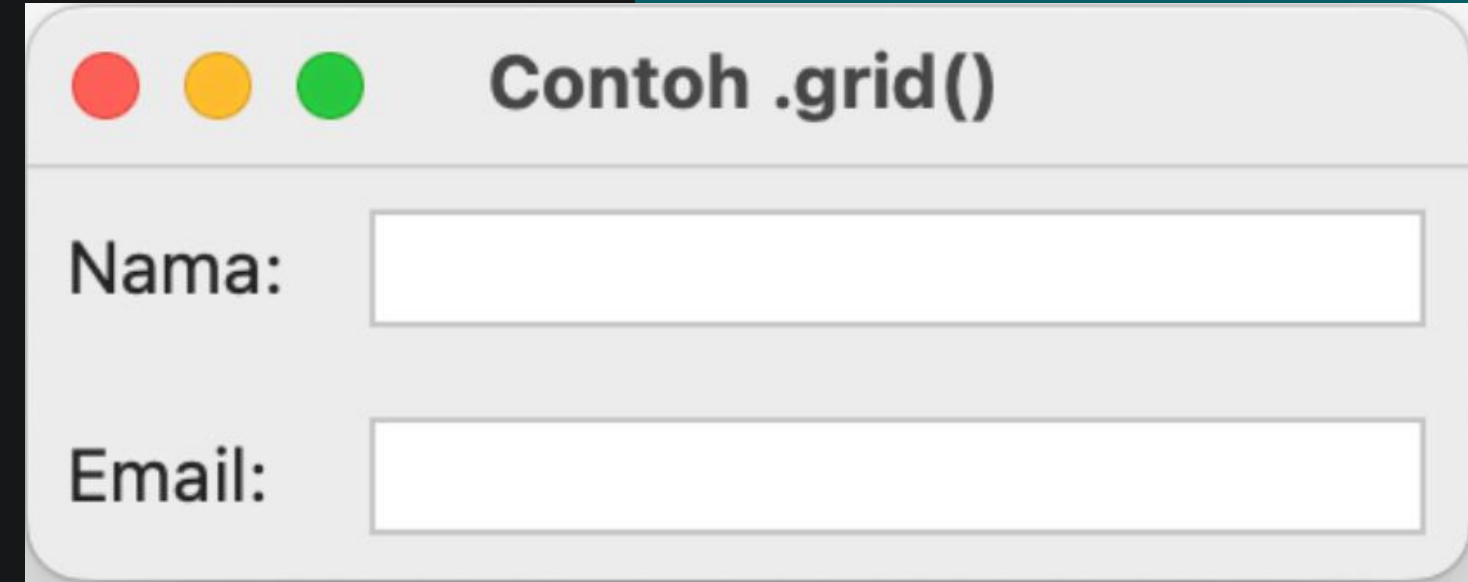
```
# Baris 0
```

```
ttk.Label(window, text="Nama:").grid(row=0, column=0, sticky="w",
padx=5, pady=5)
ttk.Entry(window).grid(row=0, column=1, padx=5, pady=5)
```

```
# Baris 1
```

```
ttk.Label(window, text="Email:").grid(row=1, column=0, sticky="w",
padx=5, pady=5)
ttk.Entry(window).grid(row=1, column=1, padx=5, pady=5)
```

```
window.mainloop()
```



Contoh .grid()

Nama:

Email:



Place

Place merupakan widget menggunakan koordinat x dan y yang absolut berdasarkan piksel

CONTOH



```
import tkinter as tk
from tkinter import ttk

window = tk.Tk()
window.title("Contoh .place()")
window.geometry("400x300") # Memberi ukuran jendela awal

# Membuat tombol pertama
tombol_satu = ttk.Button(window, text="Tombol di posisi (x=30, y=50)")
# Menempatkan tombol 30 piksel dari kiri dan 50 piksel dari atas
tombol_satu.place(x=30, y=50)

# Membuat tombol kedua
tombol_dua = ttk.Button(window, text="Tombol di posisi (x=150, y=120)")
# Menempatkan tombol 150 piksel dari kiri dan 120 piksel dari atas
tombol_dua.place(x=150, y=120)

window.mainloop()
```



Contoh .place()

Tombol di posisi (x=30, y=50)

Tombol di posisi (x=150, y=120)



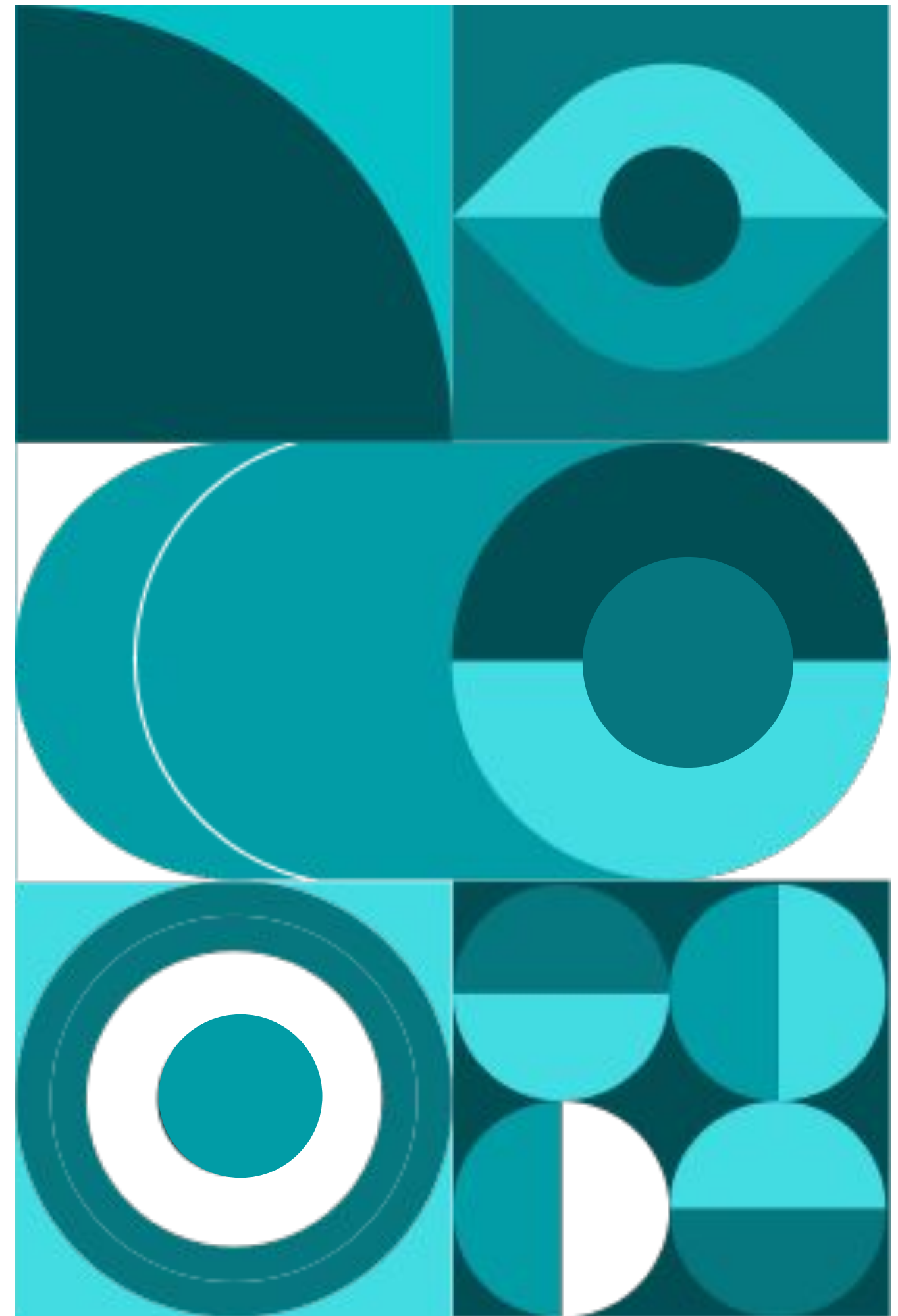
EVENT HANDLING



EVENT ATAU EVENT HANDLER

Event (Kejadian) adalah setiap aksi yang dilakukan oleh pengguna pada aplikasi contoh : mengklik mouse, menekan tombol ke keyboard, atau bahkan hanya menggerakkan mouse

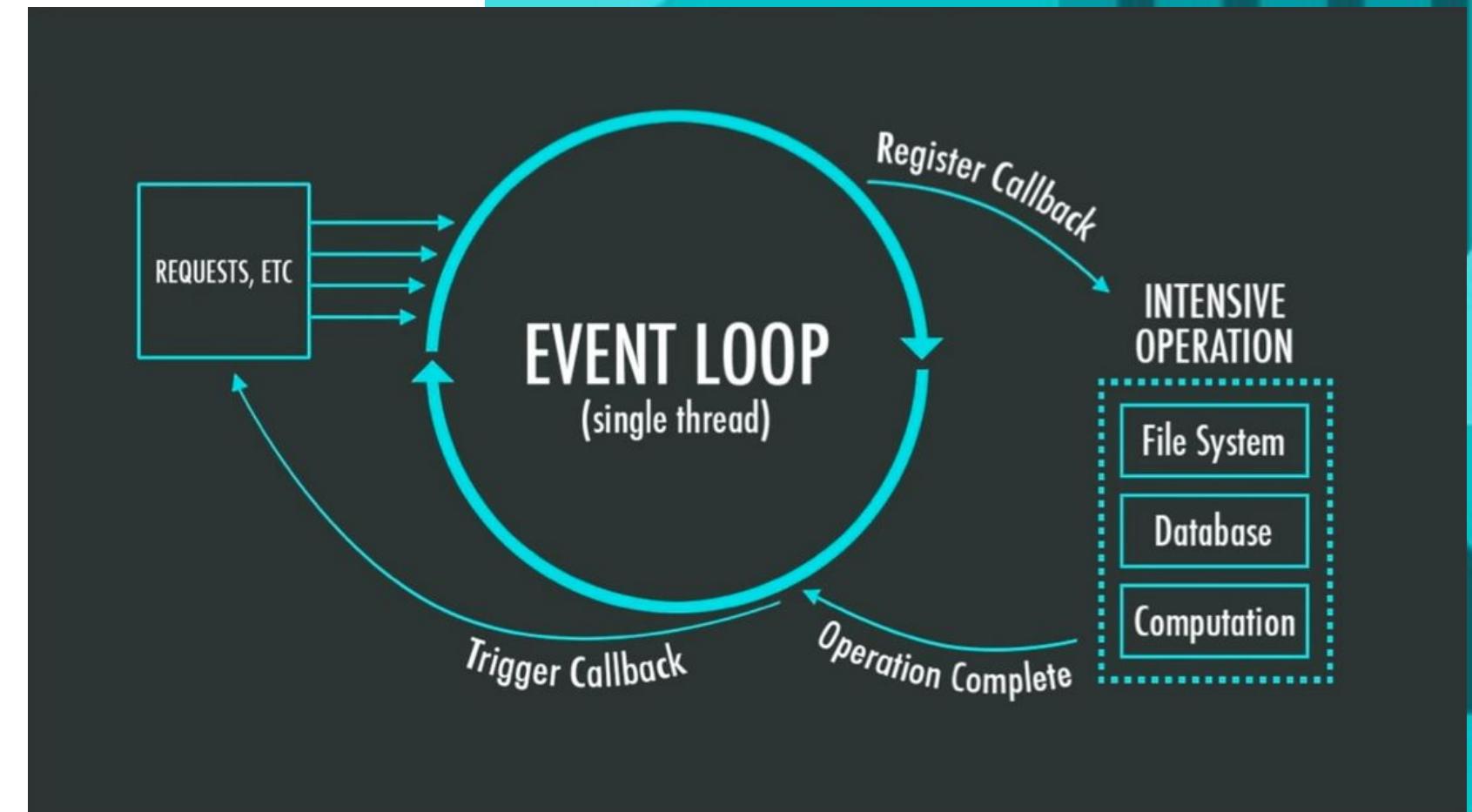
Event Handler adalah blok kode atau fungsi yang kita tulis secara spesifik untuk mendengarkan dan merespon sebuah event tertentu



EVENT LOOP

Event Loop adalah sebuah putaran tak terbatas (infinite loop) yang menjadi inti dari setiap aplikasi GUI. Tugasnya

1. Terus menerus memeriksa event
2. Apakah ada event baru (misalnya , mouse click) ?
3. Jika ya cari event handler (fungsi) yang sesuai
4. Jalankan fungsi tersebut
5. Kembali ke langkah 1




Cara Menghubungkan Event ke fungsi (Binding)

Ada dua cara utama untuk memberitahu sebuah widget fungsi mana yang harus dijalankan saat sebuah event terjadi

1. Menggunakan Opsi command adalah dengan memberikan nama fungsi tanpa tanda kurung pada command
2. Menggunakan Metode `.bind()` digunakan untuk event yang lebih spesifik atau pada widget yang tidak memiliki opsi command

MENGGUNAKAN OPSI COMMAND



```
def sapa_pengguna( ):  
    print("Halo Dunia!")
```

```
# Fungsi 'sapa_pengguna' akan dieksekusi saat tombol diklik  
tombol = ttk.Button(window, text="Sapa!",  
command=sapa_pengguna)
```

MENGGUNAKAN METODE .BIND



```
def saat_mouse_masuk(event):  
    # Fungsi handler untuk .bind() biasanya menerima argumen 'event'  
    print("Mouse memasuki area label!")  
  
label = ttk.Label(window, text="Arahkan mouse ke sini")  
# Mengikat event <Enter> (mouse masuk) ke fungsi  
label.bind("<Enter>", saat_mouse_masuk)
```

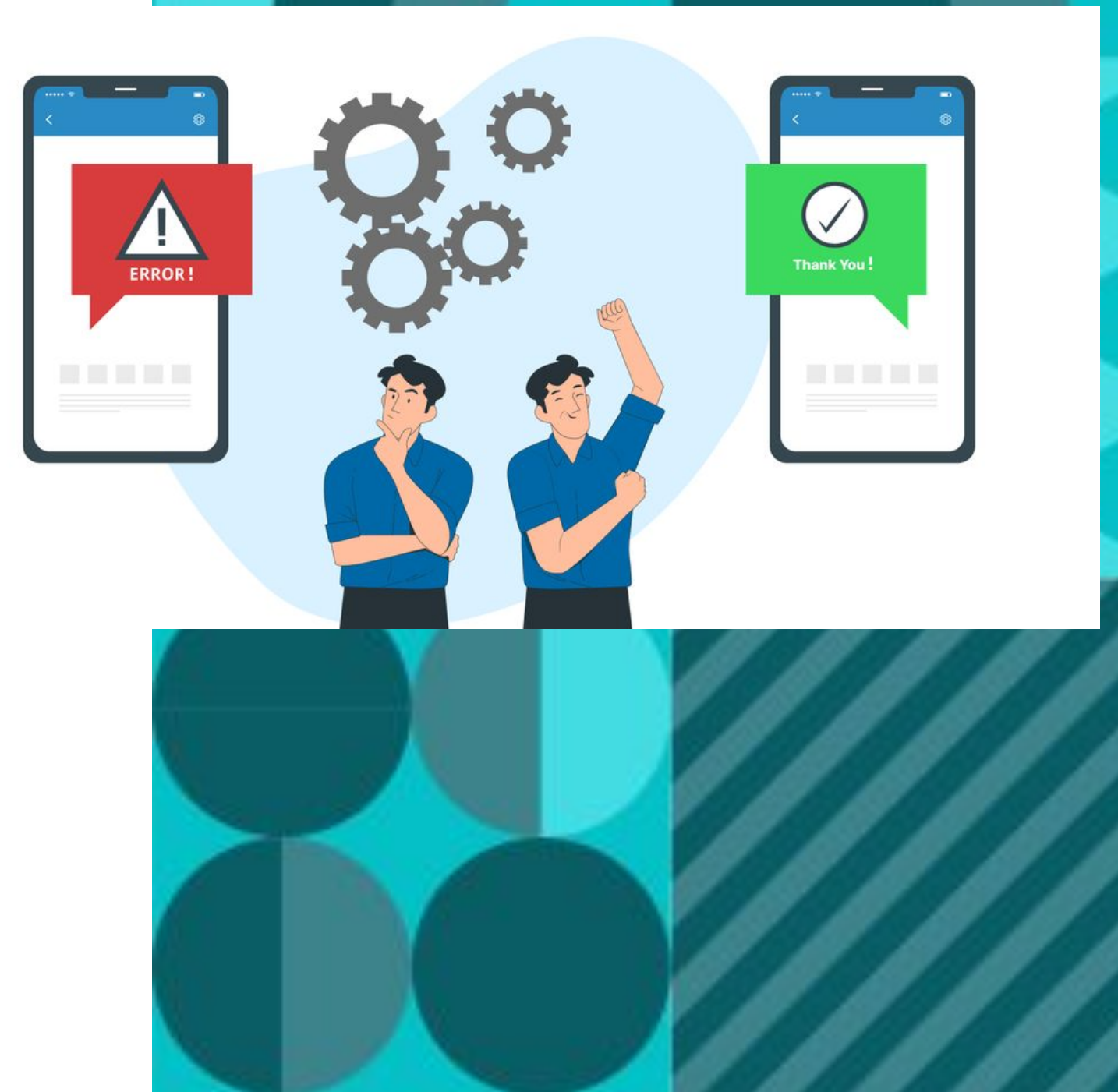
DASAR-DASAR ERROR HANDLING



ERROR HANDLING

Error Handling adalah sebuah teknik atau proses dalam pemrograman untuk mengantisipasi, mendeteksi, dan merespon kesalahan (error atau exception) yang terjadi saat program sedang berjalan.

Tujuan utamanya adalah untuk mencegah aplikasi berhenti total (crash) saat error terjadi. Sebagai gantinya kita bisa memberikan respons yang lebih terkontrol dan ramah kepada pengguna



CONTOH CLI



```
# Program tidak akan crash
try:
    # Kode yang berisiko eror
    angka = int("bukan angka")
    print(angka)
except ValueError:
    # Kode ini hanya berjalan jika terjadi ValueError
    print("Terjadi kesalahan! Input yang dimasukkan bukan
angka.")
print("Program tetap lanjut berjalan...")
```


PRAKTEK MENDALAM TENTANG GUI

```
import tkinter as tk
from tkinter import ttk, messagebox

def hitung_pembagian():
    try:
        # Coba ambil dan konversi input ke angka
        angka1 = float(entry_1.get())
        angka2 = float(entry_2.get())

        # Coba lakukan pembagian
        hasil = angka1 / angka2

        # Tampilkan hasil jika semua berhasil
        hasil_label.config(text=f"Hasil: {hasil}")

    except ValueError:
        # Tangani jika input bukan angka
        messagebox.showerror("Input Salah", "Harap masukkan angka yang valid.")
    except ZeroDivisionError:
        # Tangani jika terjadi pembagian dengan nol
        messagebox.showerror("Kesalahan Matematis", "Tidak bisa membagi dengan nol.")

# ... (kode untuk membuat jendela, entry_1, entry_2, tombol, dan hasil_label) ...
# ... (tombol dihubungkan ke command=hitung_pembagian) ...
```

Hands On

Coba test dan uji sesuai dengan yang ada di slide
dan Coba test implementasikan pada project yang akan di
rancang sebelumnya

THANK YOU
