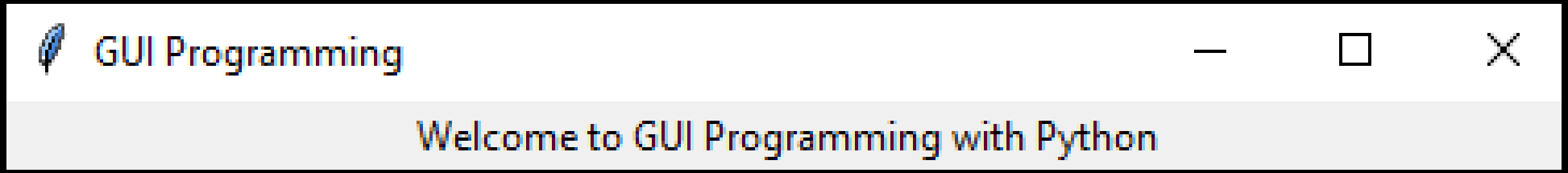
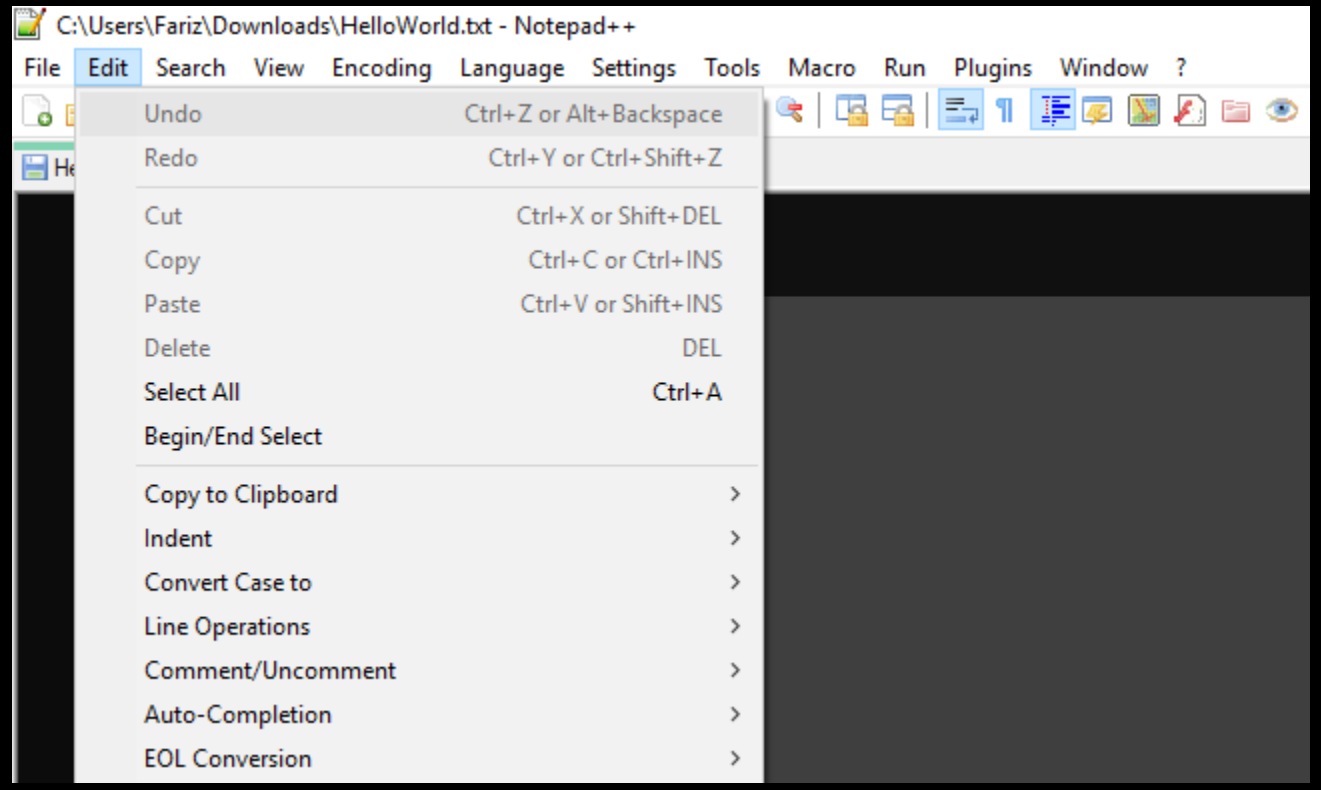
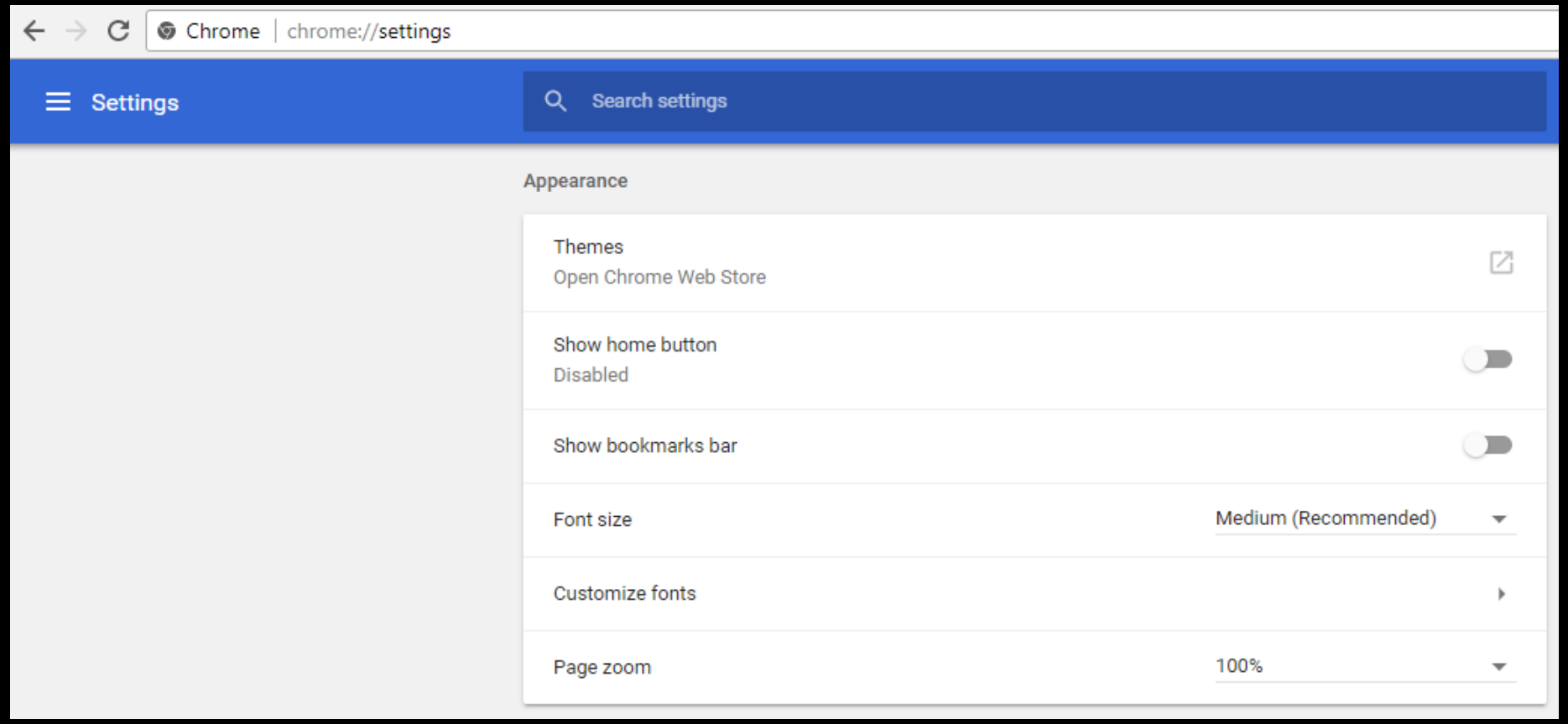

GUI Programming
Welcome to GUI Programming with Python

Graphical User Interface (GUI) ada di mana-mana



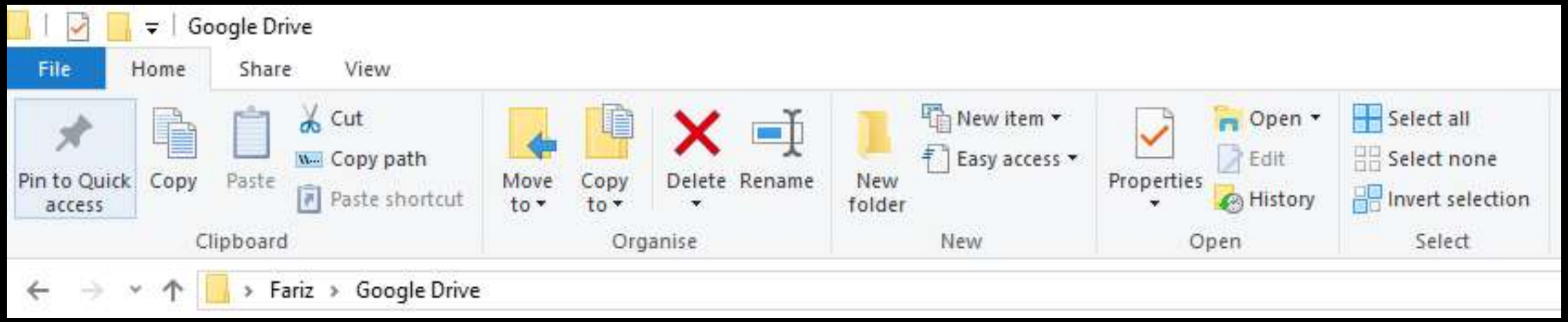
GUI of Notepad++

Graphical User Interface (GUI) ada di mana-mana



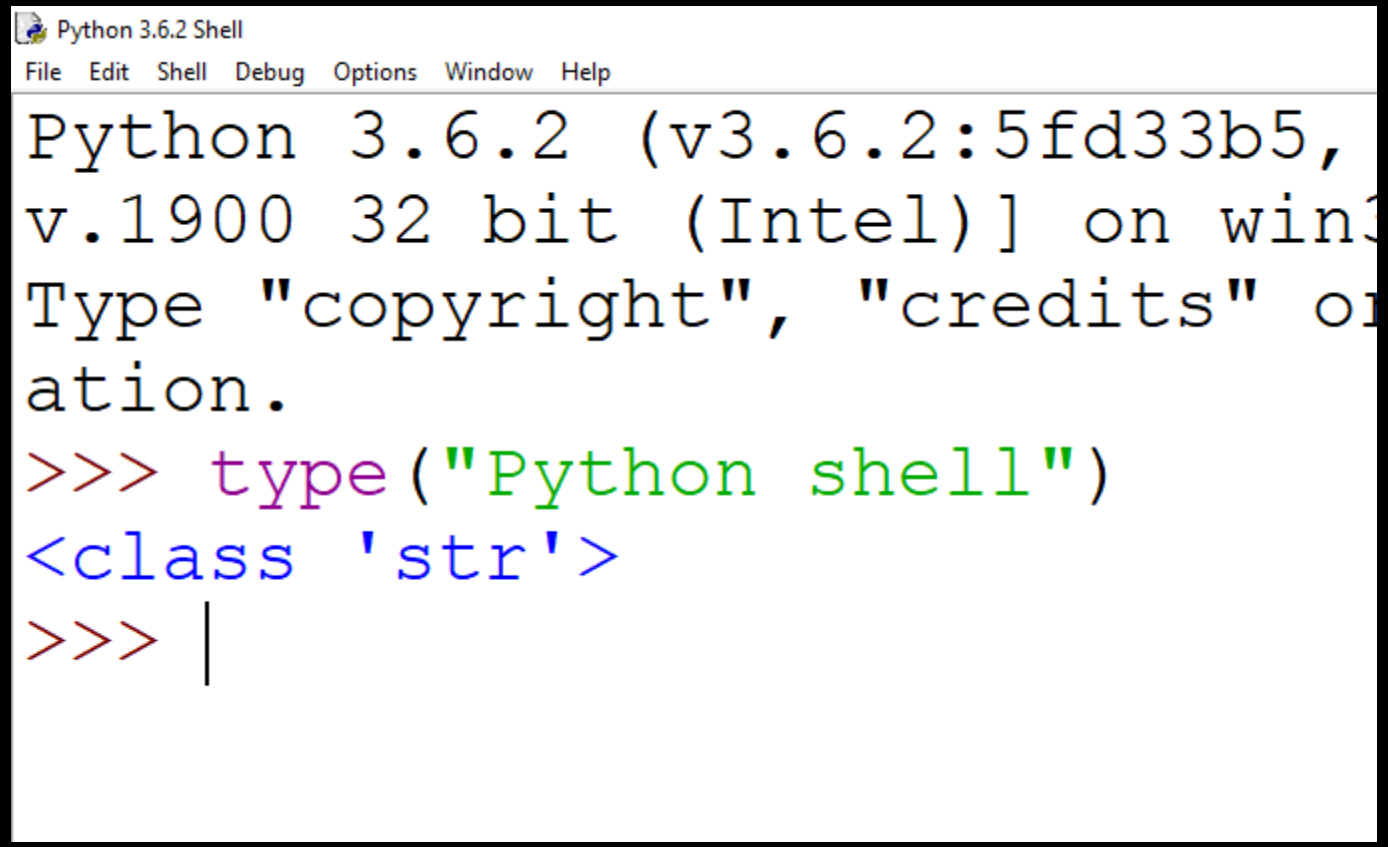
GUI untuk web browser

Graphical User Interface (GUI) ada di mana-mana



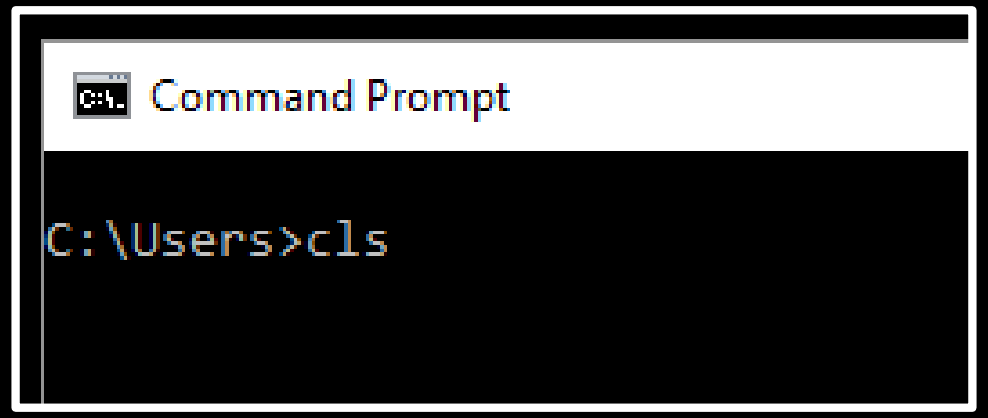
GUI untuk file explorer

Kenapa tidak pakai Command-Line Interface (CLI)?



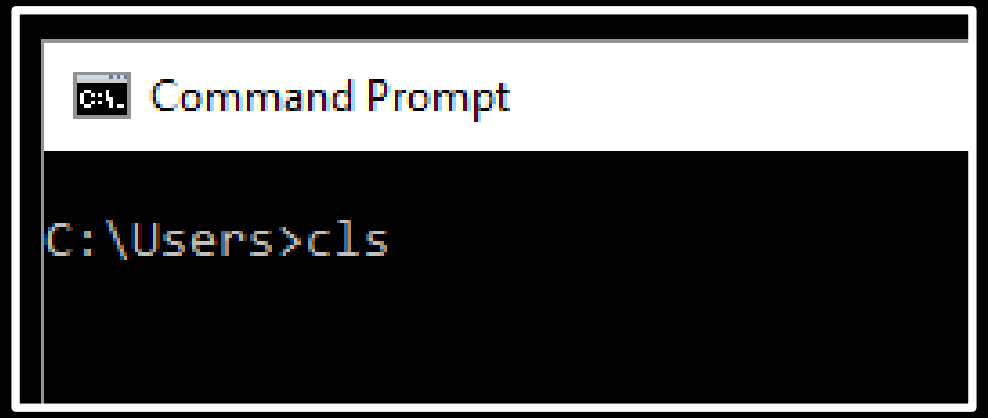
Python Shell

Kenapa tidak pakai Command-Line Interface (CLI)?



Windows Command Prompt

Kenapa tidak pakai Command-Line Interface (CLI)?



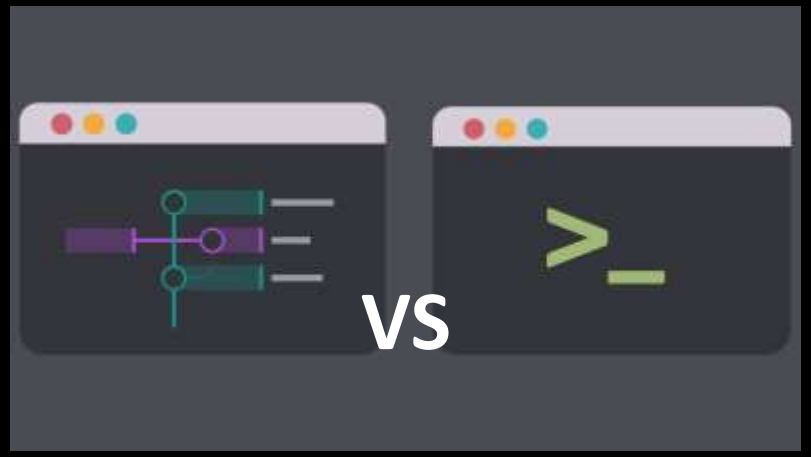
Windows Command Prompt

Tidak semua pengguna komputer nyaman menggunakan CLI, bayangkan kakek-nenek kamu menggunakan komputer!

Semua ada positif dan negatifnya..

When to use GUI

- Reduce mental work
- Make results visible
- Make the barrier of the entry lower



When to use CLI

- Do things at scale: A simple CLI command can easily adjust configurations for a large group of systems
- Something needs to be scripted and automated
- For less memory usage
- ~~To look cooler (aka. hacker-style)~~

Apa itu GUI?

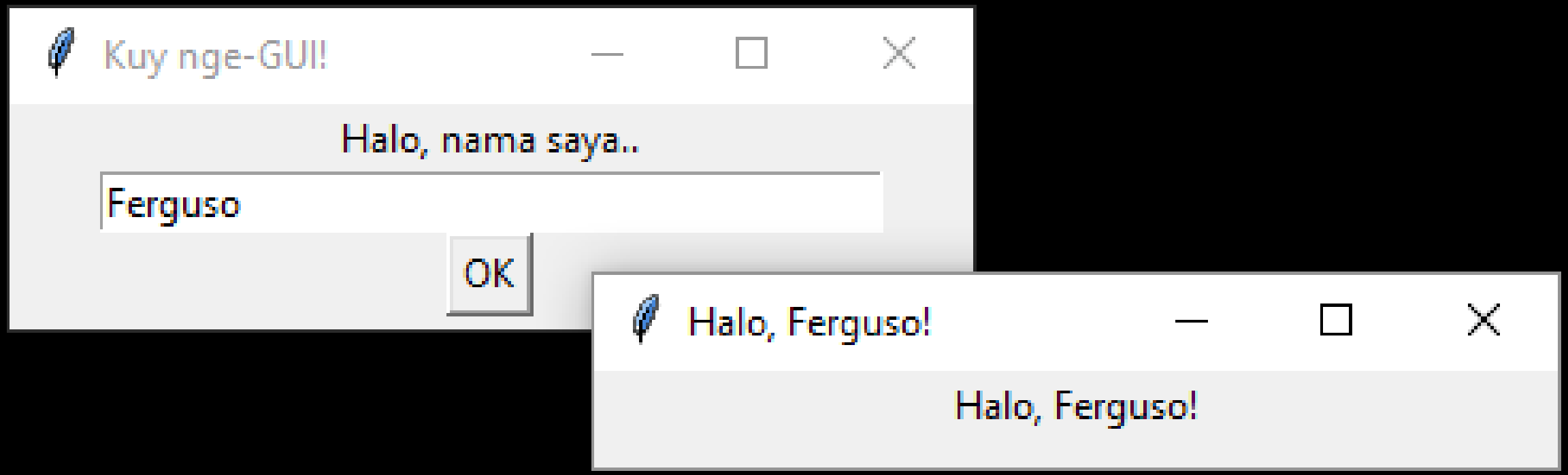
Suatu antarmuka (user interface) dimana pengguna berinteraksi melalui objek-objek visual (widgets), seperti tombol, checkbox, menu, dsb.

Dengan kata lain, **visual-based interface** (kalau CLI, lebih bersifat text-based interface).

tkinter: Modul GUI untuk Python

- GUI library (= modul) standar bawaan Python
- tkinter = **T**ool**K**it **INTER**face
- Cross-platform = jalan di Windows, Linux, Mac OS
- Menyediakan berbagai elemen-elemen GUI (widgets) seperti tombol, menu, entry (= text field, untuk buat form), dsb
- Event-driven = GUI akan merespons ke event-event (aksi-aksi) pengguna GUI (misalnya klik, ketik, scroll, dsb)

Kuy nge-GUI!


Kuy nge-GUI!
Halo, nama saya..
Ferguso
OK
Halo, Ferguso!
Halo, Ferguso!

tkinter: Tk class

- tkinter menyediakan class-class untuk membuat GUI
  - OOP (object-oriented programming) is everywhere!
- Class **Tk** membuat window untuk menampung widgets (= komponen-komponen visual)

Live Coding: Membuat window

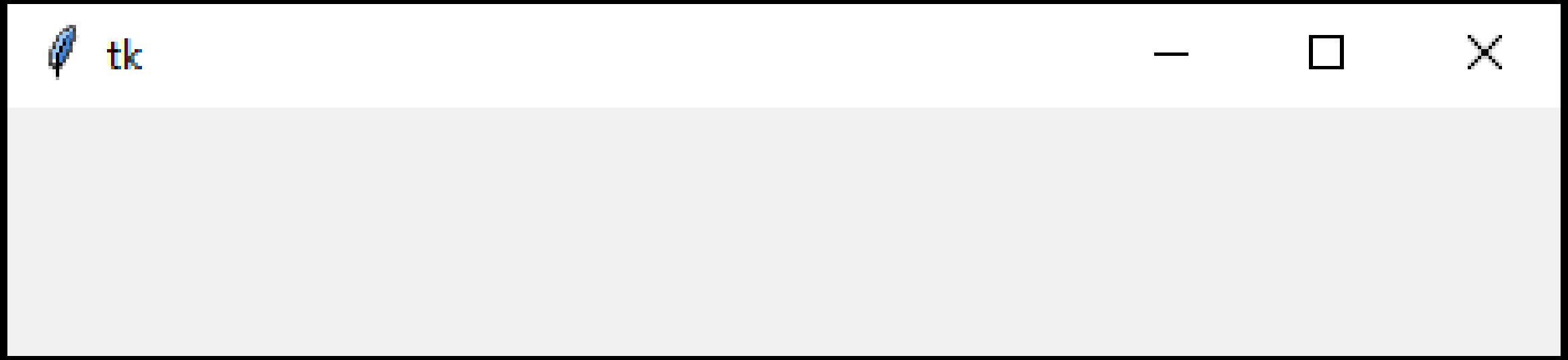
```
import tkinter
window = tkinter.Tk() # membuat window
window.mainloop() # jalankan tkinter
```

Live Coding: Membuat window

```
from tkinter import Tk # dengan cara ini jadi lebih ringkas
window = Tk() # membuat window
window.mainloop() # jalankan tkinter
```

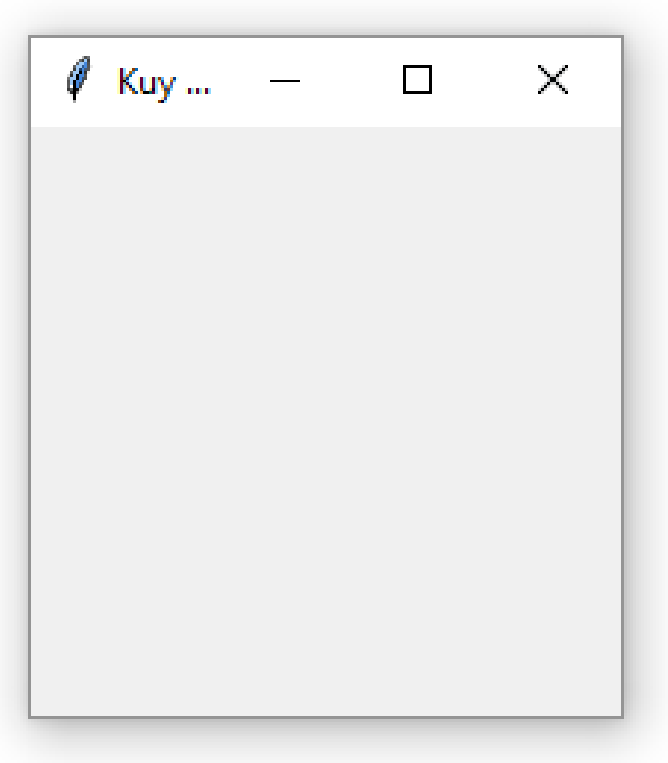
Live Coding: Membuat window

```
from tkinter import Tk # dengan cara ini jadi lebih ringkas
window = Tk() # membuat window
window.mainloop() # jalankan tkinter
```



Live Coding: Membuat window **dengan judul**

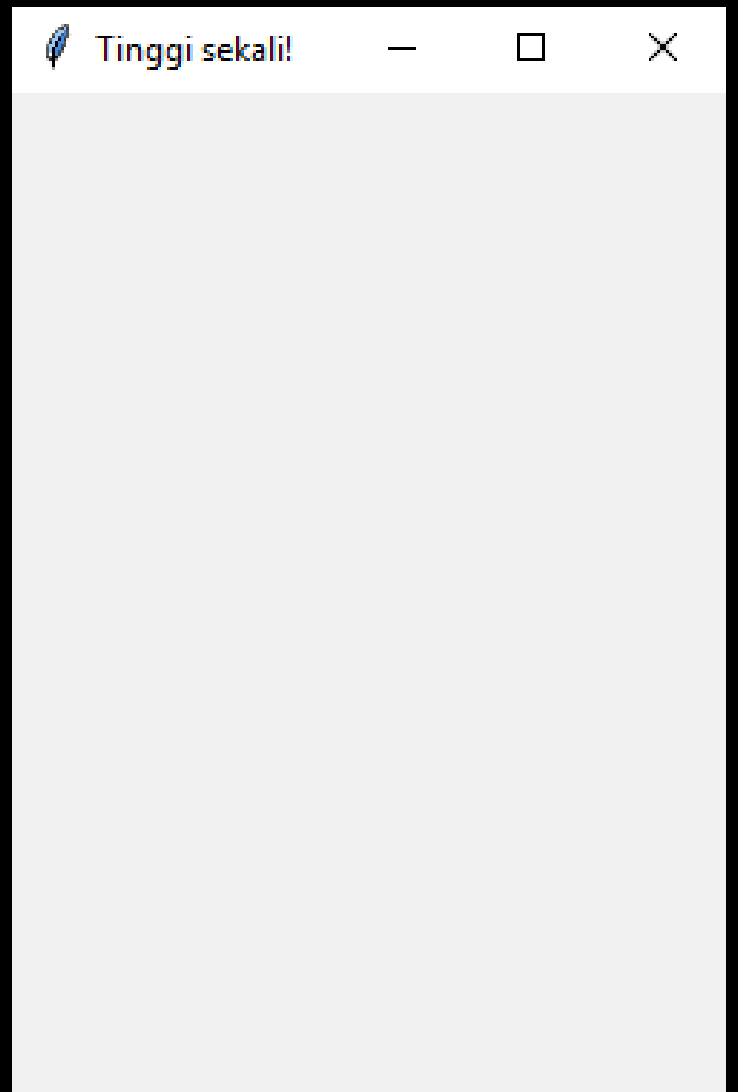
```
from tkinter import Tk
window = Tk()
window.title("Kuy nge-GUI!")
window.mainloop()
```



Live Coding: Atur ukuran window

```
from tkinter import Tk
window = Tk()
window.title("Kuy nge-GUI!")
window.geometry("500x100") # atur lebar 500, tinggi 100 pixels
window.mainloop()
```

Live Coding: Window yang tinggi sekali



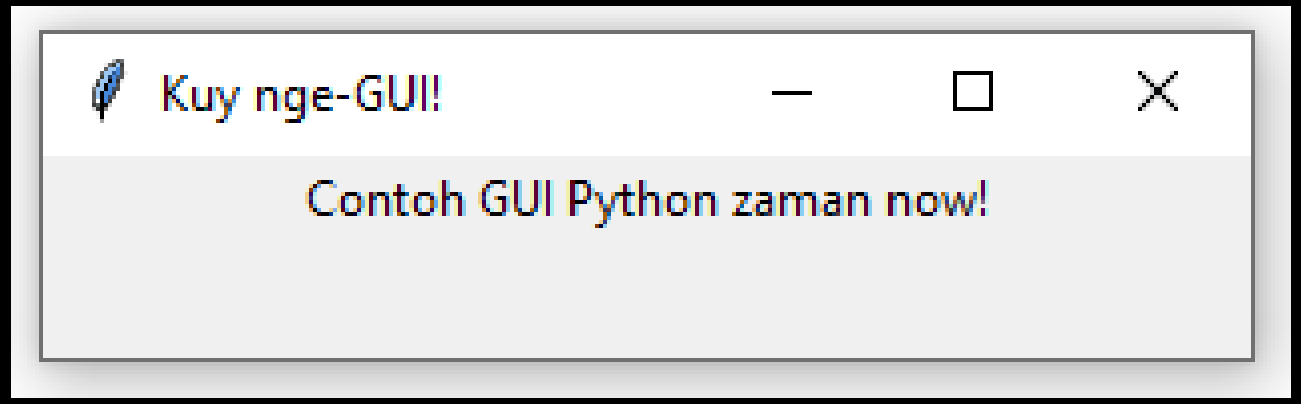
Live Coding: Window yang tinggi sekali

```
from tkinter import Tk

window = Tk()
window.title("Tinggi sekali!")
window.geometry("250x5000")
window.mainloop()
```

Live Coding: Ada label

```
from tkinter import *

class MyFirstGUI:

    def __init__(self, master):
        self.master = master
        master.title("Kuy nge-GUI!")
        master.geometry("300x50")

        # Label adalah salah satu widget di tkinter
        # dan argumen pertama di widget constructor selalu parent container
        # yang akan menampung widget terkait
        self.label = Label(master, text="Contoh GUI Python zaman now!")
        self.label.pack() # tempatkan labelnya di container

root = Tk()
my_gui = MyFirstGUI(root)
root.mainloop()
```

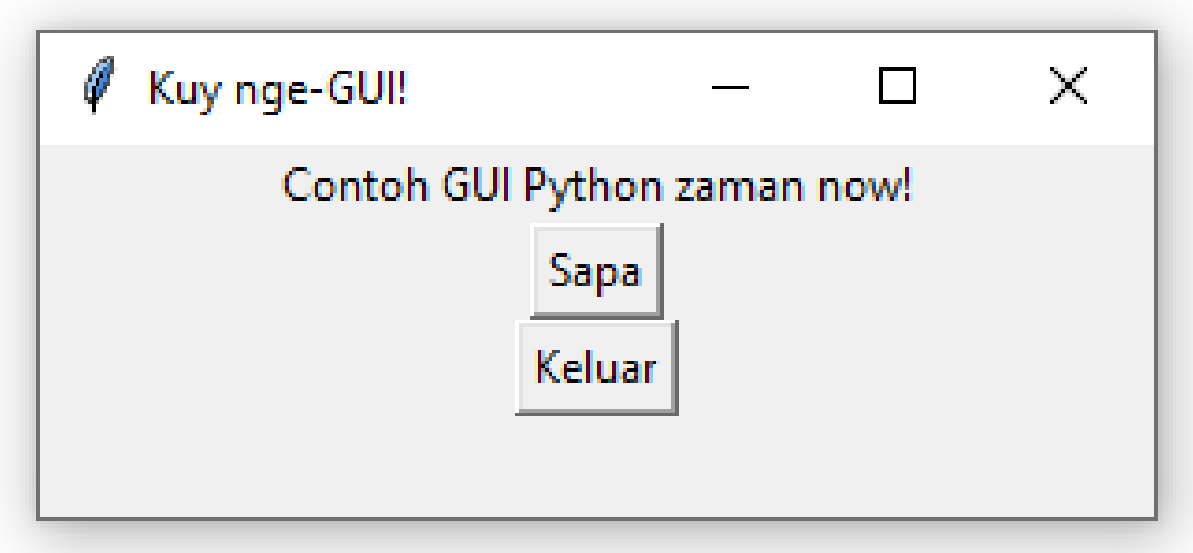
Live Coding: Ada label



Live Coding: Ada label dan button

```
# di dalam __init__ kode sebelumnya, tambahkan berikut..
# jangan lupa window height di-set ke 100

self.greet_button = Button(master, text="Sapa")
self.greet_button.pack()
self.close_button = Button(master, text="Keluar")
self.close_button.pack()
```

Live Coding: Ada label dan button

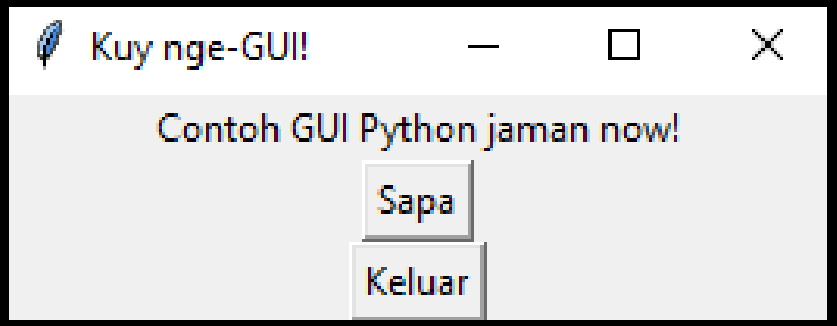


Live Coding: Biar buttonnya gak cuma hiasan

```
# ubah bagian close_button menjadi berikut

self.close_button = Button(master, text="Keluar", command=master.destroy)
```

Live Coding: Biar buttonnya gak cuma hiasan

```
# ubah bagian close_button menjadi berikut

self.close_button = Button(master, text="Keluar", command=master.destroy)
```



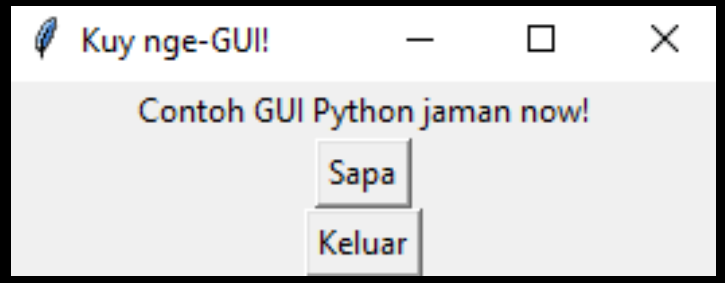
klik button Keluar → ***Window lenyap!***

Live Coding: Biar buttonnya gak cuma hiasan

```
# di definisi class MyFirstGUI tambahkan method berikut

def sapa(self):
   print("Ciaooooo!") # sapa bahasa Italia

# ubah bagian greet_button menjadi berikut

self.greet_button = Button(master, text="Sapa", command=self.sapa)
```

Live Coding: Biar buttonnya gak cuma hiasan

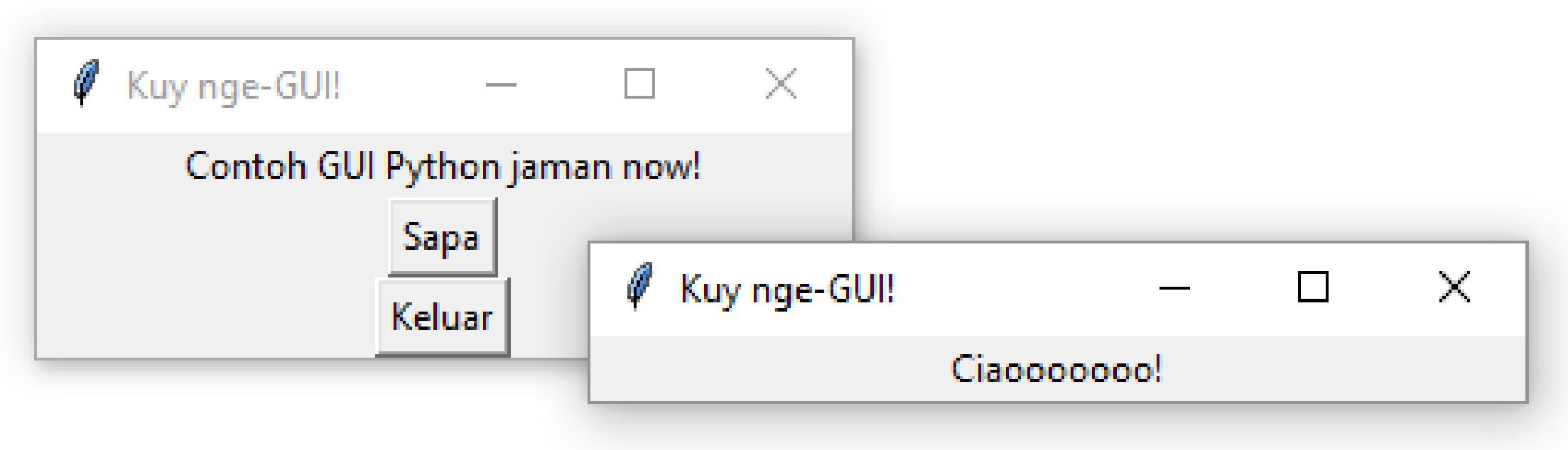
```
# di definisi class MyFirstGUI tambahkan method berikut

def sapa(self):
   print("Ciaooooo!") # sapa bahasa Italia

# ubah bagian greet_button menjadi berikut

self.greet_button = Button(master, text="Sapa", command=self.sapa)
```



klik button Sapa → ***Cetak "Ciaooooo!" pada command line/shell***

Live Coding: Biar buttonnya gak cuma hiasan (Pop-up)

```
# di definisi class MyFirstGUI tambahkan method berikut

def sapa(self):
    window = Toplevel(self.master)
    Label(window, text="Ciaooooooo!").pack()
```

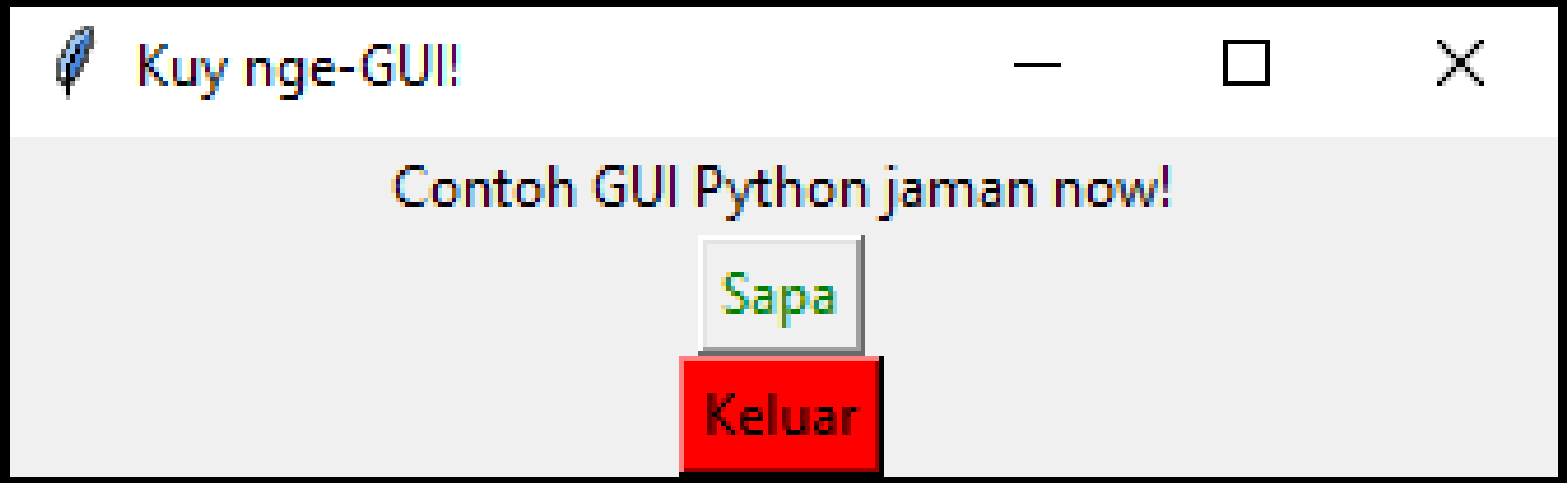


Live Coding: Mewarnai

```
# ubah bagian greet_button dan close_button menjadi berikut

self.greet_button = Button(master, text="Sapa", command=self.sapa, fg="green")
self.close_button = Button(master, text="Keluar", command=master.destroy, bg="red")
```

Live Coding: Mewarnai

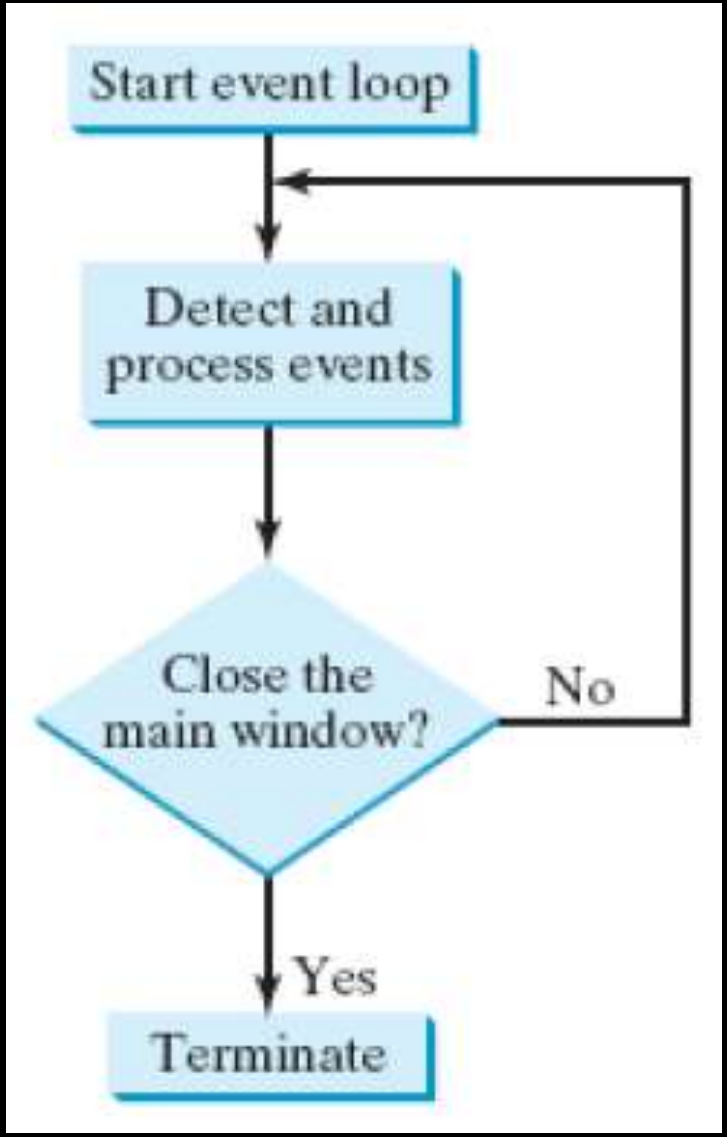
```
# ubah bagian greet_button dan close_button menjadi berikut

self.greet_button = Button(master, text="Sapa", command=self.sapa, fg="green")
self.close_button = Button(master, text="Keluar", command=master.destroy, bg="red")
```



Event Processing

- tkinter GUI bersifat event-driven,
  yakni selalu menunggu aksi (= event) pengguna
- Dispesifikasikan dengan method **`mainloop()`**
- Method tersebut membuat event-loop,
  yang akan memproses event terus menerus
  sampai window-nya ditutup (atau ada error!)

Event Processing

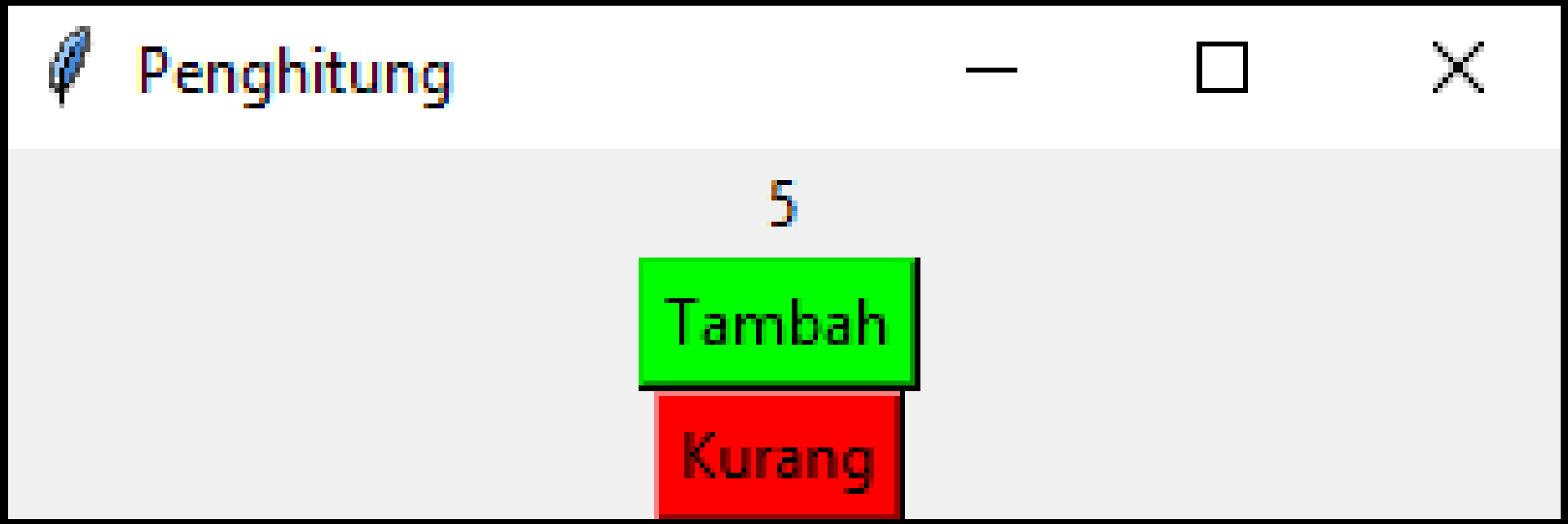


Event Processing pada Widget

- Widget dapat diberi event handler (= callback function), yang dieksekusi apabila terjadi suatu event pada widget
- Misalnya pada saat pengguna klik suatu button:

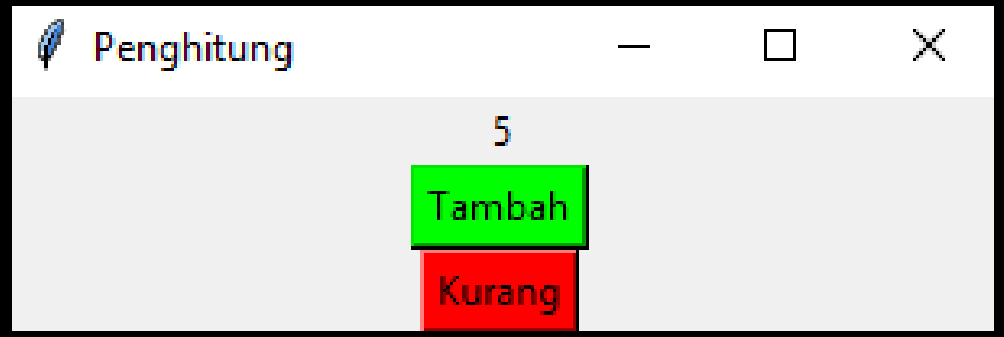
```
self.greet_button = Button(master, text="Sapa", command=self.sapa)
self.close_button = Button(master, text="Keluar", command=master.destroy)
```

Live Coding: Penghitung



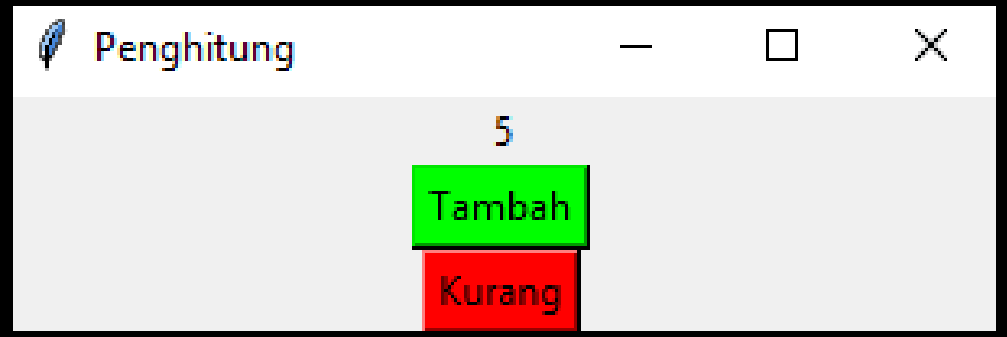
Live Coding: Penghitung

```
from tkinter import *

class Penghitung:

    def __init__(self, master):
        self.master = master
        master.title(_________)
        master.geometry("300x100")

        self.hitungan = 0
        self.label = Label(master, text=________)
        self.label.pack()

        self.tambah_button = Button(master, text="Tambah", command=_________, bg="#00FF00")
        self.tambah_button.pack()
        self.kurang_button = Button(master, text="Kurang", command=_________, bg="#FF0000")
        self.kurang_button.pack()
```



Live Coding: Penghitung

```
# ....

    def _________(self):
        self.hitungan += 1
        self._________ = self.hitungan

    def _________(self):
        self.hitungan -= 1
        self._________ = self.hitungan

root = Tk()
penghitung = Penghitung(root)
root.mainloop()
```



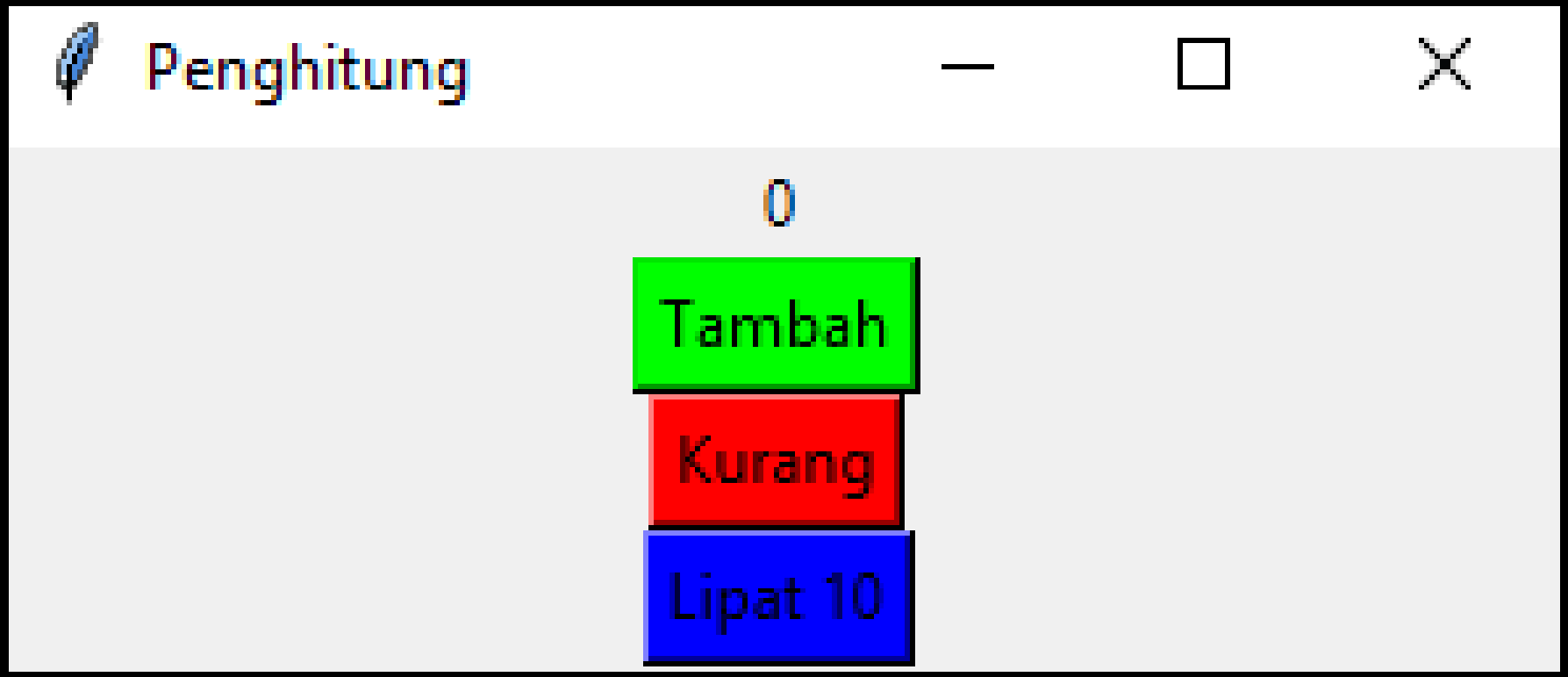
Live Coding: Penghitung

```
from tkinter import *

class Penghitung:

    def __init__(self, master):
        self.master = master
        master.title("Penghitung")
        master.geometry("300x100")

        self.hitungan = 0
        self.label = Label(master, text=self.hitungan)
        self.label.pack()

        self.tambah_button = Button(master, text="Tambah", command=self.tambah, bg="#00FF00")
        self.tambah_button.pack()
        self.kurang_button = Button(master, text="Kurang", command=self.kurang, bg="#FF0000")
        self.kurang_button.pack()
```

Live Coding: Penghitung

```
# ....

    def tambah(self):
        self.hitungan += 1
        self.label['text'] = self.hitungan

    def kurang(self):
        self.hitungan -= 1
        self.label['text'] = self.hitungan

root = Tk()
penghitung = Penghitung(root)
root.mainloop()
```

Live Coding: Penghitung + Fitur Lipat 10



Mengubah atribut dari suatu widget

```
def tambah(self):
        self.hitungan += 1
        self.label['text'] = str(self.hitungan)

def kurang(self):
        self.hitungan -= 1
        self.label['text'] = str(self.hitungan)
```

- Ketika membuat widget, kita bisa spesifikasikan atributnya seperti text, warna foreground dan background, dsb.
- Atribut tersebut dapat diubah dengan pendekatan ala dictionary

```
widgetName["nama atribut"] = nilaiBaru
```

Daftar Atribut Widget

Common Widget Properties	Description
`bg`	Background color.
`fg`	Foreground color.
`width`	Width in pixels
`height`	Height in pixels
`borderwidth`	The size of the border in pixels.
`text`	Text displayed on the widget.
`font`	The font used for text on the widget.
`cursor`	The shape of the cursor when the cursor is over the widget.
`activeforeground`	The color of the text when the widget is activated.
`activebackground`	The color of the background when the widget is activated.
`image`	An image to be displayed on the widget.

Live Coding: Font

```
from tkinter import *

class MyFirstGUI:

    def __init__(self, master):
        self.master = master
        master.title("Kuy nge-GUI!")

        self.label = Label(master, text="Contoh GUI Python jaman now!",
font="Courier 20 bold italic underline")
        self.label.pack()

root = Tk()
my_gui = MyFirstGUI(root)
root.mainloop()
```

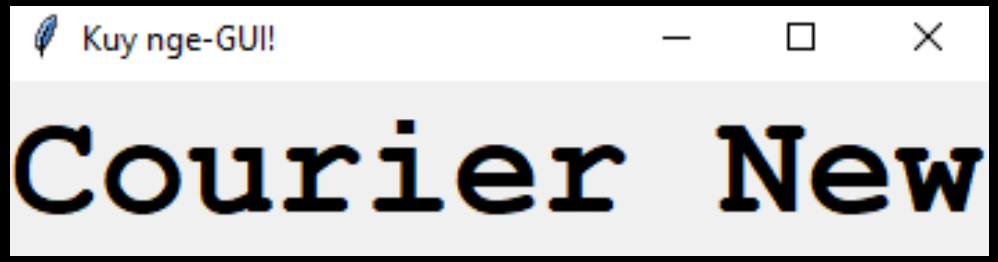
Live Coding: Font

Pilihan Font Lain

- Times
- Helvetica
- Σψμβολ
- Courier New

Live Coding: Font Instance

```
from tkinter import *
from tkinter.font import *

class MyFirstGUI:

    def __init__(self, master):
        self.master = master
        master.title("Kuy nge-GUI!")

        label_font = Font(family='Courier New', size=40, weight='bold')
        self.label = Label(master, text="Courier New", font=label_font)
        self.label.pack()

root = Tk()
my_gui = MyFirstGUI(root)
root.mainloop()
```



Daftar Widget pada Python

Widget Class	Description
Button	A simple button, used to execute a command.
Canvas	An area to display graphical elements like lines, rectangles or text.
Checkbutton	Clicking a check button toggles between the values.
Entry	A text entry field, also called a text field or a text box.
Frame	A container widget for containing other widgets.
Label	Displays text or an image.
Menu	A menu pane, used to implement pull-down and popup menus.
Menubutton	A menu button, used to implement pull-down menus.
Message	Displays a text. Similar to the label widget, but can automatically wrap text to a given width or aspect ratio.
Radiobutton	Clicking a radio button sets the variable to that value, and clears all other radio buttons associated with the same variable.
Text	Formatted text display. Allows you to display and edit text with various styles and attributes.

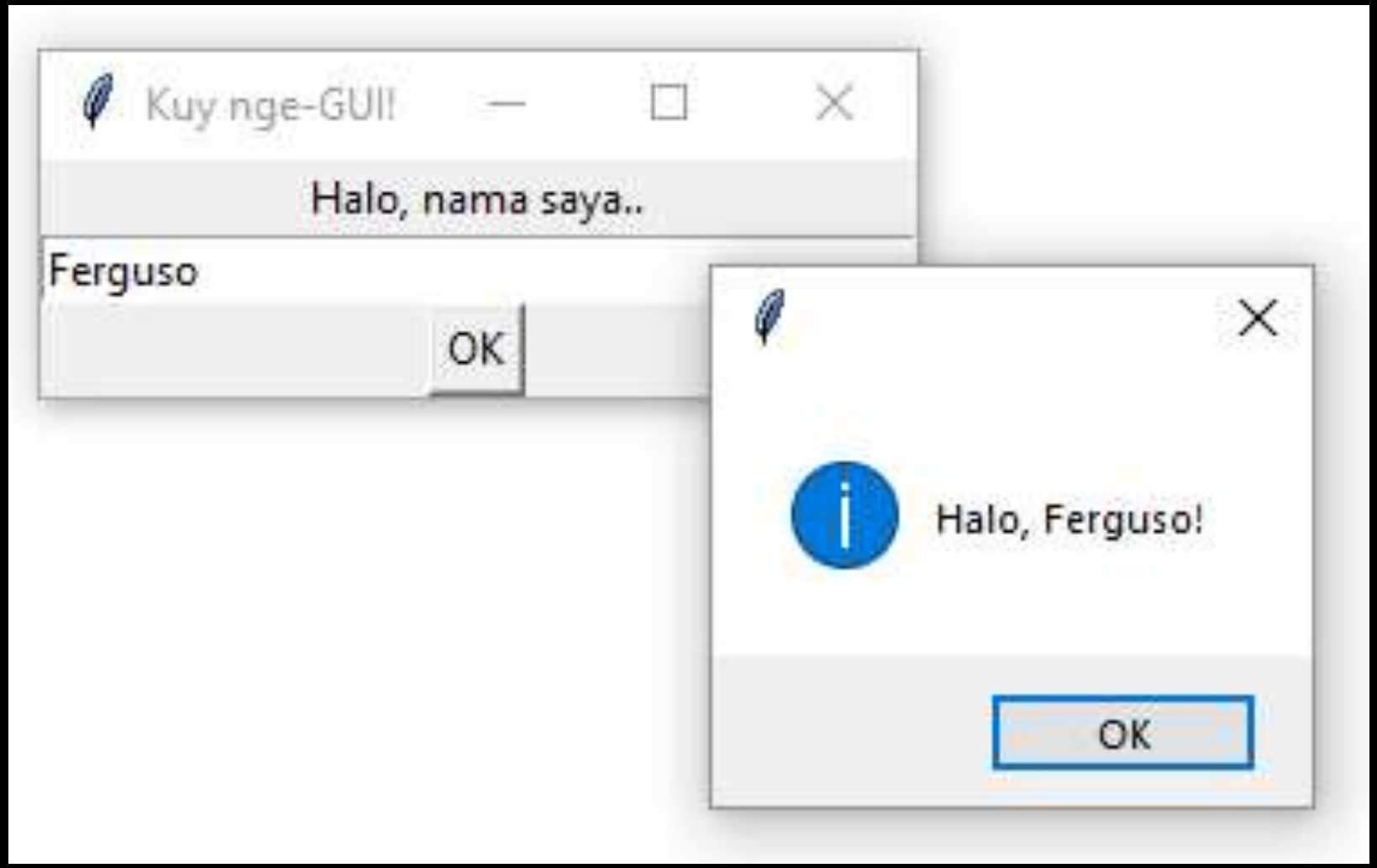
Live Coding: Entry Widget

```
from tkinter import *
from tkinter.messagebox import showinfo

class Formulirku:

    def __init__(self, master):
        self.master = master
        master.title("Kuy nge-GUI!")

        self.label = Label(master, text="Halo, nama saya..")
        self.label.pack()

        self.nama = StringVar()
        self.field_nama = Entry(master, textvariable=self.nama, width=40)
        self.field_nama.pack()

        self.button = Button(master, text="OK", command=self.edit_nama)
        self.button.pack()

    def edit_nama(self):
        showinfo(message="Halo, {}!".format(self.nama.get()))

root = Tk()
my_gui = Formulirku(root)
root.mainloop()
```

Live Coding: Entry Widget

```
...
        self.nama = StringVar()
        self.field_nama = Entry(master, textvariable=self.nama,
width=40)
        self.field_nama.pack()
...
    def edit_nama(self):
        showinfo(message="Halo, {}!".format(self.nama.get()))
...
```

Live Coding: Entry Widget



Widget Variables

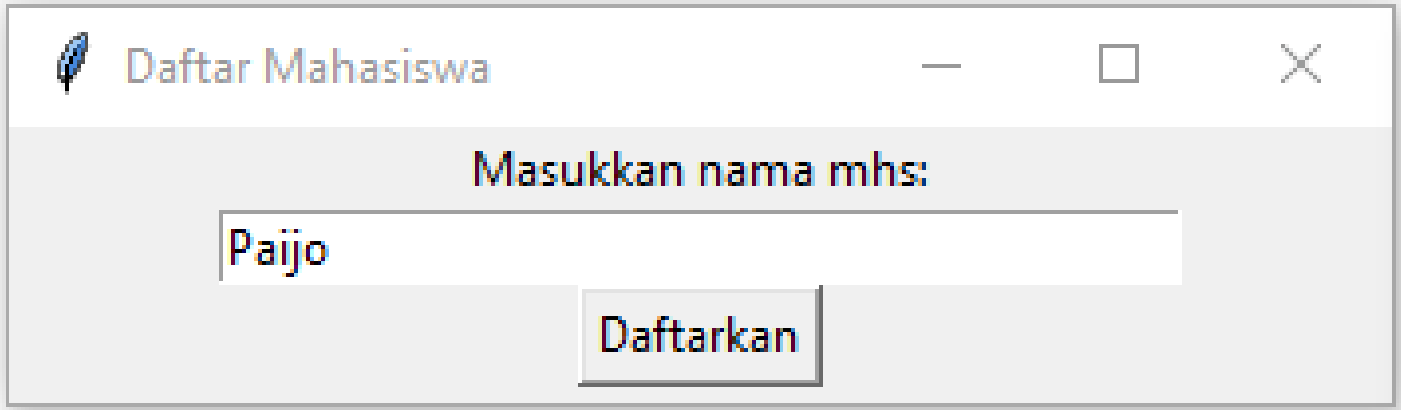
- Beberapa widget memiliki variabel spesial bawaan tkinter
- Nilai dari variabel itu dapat diatur via **.set(new_value)** dan dapat diambil via **.get()**

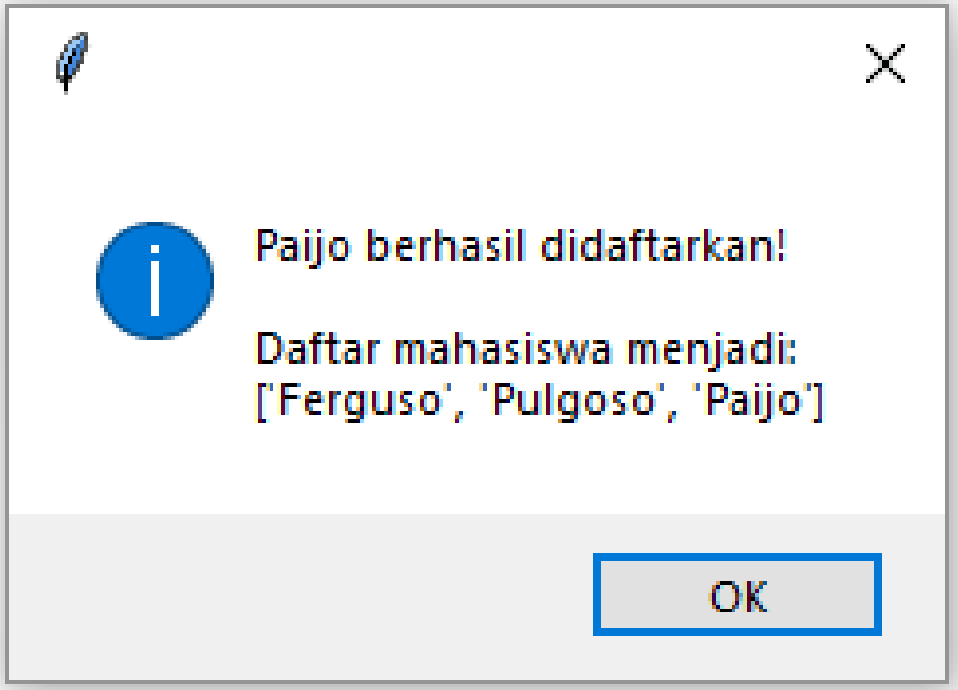
`BooleanVar`	A tk object that holds a single Boolean value
`IntVar`	A tk object that holds a single integer value
`DoubleVar`	A tk object that holds a single double value
`StringVar`	A tk object that holds a single string value

Contoh penggunaan (untuk widget Entry):

```
self.nama = StringVar()
self.field_nama = Entry(master, textvariable=self.nama)
self.field_nama.pack()
```

Live Coding: Daftar Mahasiswa





Live Coding: Daftar Mahasiswa

```
from tkinter import *
from tkinter.messagebox import showinfo

class DaftarMhs:

    daftar_mhs = []

    def __init__(self, master):
        self.master = master
        master.title("Daftar Mahasiswa")
        master.geometry("350x70")

        self.label = Label(master, text="Masukkan nama mhs:")
        self.label.pack()

        self.nama = StringVar()
        self.field_nama = Entry(master, textvariable=self.nama, width=40)
        self.field_nama.pack()

        self.button = Button(master, text="Daftarkan", command=self.daftar)
        self.button.pack()

    def daftar(self):
        mhs = self.nama.get()
        DaftarMhs.daftar_mhs.append(mhs)
        showinfo(message="{} berhasil didaftarkan!\n\nDaftar mahasiswa menjadi:\n{}".format(mhs,DaftarMhs.daftar_mhs))
        self.nama.set("")

root = Tk()
my_gui = DaftarMhs(root)
root.mainloop()
```

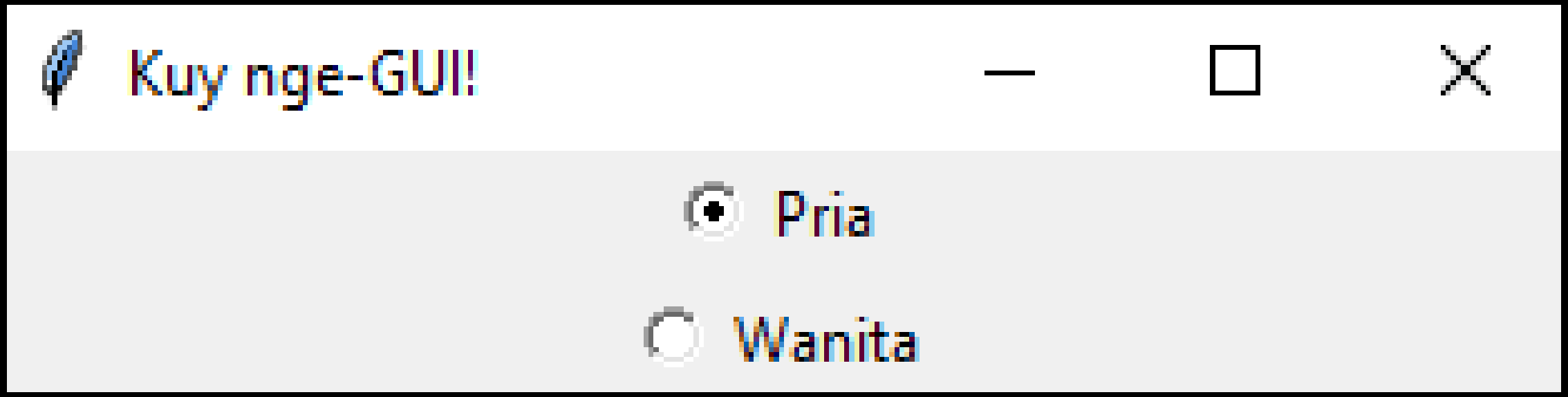
Live Coding: Radiobutton Widget

```
from tkinter import *

class MyFirstGUI:

    def __init__(self, master):
        self.master = master
        master.title("Kuy nge-GUI!")

        self.nilai = IntVar()
        self.nilai.set(0) # atur nilai bawaan/default
        rb_pria = Radiobutton(master, text="Pria", variable=self.nilai, value=0, command=self.gender)
        rb_wanita = Radiobutton(master, text="Wanita", variable=self.nilai, value=1, command=self.gender)
        rb_pria.pack()
        rb_wanita.pack()
        print("Nilai:", self.nilai.get())

    def gender(self):
        print("Nilai:", self.nilai.get())

root = Tk()
my_gui = MyFirstGUI(root)
root.mainloop()
```

Live Coding: Radiobutton Widget

```
...

        self.nilai = IntVar()
        self.nilai.set(0) # atur nilai bawaan/default
        rb_pria = Radiobutton(master, text="Pria",
variable=self.nilai, value=0, command=self.gender)
        rb_wanita = Radiobutton(master, text="Wanita",
variable=self.nilai, value=1, command=self.gender)
        rb_pria.pack()
        rb_wanita.pack()
        print("Nilai:", self.nilai.get())

    def gender(self):
        print("Nilai:", self.nilai.get())
```

Live Coding: Radiobutton Widget



Live Coding: Menampilkan gambar (harus GIF)

```
from tkinter import *

class MyFirstGUI:

    def __init__(self, master):
        self.master = master
        master.title("Kuy nge-GUI!")
        img = PhotoImage(file="gunung.gif")
        self.label = Label(image=img)
        self.label.image = img
        self.label.pack()

root = Tk()
my_gui = MyFirstGUI(root)
root.mainloop()
```

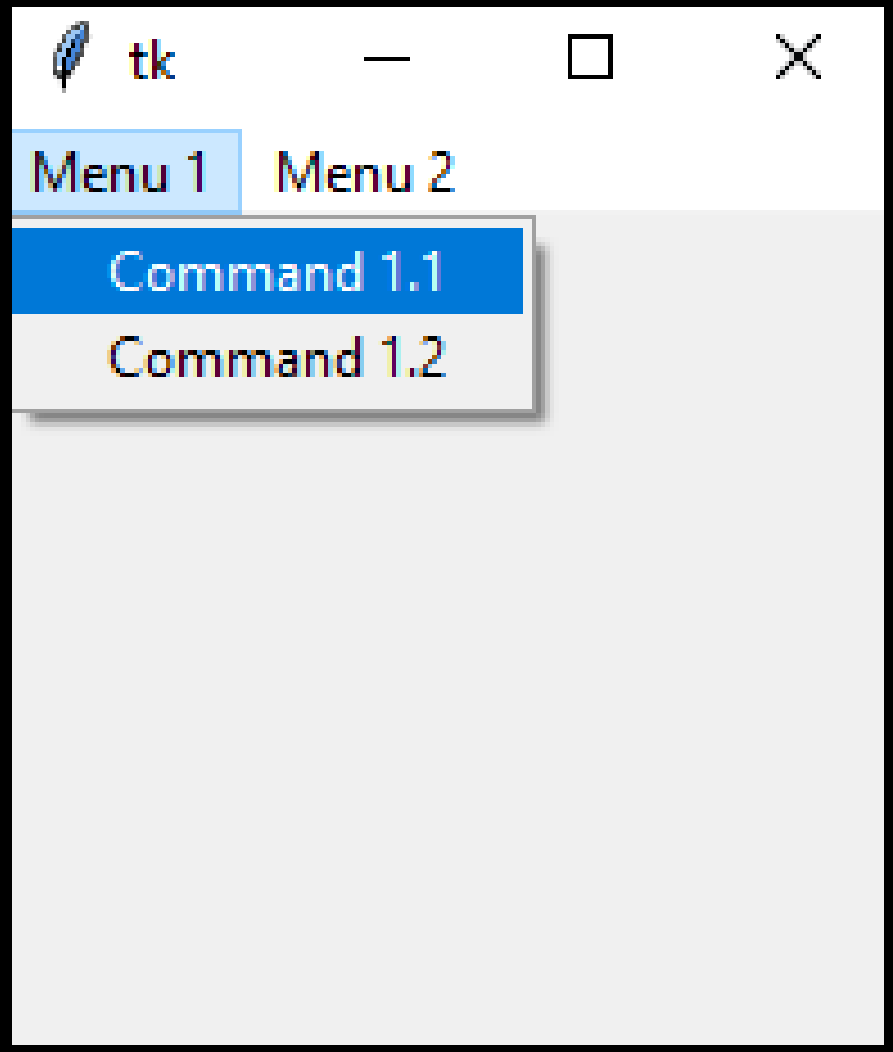
Live Coding: Menampilkan gambar (menggunakan library eksternal PILLOW)

```
from tkinter import *
from PIL import ImageTk, Image

class MyFirstGUI:

    def __init__(self, master):
        self.master = master
        master.title("Kuy nge-GUI!")
        img = ImageTk.PhotoImage(Image.open('myimage.jpg'))
        self.label = Label(image=img)
        self.label.image = img
        self.label.pack()

root = Tk()
my_gui = MyFirstGUI(root)
root.mainloop()
```

Live Coding: Membuat menu

```
from tkinter import *

class MyFirstGUI:

    def __init__(self, master):
        self.master = master
        menubar = Menu(master)
        master['menu'] = menubar

        block_menu = Menu(menubar, tearoff = 0)
        menubar.add_cascade(label = "Menu 1", menu = block_menu)
        block_menu.add_command(label = "Command 1.1")
        block_menu.add_command(label = "Command 1.2")
        block_menu2 = Menu(menubar, tearoff = 0)
        menubar.add_cascade(label = "Menu 2", menu = block_menu2)
        block_menu2.add_command(label = "Command 2.1")

root = Tk()
my_gui = MyFirstGUI(root)
root.mainloop()
```

Live Coding: Membuat menu



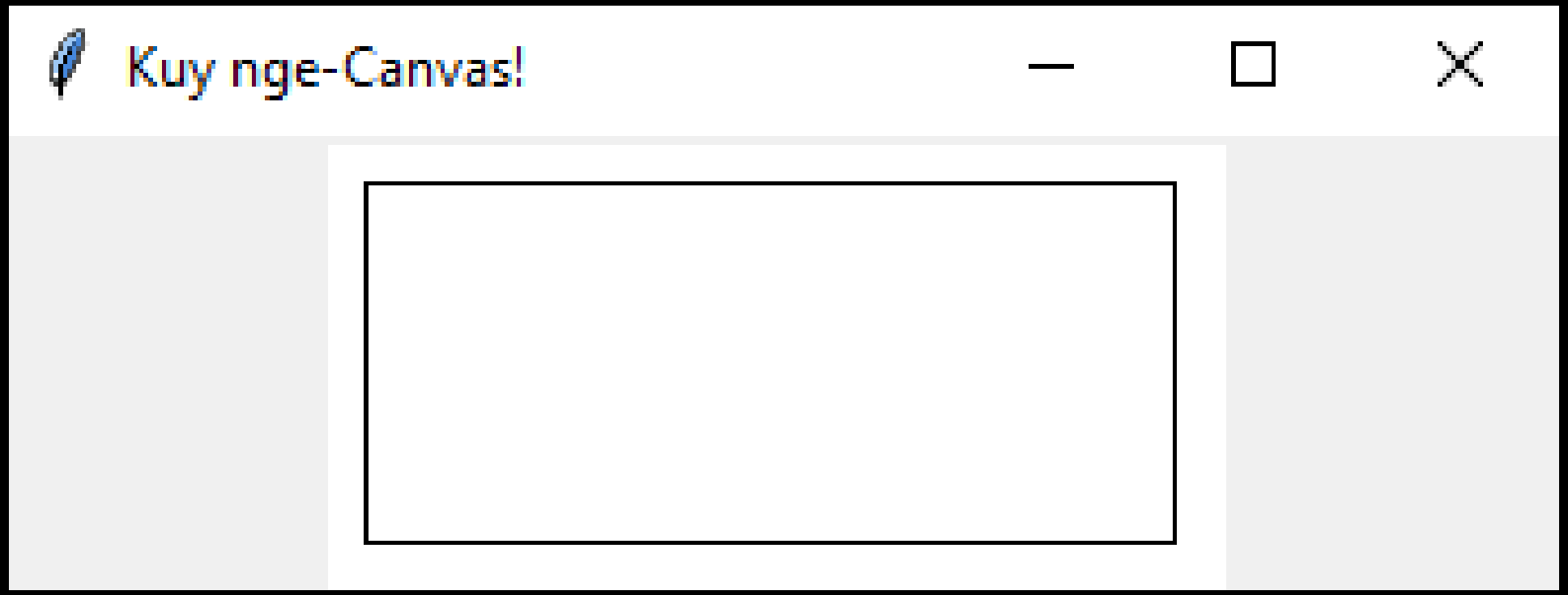
Canvas

- Widget Canvas dapat digunakan untuk menampilkan shapes (bentuk-bentuk):
  - create_rectangle
  - create_oval
  - create_arc
  - create_polygon
  - create_line

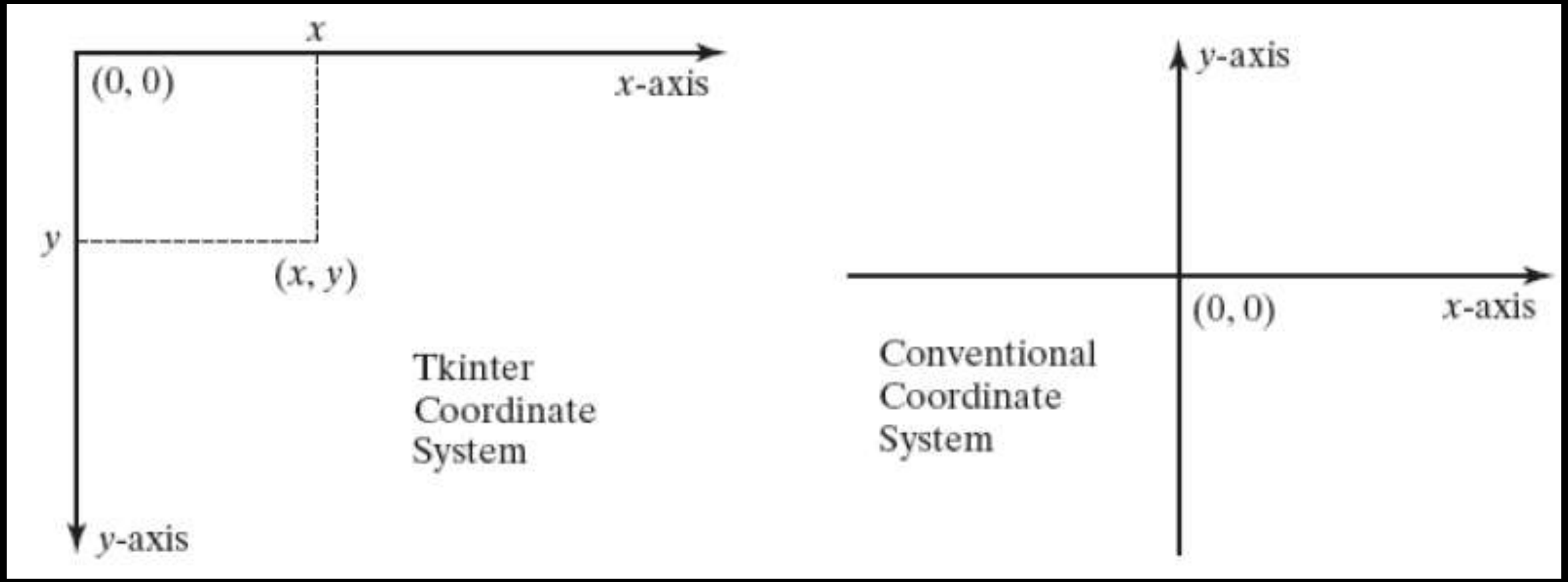
Live Coding: Rectangle

```
from tkinter import *

class MyFirstCanvas:

    def __init__(self, master):
        self.master = master
        master.title("Kuy nge-Canvas!")
        self.canvas = Canvas(master, width=200, height=100,bg="white")
        self.canvas.create_rectangle(10, 10, 190, 90)
        self.canvas.pack()

root = Tk()
my_canvas = MyFirstCanvas(root)
root.mainloop()
```

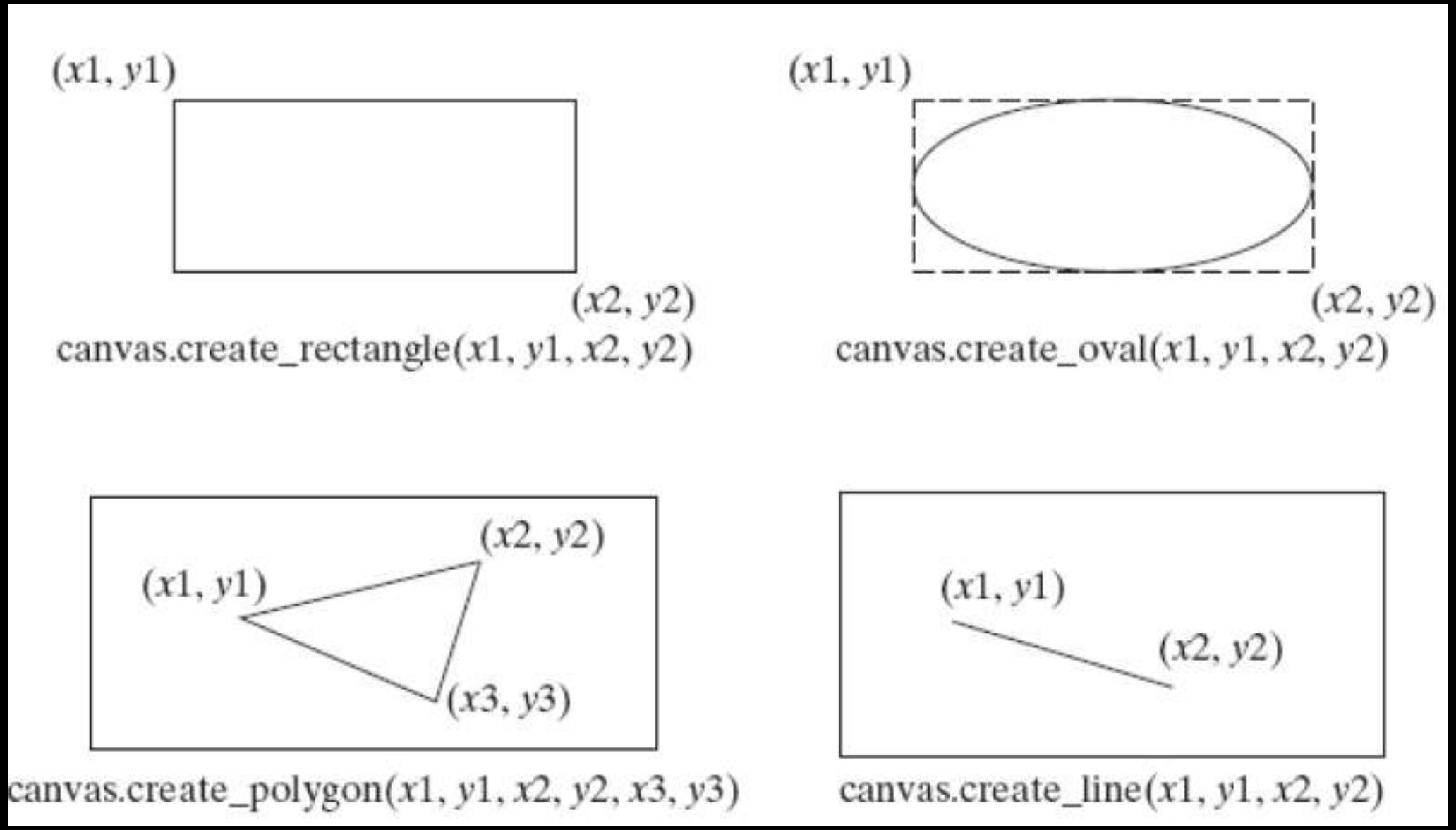
Live Coding: Rectangle



Canvas: Sistem Koordinat



Canvas: Macam Bentuk



Live Coding: Macam Bentuk

```
from tkinter import *

class MyFirstCanvas:

    def __init__(self, master):
        self.master = master
        master.title("Kuy nge-Canvas!")
        self.canvas = Canvas(master, width=1000, height=1000, bg="white")
        self.canvas.create_rectangle(10, 10, 150, 50, fill="red")
        self.canvas.create_oval(10, 60, 100, 100, fill="#FFFF00")
        self.canvas.create_polygon(100, 120, 10, 130, 10, 110, fill="black")
        self.canvas.create_line(10, 140, 500, 140)
        self.canvas.pack()

root = Tk()
my_canvas = MyFirstCanvas(root)
root.mainloop()
```

Live Coding: Widget Text

```
from tkinter import Tk, Text, BOTH

def record(event):
    '''event handling function for key press events;
       input event is of type tkinter.Event'''
    print('char = {}'.format(event.keysym)) # print key symbol

root = Tk()

text = Text(root,
            width=20,  # set width to 20 characters
            height=5)  # set height to 5 rows of characters

# Bind a key press event with the event handling function record()
text.bind('<KeyPress>', record)

# widget expands if the master does
text.pack(expand=True, fill=BOTH)

root.mainloop()
```

Event pattern and `tkinter` class `Event`

Type	Description
`Button`	Mouse button
`Return`	Enter/Return key
`KeyPress`	Press of a keyboard key
`KeyRelease`	Release of a keyboard key
`Motion`	Mouse motion
Modifier	**Description**
`Control`	Ctrl key
`Button1`	Left mouse button
`Button3`	Right mouse button
`Shift`	Shift key
Detail	**Description**
`<button number>`	Ctrl key
`<key symbol>`	Left mouse button

The first argument of method `bind()` is the type of event we want to bind

The type of event is described by a string that is the concatenation of one or more event patterns

An event pattern has the form

```
<modifier-modifier-type-detail>
```

- **`<Control-Button-1>`: Hitting Ctrl and the left mouse button simultaneously**
- **`<Button-1><Button-3>`: Clicking the left mouse button and then the right one**
- **`<KeyPress-D><Return>`: Hitting the keyboard key and then Return**
- **`<Button1-Motion>`:** Mouse motion while holding left mouse button

Live Coding: Mouse Clicks

```
from tkinter import *

def hello(event):
    print("Single Click, Button-l")

def quit(event):
    print("Double Click, so let's stop")
    root.destroy()

root = Tk()
widget = Button(root, text='Mouse Clicks')
widget.pack()
widget.bind('<Button-1>', hello)
widget.bind('<Double-1>', quit)
root.mainloop()
```

Live Coding: Mouse Motion

```
from tkinter import *

def motion(event):
  print("Mouse position: (%s %s)" % (event.x, event.y))
  return

master = Tk()
whatever_you_do = "Put some nice quotes here.\n(Quote author)"
msg = Message(master, text = whatever_you_do)
msg.config(bg='lightgreen', font=('times', 24, 'italic'))
msg.bind('<Motion>',motion)
msg.pack()
master.mainloop()
```

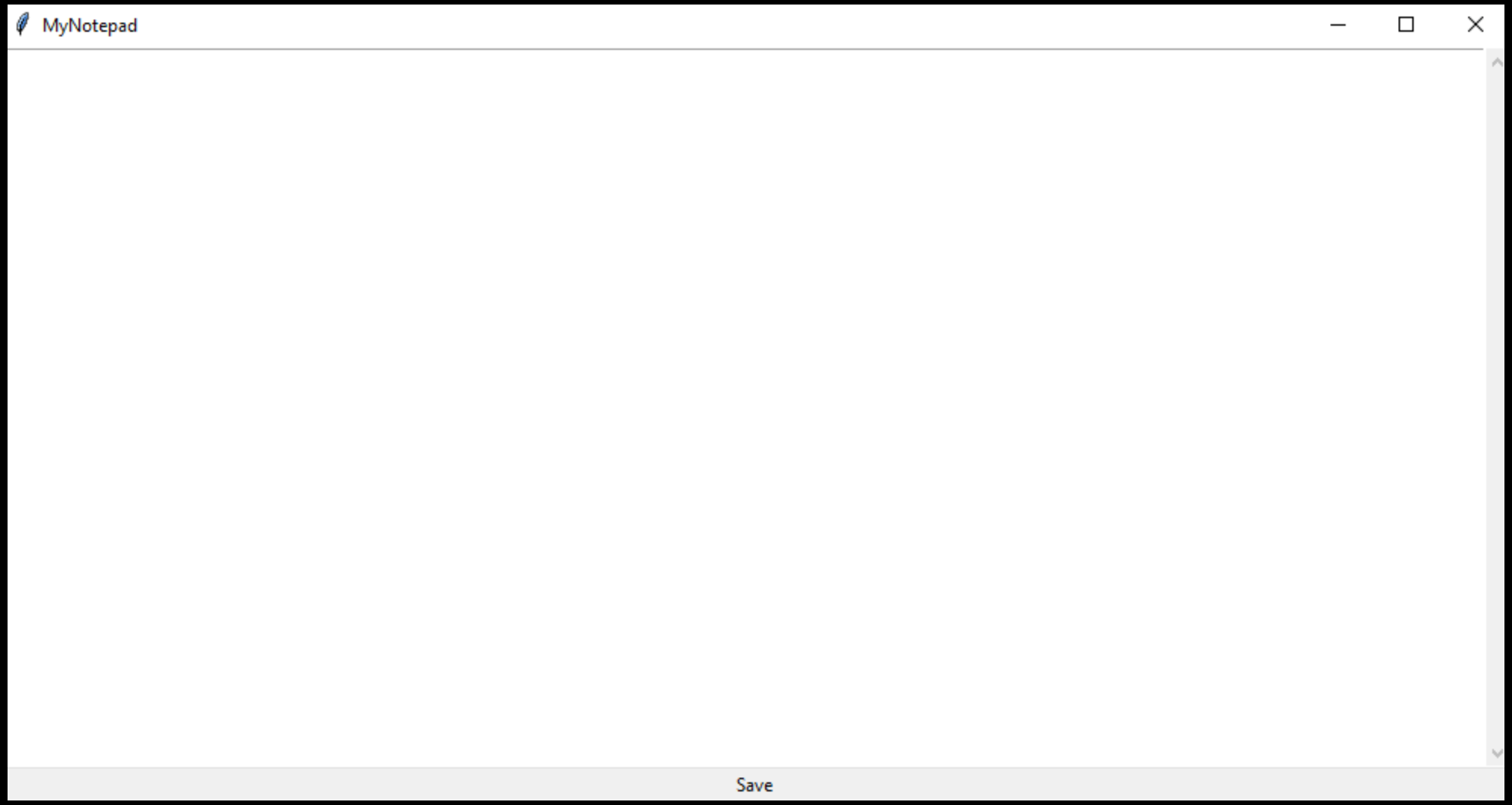
Live Coding: Canvas drawing (1/2)

```
from tkinter import Tk, Canvas

# event handlers begin() and draw() to be defined

root = Tk()
x, y = 0, 0
canvas = Canvas(root, height=100, width=150)

# bind left mouse button click event to function begin()
canvas.bind("<Button-1>", begin)

# bind mouse motion while pressing left button event
canvas.bind("<Button1-Motion>", draw)

canvas.pack()
root.mainloop()
```

Live Coding: Canvas drawing (2/2)

```
# ...

def begin(event):
    global x, y
    x, y = event.x, event.y

def draw(event):
    global x, y, canvas
    newx, newy = event.x, event.y
    # connect previous mouse position to current one
    canvas.create_line(x, y, newx, newy)
    # new position becomes previous
    x, y = newx, newy

# ...
```

Widget ScrolledText - MyNotepad



Live Coding: ScrolledText

```
from tkinter import *
from tkinter.scrolledtext import *

class MyNotepad:

    def __init__(self, master):
        self.master = master
        master.title("MyNotepad")
        master.geometry("1000x500")

        self.txt = ScrolledText(master)
        self.txt.pack(fill=BOTH, expand=True)
        self.txt.focus()

        self.btn = Button(master, text="Save", command=self.save)
        self.btn.pack(fill=X)

    def save(self):
        out = open("MyNotepad.txt", "w")
        # input should be read from line one, char zero, till END
        out.write(self.txt.get("1.0", END))
        out.close()

root = Tk()
my_notepad = MyNotepad(root)
root.mainloop()
```

Geometry (= Layout) Manager

- tkinter menggunakan Geometry Manager untuk mengatur penempatan widget pada suatu container/window
- Ada 3 jenis Geometry Manager:
  pack manager, grid manager, place manager
- Pack manager sudah kita lihat, melalui method **pack()**

Live Coding: Pack Manager fill

```
from tkinter import *

root = Tk()

w = Label(root, text="Red", bg="red", fg="white")
w.pack(fill=X) # coba bandingkan kalau tanpa fill=X
w = Label(root, text="Green", bg="green", fg="black")
w.pack(fill=X)
w = Label(root, text="Blue", bg="blue", fg="white")
w.pack(fill=X)

mainloop()
```

Live Coding: Pack Manager side

```
from tkinter import *

root = Tk()

w = Label(root, text="Red", bg="red", fg="white")
w.pack(side=LEFT)
w = Label(root, text="Green", bg="green", fg="black")
w.pack(side=LEFT)
w = Label(root, text="Blue", bg="blue", fg="white")
w.pack(side=LEFT)

mainloop()
```

Live Coding: Pack Manager side

```
from tkinter import Tk, Label, PhotoImage, BOTTOM, LEFT, RIGHT, RIDGE

class MyPeace:

    def __init__(self, root):
        text = Label(root,
             font=('Helvetica', 16, 'bold italic'),
             foreground='white',
             background='black',
             pady=10,
             text='Peace begins with a smile.')
        text.pack(side=BOTTOM)

        peace = PhotoImage(file='peace.gif')
        peaceLabel = Label(root,
                           borderwidth=3,
                           relief=RIDGE,
                           image=peace)
        peaceLabel.image = peace
        peaceLabel.pack(side=LEFT)

        smiley = PhotoImage(file='smile.gif')
        smileyLabel = Label(root,
                            image=smiley)
        smileyLabel.image = smiley
        smileyLabel.pack(side=RIGHT)

root = Tk()
my_peace = MyPeace(root)
root.mainloop()
```

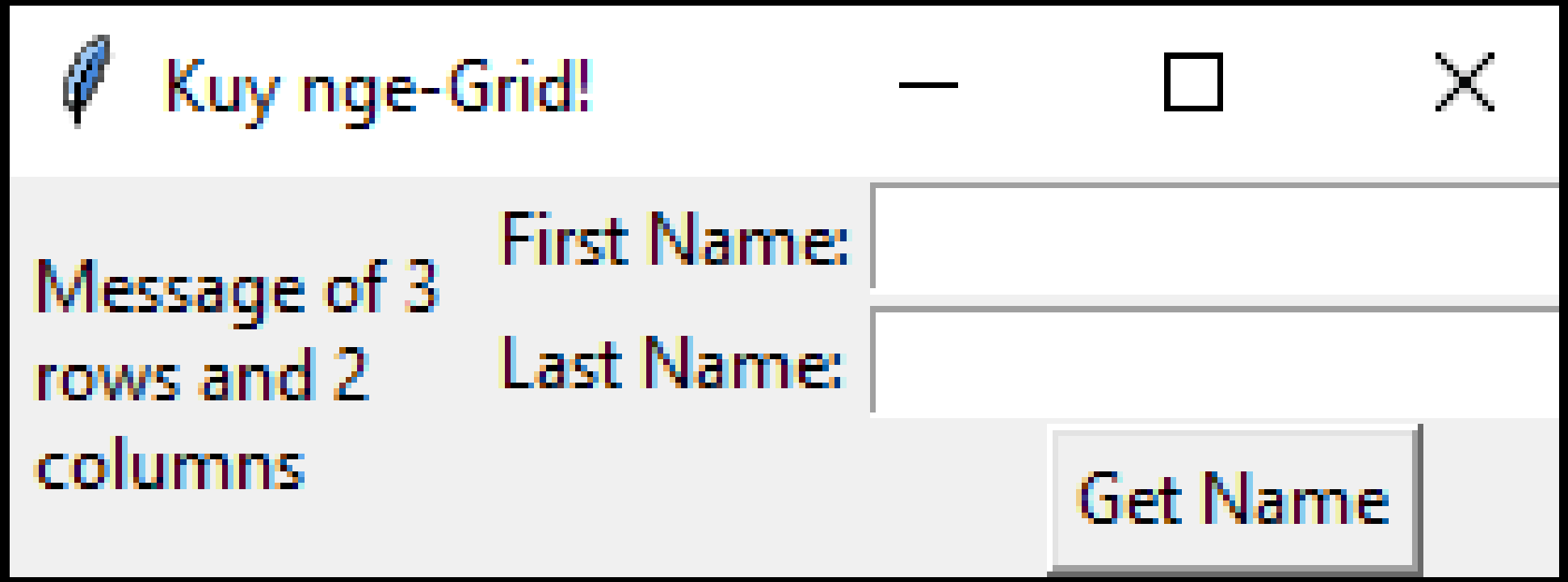
Selain Pack Manager, Ada Grid Manager

- Grid manager menggunakan sistem tabel
- Widget dapat ditempatkan pada **cell di baris dan kolom tertentu**
- Bisa digunakan rowspan dan columnspan sehingga widget dapat menempati beberapa baris dan kolom sekaligus

Live Coding: Grid Manager

```
from tkinter import *

class MyGrid:

    def __init__(self, master):
        self.master = master
        master.title("Kuy nge-Grid!")

        message = Message(master, text = "Message of 3 rows and 2 columns")
        message.grid(row = 1, column = 1, rowspan = 3, columnspan = 2)
        Label(master, text = "First Name:").grid(row = 1, column = 3)
        Entry(master).grid(row = 1, column = 4)
        Label(master, text = "Last Name:").grid(row = 2, column = 3)
        Entry(master).grid(row = 2, column = 4)
        Button(master, text = "Get Name").grid(row = 3, column = 4)

root = Tk()
my_grid = MyGrid(root)
root.mainloop()
```

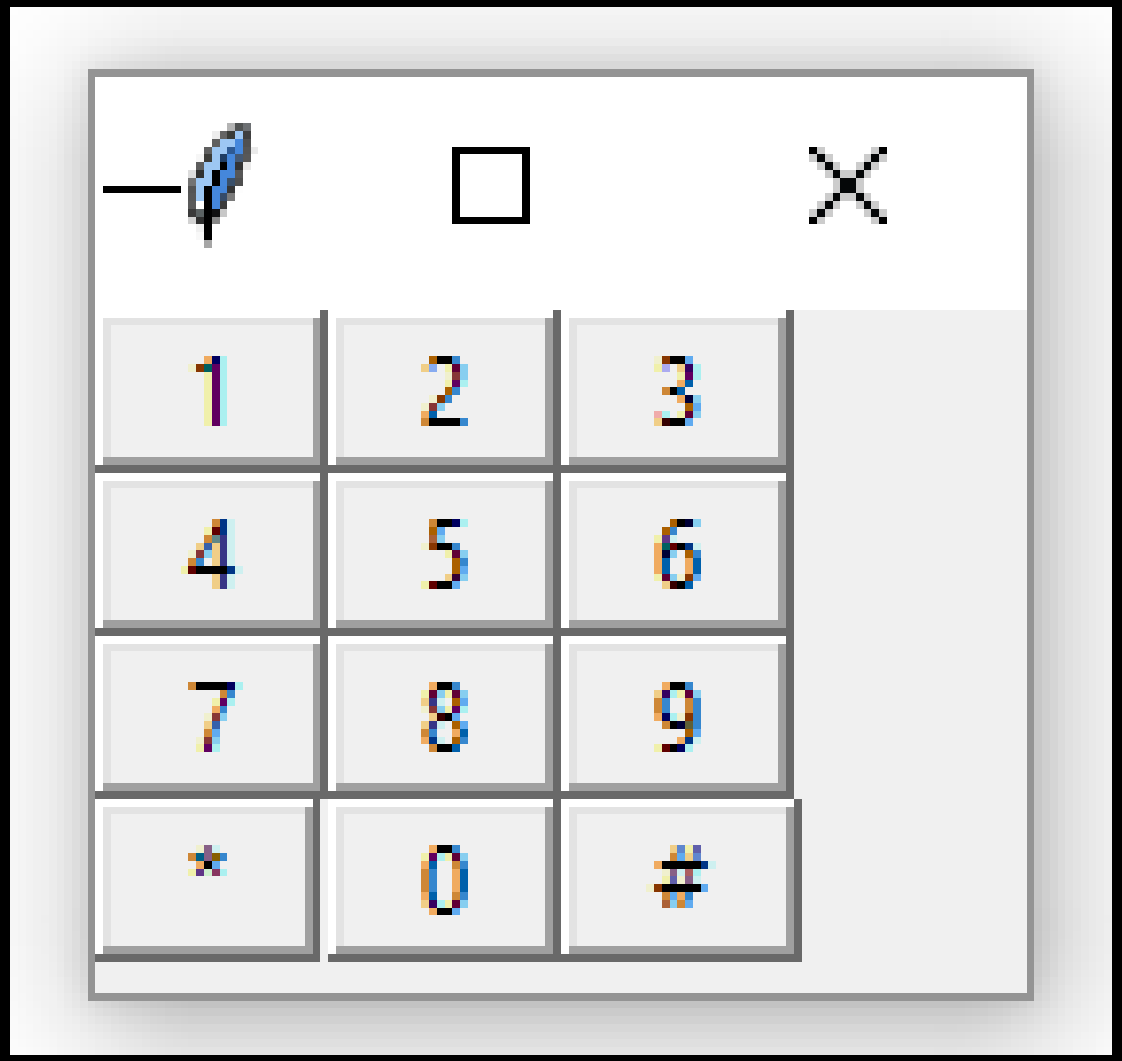
Live Coding: Grid Manager



Live Coding: Grid Manager

```
from tkinter import Tk, Label, RAISED, Button

class Keypad:

    def __init__(self, root):
        labels = [['1', '2', '3'],
          ['4', '5', '6'],
          ['7', '8', '9'],
          ['*', '0', '#']]

        for r in range(4):
            for c in range(3):
                # create button for row r and column c
                # yet still without event handler (pls add yourself)
                button = Button(root,
                              relief=RAISED,
                              padx=10,
                              text=labels[r][c])
                # place label in row r and column c
                button.grid(row=r, column=c)

root = Tk()
keypad = Keypad(root)
root.mainloop()
```

Live Coding: Grid Manager



Live Coding: Place Manager

```
from tkinter import *

root = Tk()

w = Label(root, text="Red", bg="red", fg="white")
w.place(x=0,y=0)
w = Label(root, text="Green", bg="green", fg="white")
w.place(x=100,y=0)
w = Label(root, text="Blue", bg="blue", fg="white")
w.place(x=50,y=50)

mainloop()
```

Perbandingan antara Geometry/Layout Manager

Layout Manager	Description
place	You specify the exact size and position of each widget.
pack	You specify the size and position of each widget *relative to each other*.
grid	You place widgets in a cell of a 2-dimensional table defined by rows and columns.

Pesan untuk Dibungkus

- GUI memudahkan interaksi pengguna
- GUI terdiri atas widgets (button, menu, dsb)
- Widgets tersebut dapat diberi event listener (penanganan event)
- Widgets tersebut juga dapat dimodifikasi (misalkan teksnya, warnanya, dsb)
- Geometry/layout manager untuk mengatur penempatan widgets