Materi Lab 10

Graphical User Interface (GUI)

GUI

***Graphical User Interface* (GUI)** adalah suatu antarmuka pengguna (*user interface*) di mana pengguna berinteraksi melalui objek-objek grafis (yang disebut *widgets*), seperti tombol (button), checkbox, menu, dsb. Dalam bahasa Python, kita dapat memanfaatkan modul **tkinter** untuk membuat GUI.

tkinter

tkinter merupakan module GUI bawaan[1] Python yang memanfaatkan toolkit GUI Tk. **tkinter** bersifat *cross-platform* dan dapat digunakan di Windows, Linux, maupun macOS. Modul ini menyediakan berbagai elemen-elemen GUI (*widgets*) seperti tombol, menu, entri teks (*text field*), dsb. **tkinter** juga dapat digunakan untuk menerapkan *event-driven programming*, di mana GUI akan merespon ke berbagai event, misalnya aksi-aksi pengguna GUI seperti klik, ketik, *scroll*, dsb.

Di dalam modul **tkinter**, terdapat berbagai class yang dapat digunakan untuk membuat GUI. Class **Tk**, misalnya, dapat digunakan untuk membuat window untuk menampung *widgets*.

[1] Apabila mendapatkan *exception* “ModuleNotFoundError”, kemungkinan kalian menggunakan Python 2 untuk menjalankan kode kalian. Apabila masih mendapat error yang sama setelah berganti ke Python 3, maka kalian perlu melakukan instalasi modul tkinter, silahkan ikuti tutorial berikut ini:
Linux (https://tkdocs.com/tutorial/install.html#installlinux)
MacOS (https://tkdocs.com/tutorial/install.html#installmac)
Windows (https://tkdocs.com/tutorial/install.html#installwin)

Coba jalankan kode berikut, kemudian pahami dan modifikasi sesuka hati kalian.

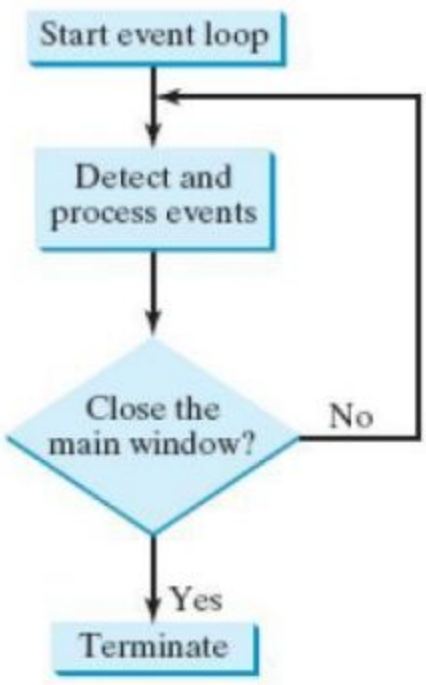
contohGUI.py

```
import tkinter as tk

class MyFirstGUI:
  def __init__(self, master):
    self.master = master
    # Mengubah judul window
    self.master.title("Kuy nge-GUI!")
    # Mengatur ukuran window
    # Lebar: 500, tinggi: 100
    self.master.geometry("500x100")

    # Label adalah salah satu widget tkinter
    # Argumen pertama pada widget constructor selalu parent container
    self.label1 = tk.Label(master, text="Contoh GUI Python")
    self.label1.pack() # Menempatkan label di container

    self.greet_button = tk.Button(master, text="Sapa",
                                  command=self.sapa, fg="green")
    self.greet_button.pack()

    self.close_button = tk.Button(master, text="Keluar",
                          command=master.destroy, bg="red", fg="white")
    self.close_button.pack()

  def sapa(self):
    print("Halo!")

root = tk.Tk() # Membuat window
my_gui = MyFirstGUI(root)
root.mainloop() # Menjalankan event loop-nya tkinter
```

Event processing

Kita dapat memanfaatkan **tkinter** untuk menerapkan ***event-driven programming***, yang berarti program yang kita buat akan merespon terhadap *event*, pada umumnya berupa aksi pengguna. Hal ini dispesifikasikan dalam *method mainloop()*. Method *mainloop()* membuat suatu *event loop*, yang akan memproses berbagai event terus menerus sampai *window*-nya ditutup.

Berikut adalah ilustrasi dari sebuah *event loop*.



Suatu widget dapat diberi *event handler* (atau *callback function*) yang akan dieksekusi apabila suatu event terhadap widget itu terjadi, misalnya pada saat pengguna melakukan klik pada suatu tombol.

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

Mengubah atribut suatu widget

Ketika membuat (*construct*) suatu widget, kita bisa memberi spesifikasi untuk atributnya, seperti *text*, *command*, dsb. Atribut-atribut tersebut dapat **diubah** dengan pendekatan ala *dictionary*:

```
widget_name["nama_atribut"] = nilai_baru_dapat_berupa_apapun
```

Ubahlah fungsi `sapa()` pada contohGUI.py dengan instruksi di bawah ini, lalu jalankan dan lihat perubahannya.

```
def sapa(self):
  self.label1["text"] = "Halo!"
```

Berbagai widget tkinter

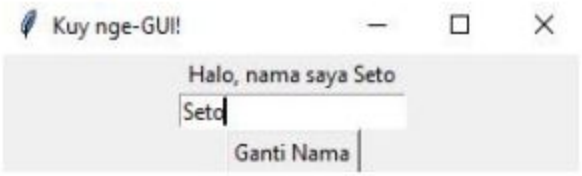
Berikut adalah beberapa ***widget*** yang tersedia di tkinter.

Widget	***Deskripsi***
Button	Tombol sederhana, digunakan untuk menjalankan perintah.
Canvas	Area untuk menampilkan elemen grafis seperti garis, segi empat, atau teks.
Checkbutton	*Check box* untuk men-toggle antara dua nilai.
Entry	Entri teks, disebut juga *text field* atau *text box*
Frame	*Container* untuk *widget* lainnya
Label	Menampilkan teks atau gambar
Menu	Panel menu, untuk mengimplementasikan menu *pull-down* atau *popup*
MenuBottom	Tombol menu, untuk mengimplementasikan menu *pull-down*
Message	Mirip dengan *Label*, tetapi bisa otomatis *wrap* sesuai ukuran tertentu.
Radiobutton	Menetapkan nilai variabel ke tombolnya dan mengosongkan *Radiobutton* lainnya.

Text	Tampilan teks yang dapat diberi berbagai format.

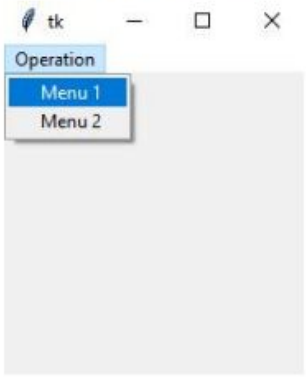
Berikut adalah ilustrasi untuk sebagian widget di atas.

- `Entry`

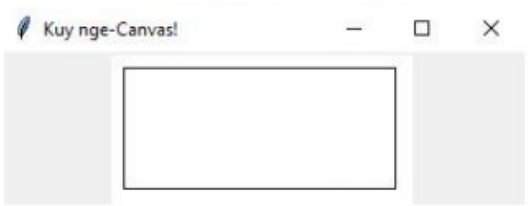


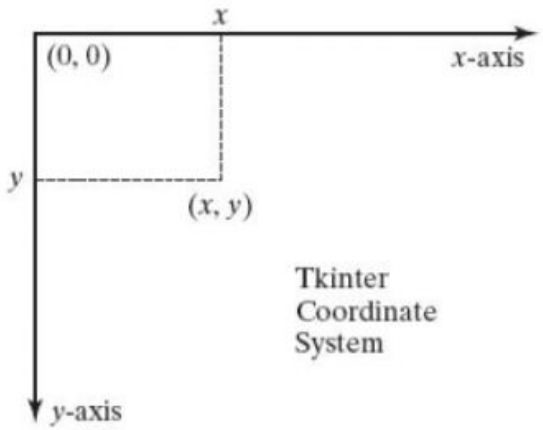
- `Radiobutton`

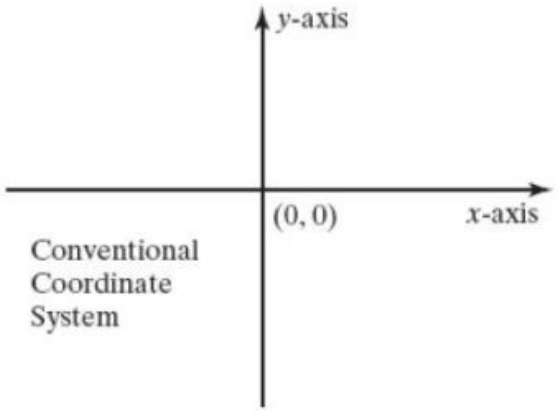
- `Menu` dan `Menubutton`

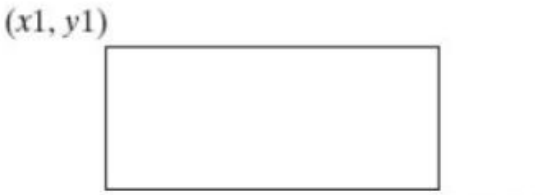


- `Canvas` dan sistem koordinatnya



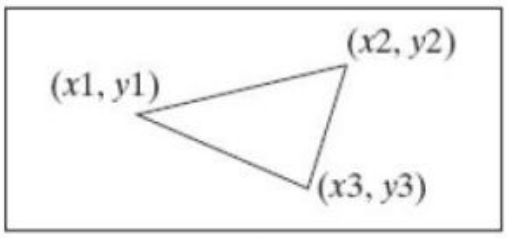




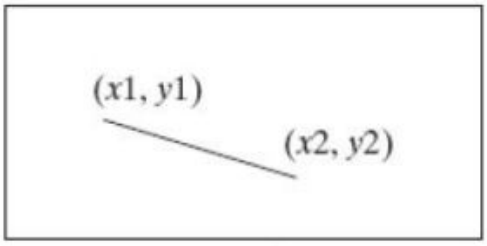


canvas.create_rectangle(x1, y1, x2, y2)

canvas.create_oval(x1, y1, x2, y2)



canvas.create_polygon(x1, y1, x2, y2, x3, y3)



canvas.create_line(x1, y1, x2, y2)

Message widget

W*idget* **Message** mirip dengan *widget* Label 1, hanya saja ia lebih fleksibel. *Widget* ini akan otomatis melakukan *text-wrapping* dan mengatur lebarnya untuk menjaga rasio tertentu.

Contoh:

messageVsLabelGUI.py
`import tkinter as tk`

```
class MessageVsLabel:
  def __init__(self, master):
    self.master = master
    self.master.title("Message vs Label, apa bedanya?")
    text = (
      "Sebuah teks yang panjang sekali sampai-sampai saya "
      "lelah membacanya karena sebenarnya teks ini tidak memiliki "
      "makna sama sekali."
    )

    self.message1 = tk.Message(master, text="Message: "+text,
                      font="Helvetica 20 bold", width=500)
    self.message1.pack()

    self.label1 = tk.Label(master, text="Label: "+text,
                      font="Helvetica 20 bold", width=500)
    self.label1.pack()

root_window = tk.Tk()
my_gui = MessageVsLabel(root_window)
root_window.mainloop()
```

Coba jalankan kode di atas dan bandingkan penggunaan **tk.Message** dengan **tk.Label** !

Submodule filedialog

Modul **tkinter** mempunyai sebuah *submodule* bernama **filedialog**. Modul ini dapat digunakan untuk membuat kotak-kotak dialog yang berkaitan dengan berkas agar program kita lebih interaktif. Salah satu *method* yang akan kita manfaatkan dari submodul ini adalah method **askopenfilename**.

Method ini akan mengembalikan nama berkas yang dipilih oleh pengguna (beserta lokasinya). Nama berkas ini kemudian bisa kita gunakan untuk membuka dan mengolah berkas tersebut dalam kode program kita.

Berikut adalah contoh penggunaan method **askopenfilename**.

```
from tkinter import filedialog

# Fungsi ini ditaruh di dalam class
def load_file(self):
  file_name = filedialog.askopenfilename()
  if not file_name: # Jika pengguna membatalkan dialog, langsung return
    return
  text_file = open(file_name, 'r', encoding="utf-8")
  result = text_file.readlines()
  text_file.close()
  return result
```

Geometry manager

tkinter menggunakan ***geometry manager*** untuk mengatur penempatan widget pada suatu *container/window*. Ada beberapa jenis *geometry manager* pada **tkinter**, seperti *grid manager*, *pack manager*, dan *place manager*.

Geometry manager

Grid manager menempatkan *widget* pada suatu "tabel". Widget dapat ditempatkan pada *cell* di baris dan kolom tertentu. Kita bisa menggunakan **rowspan** dan **columnspan** agar suatu widget dapat menempati beberapa baris dan kolom.

Perhatikan contoh berikut.

gridGUI.py

```
import tkinter as tk

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

```
    message = tk.Message(master, text="Message of 3 rows and 2 columns")
    message.grid(row=0, column=0, rowspan=3, columnspan=2)
    tk.Label(master, text="First Name:").grid(row=0, column=2)
    tk.Entry(master).grid(row=0, column=3)
    tk.Label(master, text="Last Name:").grid(row=1, column=2)
    tk.Entry(master).grid(row=1, column=3)
    tk.Button(master, text="Get Name").grid(row=2, column=3)

root_window = tk.Tk()
my_gui = MyGrid(root_window)
root_window.mainloop()
```

Hasil:

Event Binding

tkinter menyediakan mekanisme agar developer dapat dengan mudah menangkap event-event yang terjadi. Event yang dimaksud seperti user menekan tombol enter, Ctrl + C, Left mouse click, dsb. Secara general, syntax yang digunakan sebagai berikut:

```
widget_name.bind(event, handler)
```

event haruslah berupa string, menandakan hal yang ingin ditangkap. Selengkapnya mengenai event yang dapat dimanfaatkan, Anda dapat membaca artikel berikut ini. Sedangkan **handler** merupakan sebuah fungsi yang akan dijalankan apabila event yang diinginkan tertangkap.

Untuk memahami lebih lanjut mengenai event binding, silakan amati contoh kode berikut ini:

eventBindingExample.py

```
import tkinter as tk

class eventBindings:
  def __init__(self, master):
    self.master = master
    self.master.title("Belajar Event Bindings")
    self.label1 = tk.Label(master, text="Contoh Event Binding")
    self.label1.pack()
    self.entry1 = tk.Entry(master)
    self.entry1.pack()

    # Apabila menekan Left Shift + C saat fokus di window utama
    self.master.bind("<Shift_L><C>", self.tanya)

    # Apabila menekan tombol enter saat fokus di entry1
    self.entry1.bind("<Return>", self.sapa)

  def sapa(self, event):
    self.label1["text"] = "Halo! " + self.entry1.get()

   def tanya(self, event):
    self.label1["text"] = "Siapa namamu?"

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

Untuk mempelajari lebih lanjut tentang GUI menggunakan tkinter, silahkan buka dokumentasinya dan referensi ini.