

# Bab 8. Graphical User Interface (GUI)

## OBJEKTIF:

1. Mahasiswa Mampu Memahami GUI pada Python.
2. Mahasiswa Mampu Menggunakan Module TKinter untuk Membuat Program GUI pada Python

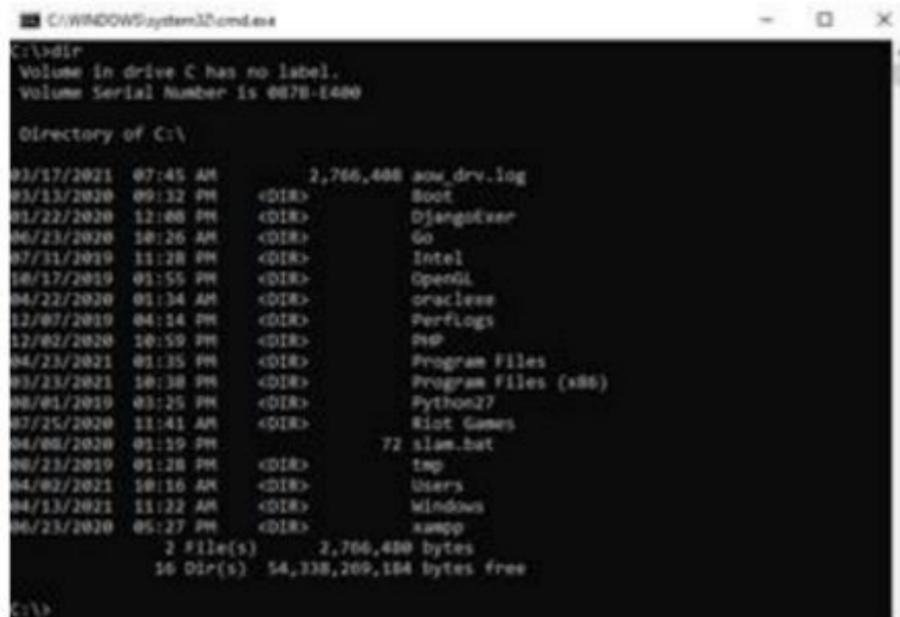
## 8.1 Graphical User Interface (GUI)

*Graphical User Interface* (GUI) atau antarmuka pengguna grafis ialah bagian yang digunakan pengguna untuk berinteraksi dengan komputer. GUI terdiri dari perangkat keras dan lunak, seperti *keyboard*, *mouse*, monitor, serta sistem operasi untuk menerima perintah pengguna. GUI membuat pengguna dapat berinteraksi dengan komputer melalui elemen-elemen grafis seperti ikon, tombol, dan kotak dialog.



Sebelum ditemukannya GUI, pengguna komputer harus menggunakan *Command Line Interface* (CLI) atau antarmuka baris perintah dalam berinteraksi dengan sistem operasi komputer. Banyak pengguna merasa bahwa CLI sulit untuk digunakan karena beberapa hal, seperti:

1. Banyak sekali perintah yang harus dipelajari.
2. Perintah memiliki sintaks-nya sendiri sehingga mirip statement pemrograman.



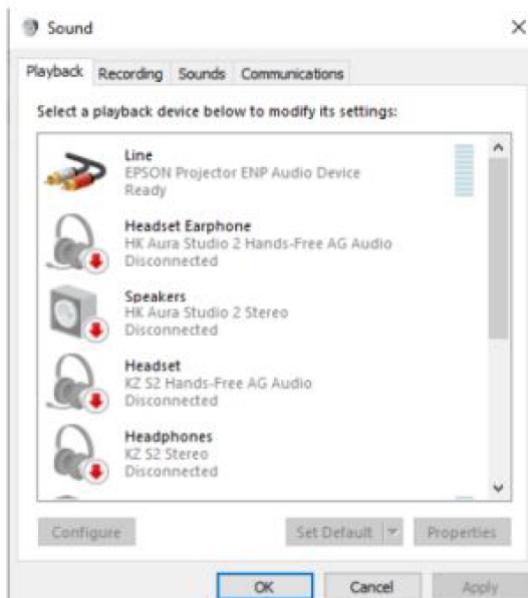
```
C:\>dir
Volume in drive C has no label.
Volume Serial Number is 007B-E400

Directory of C:\

03/17/2021  07:45 AM           2,766,488 aow_drv.log
03/13/2020  09:32 PM        <DIR>    Boot
03/22/2020  12:00 PM        <DIR>    DjangoEver
06/23/2020  10:26 AM        <DIR>    Go
07/31/2019  11:28 PM        <DIR>    Intel
08/17/2019  01:55 PM        <DIR>    OpenGL
04/22/2020  01:34 AM        <DIR>    oraclese
12/07/2019  04:14 PM        <DIR>    PerfLogs
12/02/2020  10:59 PM        <DIR>    PHP
04/23/2021  01:35 PM        <DIR>    Program Files
03/23/2021  10:38 PM        <DIR>    Program Files (x86)
08/01/2019  03:25 PM        <DIR>    Python27
07/25/2020  11:41 AM        <DIR>    Riot Games
04/08/2020  01:59 PM           72 slam.bat
08/23/2019  01:28 PM        <DIR>    tmp
04/02/2021  10:16 AM        <DIR>    Users
04/13/2021  11:22 AM        <DIR>    Windows
06/23/2020  05:27 PM        <DIR>    xampp
               2 File(s)      2,766,488 bytes
              16 Dir(s)   54,338,269,184 bytes free

C:\>
```

Dalam GUI, pengguna menentukan urutan program yang terjadi. Misalkan, kita mempunyai sebuah *file* program GUI



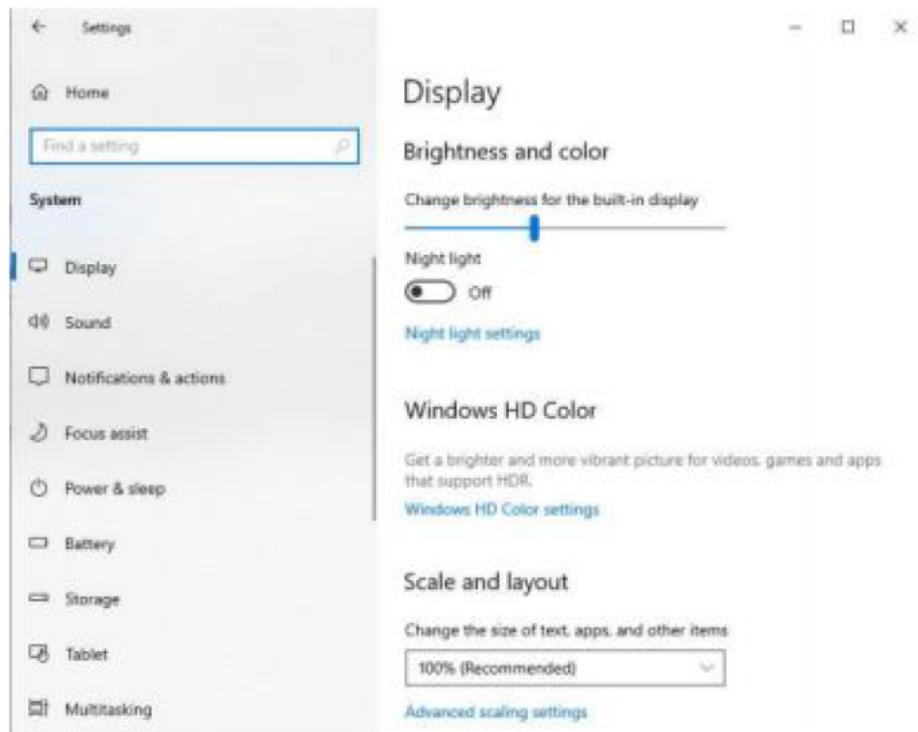
Contoh Dialog Box GUI

## 8.2 Module Tkinter

Python tidak memiliki fitur pemrograman GUI bawaan. Terdapat sebuah *module* dari pihak ketiga yang dapat membantu kita dalam membuat sebuah program GUI sederhana, salah satunya bernama tkinter. Tkinter merupakan kependekan dari TK *Interface*. Nama ini muncul karena *module* ini membuat pemrograman Python dapat menggunakan *library* GUI bernama Tk.

### 8.2.1 Widget pada Tkinter

Program GUI menampilkan sebuah *window* atau wadah jendela dengan berbagai macam *widget*. *Widget* ialah sebuah komponen dalam program GUI yang dapat digunakan oleh pengguna sebagai elemen untuk interaksi dengan program atau sebagai sumber informasi program.



Terdapat 15 *widget* yang dapat kita gunakan. Berikut ini adalah tabel penggunaan *widget* pada tkinter:

Widget	Keterangan
Button	Tombol yang akan menjalankan perintah ketika di-klik
Canvas	Area persegi yang dapat digunakan untuk menampilkan grafis
Checkbutton	Tombol yang digunakan untuk mengubah posisi 'on' atau 'off'
Entry	Area dimana pengguna dapat memasukkan <i>input</i> berupa sebuah baris dari <i>keyboard</i>
Frame	Wadah yang digunakan untuk menyimpan <i>widget</i> lain
Label	Area yang menampilkan sebaris teks atau sebuah gambar
Listbox	Daftar objek yang dapat pengguna pilih
Menu	Daftar menu pilihan yang akan ditampilkan ketika di-klik
Menubutton	Menu yang ditampilkan dilayar dan dapat di-klik oleh pengguna
Message	Menampilkan beberapa baris teks
Radiobutton	Tombol yang dapat dipilih atau dibatalkan pilihannya
Scale	Widget yang dapat digunakan untuk mengubah nilai dengan menggerakkan <i>slider</i> sesuai trek
Scrollbar	Menambahkan kemampuan <i>widget</i> untuk menggulir
Text	Area yang dapat diisikan beberapa baris teks
Toplevel	Wadah seperti <i>frame</i> tetapi ditampilkan didalam <i>window</i> -nya tersendiri

## 8.2.2 Program GUI pada Tkinter

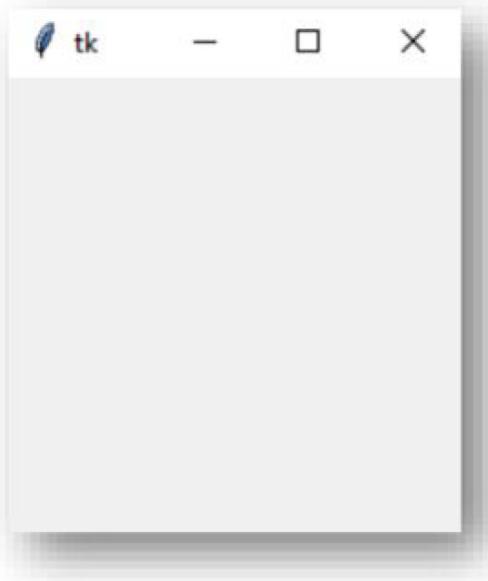
Pembuatan program GUI menggunakan tkinter pada Python dapat dilakukan dengan sederhana. Berikut ini contoh programnya:

```
# tk_fungsi.py
# Import modul tkinter
import tkinter

def main():
    # Membuat widget window utama GUI
    wind_utama = tkinter.Tk()
    # Masuk ke dalam loop utama tkinter
    tkinter.mainloop()

# Panggil fungsi main
main()
```

Output:



Catatan:

Program di atas merupakan GUI kosong yang dibuat menggunakan tkinter. *Widget* di atas masih kosong dan belum memproses perintah apapun. Jika ingin keluar dari program, dapat klik tombol tutup (x).

Pembuatan program GUI pun dapat dibentuk dengan menggunakan paradigma pemrograman orientasi objek. Dibandingkan menggunakan fungsi untuk membuat elemen tampilan dari program, membuat kelas dengan metode `__init__` di dalamnya lebih umum digunakan. Ketika instans kelas dibuat, program GUI akan ditampilkan pada layar. Perbedaan mendasar dari penggunaan paradigma orientasi objek ialah pada proses pembuatan program yang menjadi lebih mudah dan terstruktur.

Berikut ini adalah contoh pembuatan program GUI menggunakan paradigma pemrograman orientasi objek:

```
# tk_kelas.py
# Import modul tkinter
import tkinter
```

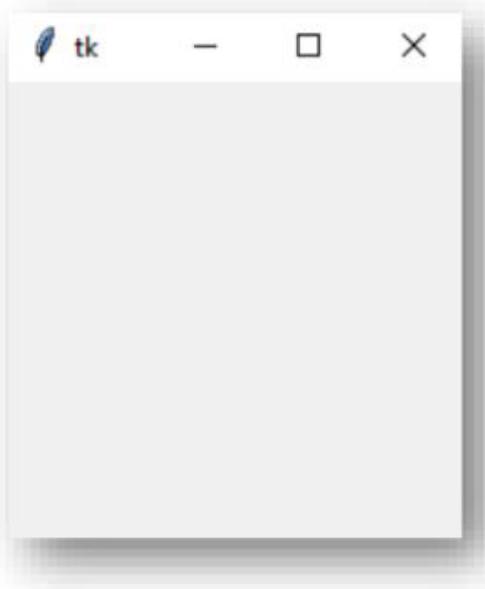
```

# Membuat kelas
class MyGUI:
    def __init__(self):
        # Membuat widget window utama
        self.main_window = tkinter.Tk()
        # Memasukkan loop utama tkinter
        tkinter.mainloop()

# Membuat instans untuk kelas MyGUI
my_gui = MyGUI()

```

*Output:*



### 8.2.3 Menampilkan Teks dengan *Widget Label*

Kita dapat menggunakan *widget label* untuk menampilkan sebaris teks dalam sebuah *window*. Berikut ini adalah contoh programnya:

```

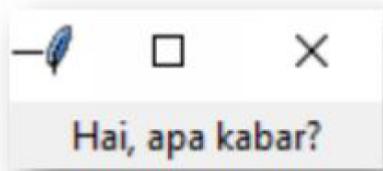
# Labelwidget.py
# Import modul tkinter
import tkinter

class MyGUI:
    def __init__(self):
        # Membuat widget window utama
        self.main_window = tkinter.Tk()
        # Membuat widget label yang menyimpan
        # pesan 'Hai, apa kabar?'
        self.label = tkinter.Label(self.main_window, \
text='Hai, apa kabar?')
        # Panggil metode pack dari label
        self.label.pack()
        # Masuk ke dalam loop utama tkinter
        tkinter.mainloop()

# Membuat instans untuk kelas MyGUI
my_gui = MyGUI()

```

Output:



```
# Labelwidget.py
# Import modul tkinter
import tkinter

class MyGUI:
    def __init__(self):
        # Membuat widget window utama
        self.main_window = tkinter.Tk()
        # Membuat widget label yang menyimpan
        # pesan 'Hai, apa kabar?'
        self.label = tkinter.Label(self.main_window, \
text='Hai, apa kabar?')
        # Panggil metode pack dari label
        self.label.pack()
        # Masuk ke dalam loop utama tkinter
        tkinter.mainloop()

# Membuat instans untuk kelas MyGUI
my_gui = MyGUI()
```

Dengan membuat instans widget `.Label()` dari modul tkinter, kita dapat menambahkan text pesan yang ingin disampaikan

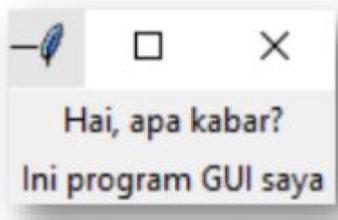
Contoh program menampilkan 2 baris teks pada *widget label*:

```
# Labelwidget2.py
import tkinter

class MyGUI:
    def __init__(self):
        # Membuat widget window utama
        self.main_window = tkinter.Tk()
        # Membuat 2 widget label yang menyimpan
        # pesan 'Hai, apa kabar?'
        self.label1 = tkinter.Label(self.main_window, \
text='Hai, apa kabar?')
        self.label2 = tkinter.Label(self.main_window, \
text='Ini program GUI saya')
        # Panggil metode pack dari label
        self.label1.pack()
        self.label2.pack()
        # Masuk ke dalam loop utama tkinter
        tkinter.mainloop()

# Membuat instans untuk kelas MyGUI
my_gui = MyGUI()
```

Output:



```
# Labelwidget2.py
import tkinter

class MyGUI:
    def __init__(self):
        # Membuat widget window utama
        self.main_window = tkinter.Tk()
        # Membuat 2 widget label yang menyimpan
        # pesan 'Hai, apa kabar?'
        self.label1 = tkinter.Label(self.main_window, \
text='Hai, apa kabar?')
        self.label2 = tkinter.Label(self.main_window, \
text='Ini program GUI saya')
        # Panggil metode pack dari label
        self.label1.pack()
        self.label2.pack()
        # Masuk ke dalam loop utama tkinter
        tkinter.mainloop()

# Membuat instans untuk kelas MyGUI
my_gui = MyGUI()
```

Membuat 2 instans widget .Label() dari modul tkinter

Metode pack() digunakan untuk menyusun peletakan label dalam window, secara default, peletakan secara bertumpuk/vertikal

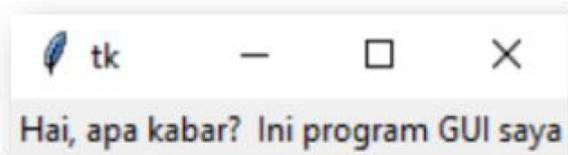
Contoh program menampilkan 2 baris teks pada *widget label*:

```
# Labelwidget3.py
import tkinter

class MyGUI:
    def __init__(self):
        # Membuat widget window utama
        self.main_window = tkinter.Tk()
        # Membuat 2 widget label yang menyimpan
        # pesan 'Hai, apa kabar?'
        self.label1 = tkinter.Label(self.main_window, \
text='Hai, apa kabar?')
        self.label2 = tkinter.Label(self.main_window, \
text='Ini program GUI saya')
        # Panggil metode pack dari kedua label
        self.label1.pack(side='left')
        self.label2.pack(side='left')
        # Masuk ke dalam loop utama tkinter
        tkinter.mainloop()

# Membuat instans untuk kelas MyGUI
my_gui = MyGUI()
```

Output:



```
# Labelwidget3.py
import tkinter

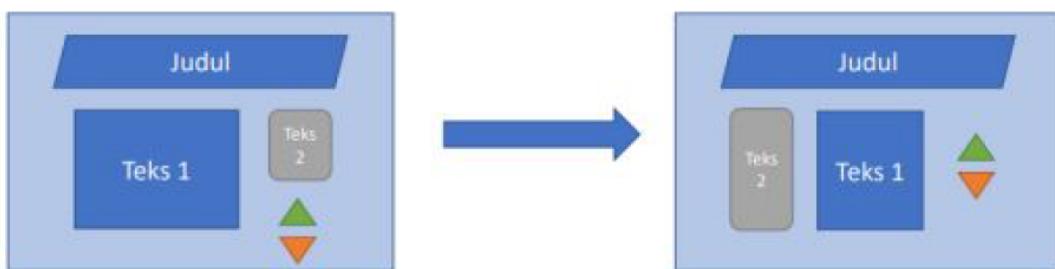
class MyGUI:
    def __init__(self):
        # Membuat widget window utama
        self.main_window = tkinter.Tk()
        # Membuat 2 widget label yang menyimpan
        # pesan 'Hai, apa kabar?'
        self.label1 = tkinter.Label(self.main_window, \
text='Hai, apa kabar?')
        self.label2 = tkinter.Label(self.main_window, \
text='Ini program GUI saya')
        # Panggil metode pack dari kedua label
        self.label1.pack(side='left')
        self.label2.pack(side='left')
        # Masuk ke dalam loop utama tkinter
        tkinter.mainloop()

# Membuat instans untuk kelas MyGUI
my_gui = MyGUI()
```

Metode pack() dengan argumen side='left' dimaksudkan untuk menempatkan posisi label berurutan ke arah kiri/horizontal

## 8.2.4 Menyusun *Widget* dengan *Frame*

*Frame* atau kerangka merupakan sebuah wadah yang digunakan untuk menyimpan *widget*. *Frame* berguna untuk mengorganisir beberapa *widget* dalam sebuah *window* GUI. Misal, kita dapat meletakan sekelompok *widget* dalam sebuah *frame* dan menyusunnya sesuai keinginan. Lalu, meletakkan sekelompok *widget* lain ke dalam *frame* lain dengan susunan yang berbeda.



Contoh program membuat label dalam dua *frame* berbeda:

```
# Label_frame.py
# Program ini membuat label didalam dua frame berbeda

import tkinter

class MyGUI:
    def __init__(self):
        # Membuat widget window utama
        self.window_utama = tkinter.Tk()
```

```

# Membuat dua frame, satu untuk bagian atas window
# yang lainnya untuk bagian bawah window
self.atas_frame = tkinter.Frame(self.window_utama)
self.bawah_frame = tkinter.Frame(self.window_utama)

# Membuat tiga widget label untuk bagian atas frame

self.label1 = tkinter.Label(self.atas_frame, \
text='Atas')
self.label2 = tkinter.Label(self.atas_frame, \
text='Tengah')
self.label3 = tkinter.Label(self.atas_frame, \
text='Bawah')

# Gabungkan label-label yang ada pada bagian
# atas frame. Gunakan perintah side='top' untuk
# menumpuk satu diatas yang lainnya
self.label1.pack(side='top')
self.label2.pack(side='top')
self.label3.pack(side='top')

# Membuat tiga widget label untuk bagian
# bawah frame

self.label4 = tkinter.Label(self.atas_frame, \
text='Kiri')
self.label5 = tkinter.Label(self.atas_frame, \
text='Tengah')
self.label6 = tkinter.Label(self.atas_frame, \
text='Kanan')

# Gabungkan label-label yang ada pada bagian
# bawah frame. Gunakan perintah side='left' untuk
# menyusun secara horizontal, dari bagian
# kiri frame
self.label4.pack(side='left')
self.label5.pack(side='left')
self.label6.pack(side='left')

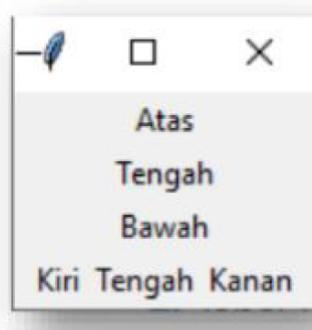
# Susun pula framenya
self.atas_frame.pack()
self.bawah_frame.pack()

# Masuk ke dalam loop utama tkinter
tkinter.mainloop()

# Membuat instansi dari MyGui
my_gui = MyGUI()

```

*Output:*



```

# Membuat dua frame, satu untuk bagian atas window
# yang lainnya untuk bagian bawah window
self.atas_frame = tkinter.Frame(self.window_utama)
self.bawah_frame = tkinter.Frame(self.window_utama)

# Membuat tiga widget label untuk bagian atas frame
self.label1 = tkinter.Label(self.atas_frame, \
text='Atas')
self.label2 = tkinter.Label(self.atas_frame, \
text='Tengah')
self.label3 = tkinter.Label(self.atas_frame, \
text='Bawah')

# Gabungkan label-label yang ada pada bagian
# atas frame. Gunakan perintah side='top' untuk
# menempatkan tiga diatas yang lainnya
self.label1.pack(side='top')
self.label2.pack(side='top')
self.label3.pack(side='top')

# Membuat tiga widget label untuk bagian
# bawah frame
self.label4 = tkinter.Label(self.atas_frame, \
text='Kiri')
self.label5 = tkinter.Label(self.atas_frame, \
text='Tengah')
self.label6 = tkinter.Label(self.atas_frame, \
text='Kanan')

# Gabungkan label-label yang ada pada bagian
# bawah frame. Gunakan perintah side='left' untuk
# menyusun secara horizontal, dari bagian
# kiri frame
self.label4.pack(side='left')
self.label5.pack(side='left')
self.label6.pack(side='left')

# Susun pula framenya
self.atas_frame.pack()
self.bawah_frame.pack()

# Masuk ke dalam loop utama tkinter
tkinter.mainloop()

```

Baris-baris ini membuat dua buah frame. self.window\_utama membuat dua frame, lalu ditambahkan ke widget window\_utama

Baris-baris ini membuat tiga widget label yang ditambahkan ke bagian atas frame

Baris-baris ini memanggil metode pack dan memasukan argumen side='top'

Baris-baris ini membuat tiga widget label yang ditambahkan ke bagian bawah frame

Baris-baris ini memanggil metode pack dan memasukan argumen side='left'. Hal ini membuat tampilan horizontal

Baris-baris ini memanggil metode pack widget frame, yang membuat widget frame dapat ditampilkan

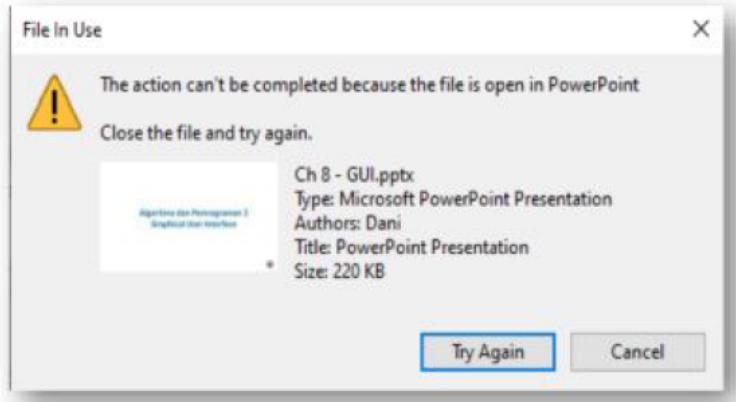
## 8.2.5 Tombol

Dalam membuat tombol pada tkinter, kita perlu menggunakan *widget* tombol yang diletakan dalam sebuah *window* pada program GUI. Tombol *widget* dapat diberi nama, sehingga ketika tombol tersebut di-klik, maka terdapat *method* atau fungsi yang sesuai dengan nama tersebut.



## 8.2.6 Kotak Dialog

Kotak dialog adalah sebuah *window* sederhana yang berisikan informasi untuk pengguna program GUI dan disertai tombol untuk menutup kotak atau melanjutkan aksi.



Contoh program membuat tombol dan kotak dialog pada GUI tkinter:

```
# tombol_dialog.py
# Program ini mendemonstrasikan penggunaan tombol dan menampilkan kotak dialog ketika tombol diklik pada program GUI

import tkinter
import tkinter.messagebox

class MyGUI:
    def __init__(self):
        # Membuat widget window utama
        self.main_window = tkinter.Tk()

        # Membuat widget tombol
        # Teks 'Tekan disini!' akan muncul ditombol
        # Metode 'lakukan_sesuatu' akan dieksekusi ketika
        # tombol diklik oleh pengguna
        self.tombol = tkinter.Button(self.main_window, \
                                      text = 'Tekan disini!', \
                                      command = self.lakukan_sesuatu)

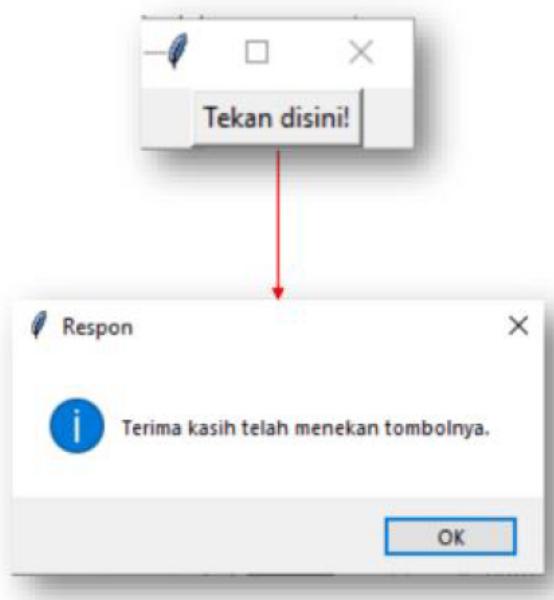
        # Gabungkan tombolnya
        self.tombol.pack()

        # Masuk ke dalam loop utama tkinter
        tkinter.mainloop()

    # Metode lakukan_sesuatu merupakan pemanggilan fungsi
    # untuk widget tombol
    def lakukan_sesuatu(self):
        # Tampilkan informasi kotak dialog
        tkinter.messagebox.showinfo('Respon', \
                                    'Terima kasih telah menekan tombolnya.')

# Buat instansi untuk kelas MyGUI
my_gui = MyGUI()
```

Output:



Contoh program menambahkan tombol keluar pada program sebelumnya:

```
# tombol_keluar.py
# Program ini memiliki tombol keluar yang memanggil
# kelas Tk yang memberhentikan program yang sedang
# berjalan ketika diklik

import tkinter
import tkinter.messagebox
class MyGUI:
    def __init__(self):
        # Membuat widget window utama
        self.main_window = tkinter.Tk()
        # Membuat widget tombol. Teks 'Tekan disini!' akan muncul ditombol
        # Metode 'lakukan_sesuatu' akan dieksekusi ketika tombol diklik oleh
        # pengguna
        self.tombol = tkinter.Button(self.main_window, \
                                      text = 'Tekan disini!', \
                                      command = self.lakukan_sesuatu)

        # Membuat tombol keluar. Ketika tombol ini di klik
        # maka program akan berhenti berjalan
        self.tombol_keluar = tkinter.Button(self.main_window, \
                                             text = 'Keluar', \
                                             command = self.main_window.destroy)

        # Gabungkan tombolnya
        self.tombol.pack()
        self.tombol_keluar.pack()
        # Masuk ke dalam loop utama tkinter
        tkinter.mainloop()

    # Metode lakukan_sesuatu merupakan pemanggilan fungsi
    # untuk widget tombol
    def lakukan_sesuatu(self):
        # Tampilkan informasi kotak dialog
        tkinter.messagebox.showinfo('Respon', \
```

```
'Terima kasih telah menekan tombolnya.')
# Buat instansi untuk kelas MyGUI
my_gui = MyGUI()
```

Output:



### 8.2.7 Input dengan Widget Entry

Widget entry adalah sebuah area berbentuk kotak yang dapat diketikkan sesuatu dan biasanya disertai oleh tombol untuk mengirim *input*-nya ke program. Widget entry memiliki *method* yang digunakan untuk menerima *input* dari yang sudah diketik oleh pengguna pada area kotak tersebut.



Contoh program untuk mengolah *input* dengan *widget entry*:

```
# konversi_kilo.py
# Program ini mengkonversi jarak kilometer ke dalam
# mil. Hasilnya akan ditampilkan ke sebuah dialog box

import tkinter
import tkinter.messagebox

class KonversikiloGUI:
    def __init__(self):
        # Membuat widget window utama
        self.main_window = tkinter.Tk()

        # Membuat dua frame untuk membentuk grup widget
        self.frame_atas = tkinter.Frame(self.main_window)
        self.frame_bawah = tkinter.Frame(self.main_window)

        # Membuat widget untuk frame atas
        self.label = tkinter.Label(self.frame_atas,
                                   text='Masukkan jarak dalam KM : ')
        self.entry_km = tkinter.Entry(self.frame_atas,
                                     width=10)
```

```

# Menggabungkan widget untuk frame atas
self.label.pack(side='left')
self.entry_km.pack(side='left')

# Membuat widget tombol untuk frame bawah
self.tombol_hitung = tkinter.Button(self.frame_bawah, \
                                      text='Konversi', \
                                      command=self.konversi)
self.tombol_keluar = tkinter.Button(self.frame_bawah, \
                                      text='Keluar', \
                                      command=self.main_window.destroy)

# Gabungkan tombol-tombol
self.tombol_hitung.pack(side='top')
self.tombol_keluar.pack(side='top')

# Gabungkan frame
self.frame_atas.pack()
self.frame_bawah.pack()

# Masuk ke dalam loop utama tkinter
tkinter.mainloop()

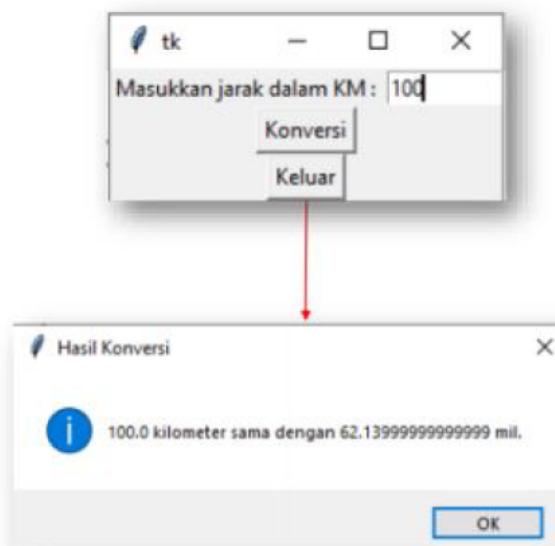
# Method konversi adalah fungsi yang dipanggil ketika tombol_hitung diklik
def konversi(self):
    # Ambil nilai yang diinput pengguna ke dalam widget entry_km
    kilo = float(self.entry_km.get())
    # Konversi kilo ke mil
    mil = kilo * 0.6214

    # Tampilkan hasil ke dalam kotak dialog
    tkinter.messagebox.showinfo('Hasil Konversi', \
                                str(kilo) + ' kilometer sama dengan ' + \
                                str(mil) + ' mil.')

# Membuat instans untuk kelas KonversiKiloGUI
kilo_mil = KonversiKiloGUI()

```

*Output:*



## 8.2.8 Output dengan Widget Label

Ketika sebuah objek `StringVar` (variabel yang menyimpan nilai data `string`) dihubungkan dengan *widget* label, label tersebut akan menampilkan data yang disimpan di objek `StringVar` tersebut. Hal ini dapat membantu menampilkan hasil dari pengolahan program tanpa menggunakan kotak dialog lain, sehingga memungkinkan untuk membuat sebuah tampilan yang lebih dinamis.



Contoh program untuk menampilkan *output* dengan *widget* label:

```
# konversi_kilo2.py
# Program ini mengkonversi jarak kilometer ke dalam
# mil. Hasilnya akan ditampilkan ke sebuah dialog box

import tkinter
import tkinter.messagebox

class KonversikiloGUI:
    def __init__(self):
        # Membuat widget window utama
        self.main_window = tkinter.Tk()

        # Membuat dua frame untuk membentuk grup widget
        self.frame_atas = tkinter.Frame(self.main_window)
        self.frame_tengah = tkinter.Frame(self.main_window)
        self.frame_bawah = tkinter.Frame(self.main_window)

        # Membuat widget untuk frame atas
        self.label = tkinter.Label(self.frame_atas, \
            text='Masukkan jarak dalam KM : ')
        self.entry_km = tkinter.Entry(self.frame_atas, \
            width=10)

        # Menggabungkan widget untuk frame atas
        self.label.pack(side='left')
        self.entry_km.pack(side='left')

        # Buatlah widget untuk frame tengah
        self.desk_label = tkinter.Label(self.frame_tengah, \
            text='Terkonversi ke dalam mil : ')
        # Dibutuhkan StringVar object untuk mengasosiasikannya
        # dengan sebuah label output. Gunakan method set()
        # untuk menyimpan sting dari karakter kosong
        self.nilai = tkinter.StringVar()

        # Buatlah sebuah label dan hubungkan dengan objek
        # StringVar. Nilai apapun yang disimpan dalam objek
```

```

# StringVar akan secara otomatis ditampilkan ke
# dalam label
self.label_mil = tkinter.Label(self.frame_tengah, \
textvariable=self.nilai)

# Gabungkan widget pada frame tengah
self.desk_label.pack(side='left')
self.label_mil.pack(side='left')

# Membuat widget tombol untuk frame bawah
self.tombol_hitung = tkinter.Button(self.frame_bawah, \
text='Konversi', \
command=self.konversi)
self.tombol_keluar = tkinter.Button(self.frame_bawah, \
text='Keluar', \
command=self.main_window.destroy)

# Gabungkan tombol-tombol
self.tombol_hitung.pack(side='top')
self.tombol_keluar.pack(side='top')

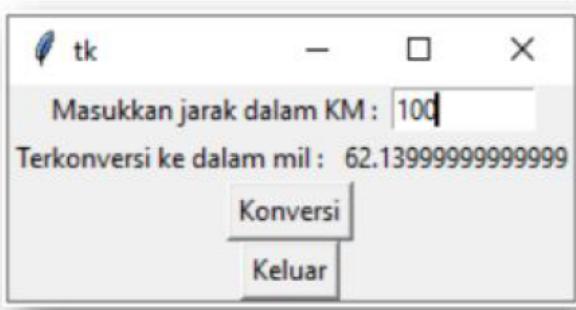
# Gabungkan frame
self.frame_atas.pack()
self.frame_tengah.pack()
self.frame_bawah.pack()
# Masuk ke dalam loop utama tkinter
tkinter.mainloop()

# Method konversi adalah fungsi yang dipanggil
# ketika tombol_hitung diklik
def konversi(self):
    # Ambil nilai yang diinput pengguna ke dalam widget entry_km
    kilo = float(self.entry_km.get())
    # Konversi kilo ke mil
    mil = kilo * 0.6214
    # Konversi mil ke sebuah string dan simpan ke
    # dalam objek StringVar. Ini akan secara
    # otomatis memperbarui widget label_mil
    self.nilai.set(mil)

# Membuat instans untuk kelas KonversikiloGUI
kilo_mil = KonversikiloGUI()

```

Output:



## **REFERENSI**

- [1] Gaddis, Tony. 2012. Starting Out With Python Second Edition. United States of America: Addison-Wesley.