Pemrograman Desktop

GUI, QT, dan PYQT6

yoga@trunojoyo.ac.id

Pendahuluan

- GUI (Graphical User Interfaces / Antarmuka Pengguna Grafis) adalah Sistem yang memiliki banyak fitur seperti pada GUI desktop modern, termasuk jendela, menu, tombol radio, kotak centang, dan ikon.
- Kombinasi fitur tersebut diistilahkan : WIMP (windows, icon, menu, pointing device - mouse).

Pendahuluan

- Qt adalah kerangka pengembangan aplikasi lintas platform untuk perangkat desktop, tertanam, dan seluler.
- Platform yang Didukung meliputi Linux, OS X, Windows, VxWorks, QNX, Android, iOS, BlackBerry, Sailfish OS, dan lainnya.
- Qt bukanlah bahasa pemrograman tersendiri. Ini adalah kerangka kerja yang ditulis dalam C++.
- Praprosesor, MOC (Meta-Object Compiler), digunakan untuk memperluas bahasa C++ dengan fitur seperti sinyal dan slot.

Pendahuluan

- PyQt6 adalah pengikatan Python dari toolkit Qt, yang dikembangkan oleh Riverbank Computing.
- Saat menulis aplikasi menggunakan PyQt6, sama dengan menulis aplikasi di Qt.
- Pustaka PyQt6 adalah pembungkus pustaka C++ Qt, yang memungkinkan untuk menggunakannya dengan Python.

Fitur Dasar PyQt6 - Aplikasi Pertama

- Langkah pertama adalah mulai membuat file Python baru. Dalam memberi nama bisa disesuaikan dengan kebutuhan (misalnya myapp.py) dan menyimpannya di tempat yang mudah diakses.

```
from PyQt6.QtWidgets import QApplication, QWidget
import sys

app = QApplication(sys.argv)

window = QWidget()

window.show()

app.exec()
```

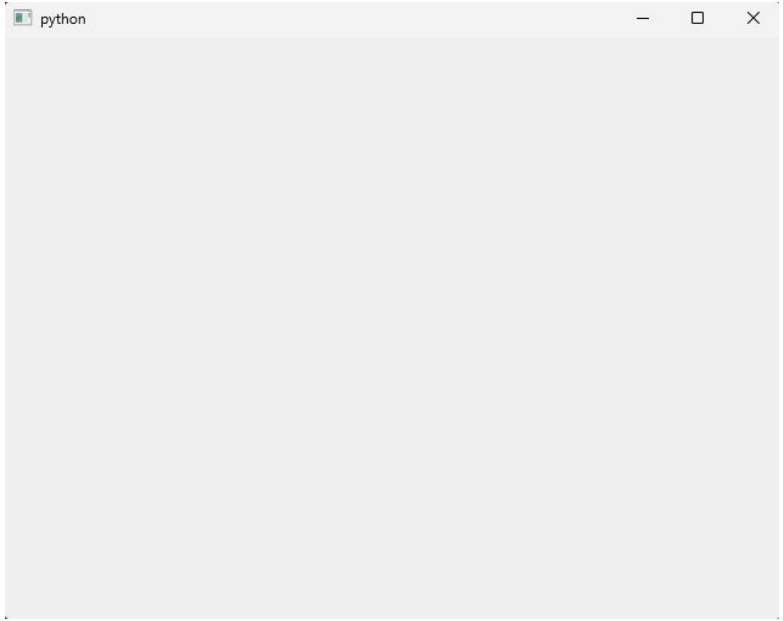
Aplikasi Pertama

- Pertama, menjalankan aplikasi. Kita bisa menjalankan menggunakan baris perintah seperti skrip Python lainnya, misalnya

```
python MyApp.py
```

Atau, untuk Python 3

```
python3 MyApp.py
```



Aplikasi Pertama

- Apa yang terjadi :
- Pertama, kita mengimpor kelas PyQt6 yang kita perlukan untuk aplikasi. Di sini kita mengimpor QApplication, pengendali aplikasi dan QWidget, widget GUI dasar yang kosong, keduanya dari modul QtWidgets.
- Modul utama untuk Qt adalah QtWidgets, QtGui dan QtCore.

```
from PyQt6.QtWidgets import QApplication, QWidget
```

Aplikasi Pertama

- Selanjutnya membuat instance QApplication, meneruskan sys.arg, yaitu Python list berisi argumen baris perintah yang diteruskan ke aplikasi.

```
app = QApplication(sys.argv)
```

- Selanjutnya kita membuat instance QWidget menggunakan nama variabel window.

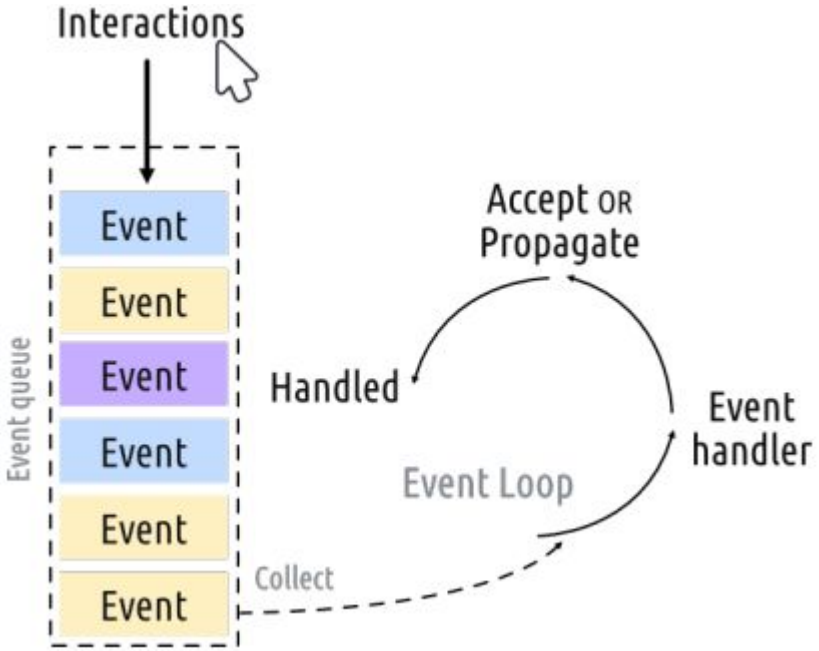
```
window = QWidget()

window.show()
```

Aplikasi Pertama

- Di Qt, semua widget tingkat atas adalah jendela, artinya, widget tersebut tidak memiliki induk dan tidak disarangkan dalam widget atau tata letak lain.
- Jadi, Apa itu jendela ?
  - Menyimpan antarmuka pengguna aplikasi Anda
  - Setiap aplikasi membutuhkan setidaknya satu (…tetapi bisa lebih)
  - Aplikasi akan (secara default) keluar ketika jendela terakhir ditutup
- Terakhir, kita memanggil app.exec() untuk memulai event loop

Event loop

- Inti dari setiap Aplikasi Qt adalah kelas QApplication. Setiap aplikasi memerlukan satu - dan hanya satu - QApplication object agar dapat berfungsi. Objek ini menyimpan event loop aplikasi - loop inti yang mengatur semua interaksi pengguna dengan GUI.



Event Loop

- Setiap interaksi dengan aplikasi, baik itu penekanan tombol, klik mouse, atau gerakan mouse menghasilkan event (peristiwa) yang ditempatkan pada antrean event.
- Dalam perulangan event, antrian diperiksa pada setiap iterasi dan jika event yang menunggu ditemukan, event dan kontrol diteruskan ke pengendali event tertentu untuk event tersebut.
- Pengendali event menangani event tersebut, dan kemudian meneruskan kendali kembali ke perulangan event untuk menunggu event lainnya.
- Hanya ada satu event loop yang berjalan per aplikasi.

QApplication

- Kelas QApplication
  - QApplication menampung event loop Qt
  - Diperlukan satu instans Aplikasi Qt
  - Aplikasi menunggu di event loop hingga tindakan diambil
  - Hanya ada satu event loop

QMainWindow

- Pada Qt widget apa pun bisa berupa windows.
- Misalnya, jika Anda mengganti QtWidget dengan QPushButton.
- Pada contoh berikut, Kita akan mendapatkan jendela dengan satu tombol yang dapat ditekan di dalamnya.

```
import sys
from PyQt6.QtWidgets import
QApplication, QPushButton

app = QApplication(sys.argv)

window = QPushButton("Tekan Saya")

window.show()

app.exec()
```

QMainWindow

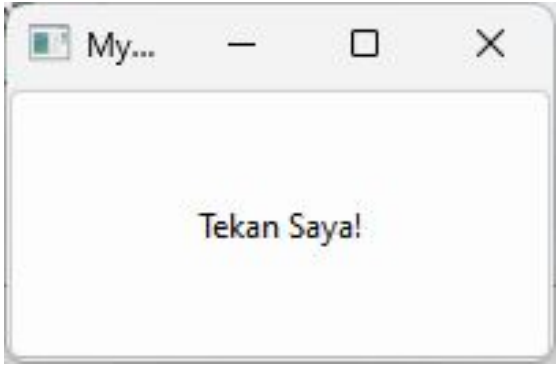
- QMainWindow. adalah widget siap pakai yang menyediakan banyak fitur jendela standar yang akan kita gunakan di aplikasi kita, termasuk toolbar, menu, statusbar, widget yang dapat dipasang ke dok, dan banyak lagi.

QMainWindow

```
import sys
from PyQt6.QtCore import QSize, Qt
from PyQt6.QtWidgets import(
     QApplication,
     QMainWindow,
     QPushButton,
)①

# Subclass QMainWindow to customize your
application's main window
class MainWindow(QMainWindow):
   def __init__(self):
       super().__init__()②
       self.setWindowTitle("My App")
       button = QPushButton("Tekan
                                  Saya!")

       self.setCentralWidget(button)③

app = QApplication(sys.argv)
window = MainWindow()
window.show()
app.exec()
```

QMainWindow

- Penjelasan:
- ① Widget Qt umum selalu diimpor dari namespace QtWidgets.
- ② Kita harus selalu memanggil metode __init__ dari kelas super().
- ③ Gunakan .setCentralWidget untuk menempatkan widget di QMainWindow.
- Saat membuat subkelas kelas Qt, kita harus selalu memanggil fungsi super__init__ agar Qt dapat menyiapkan objek.



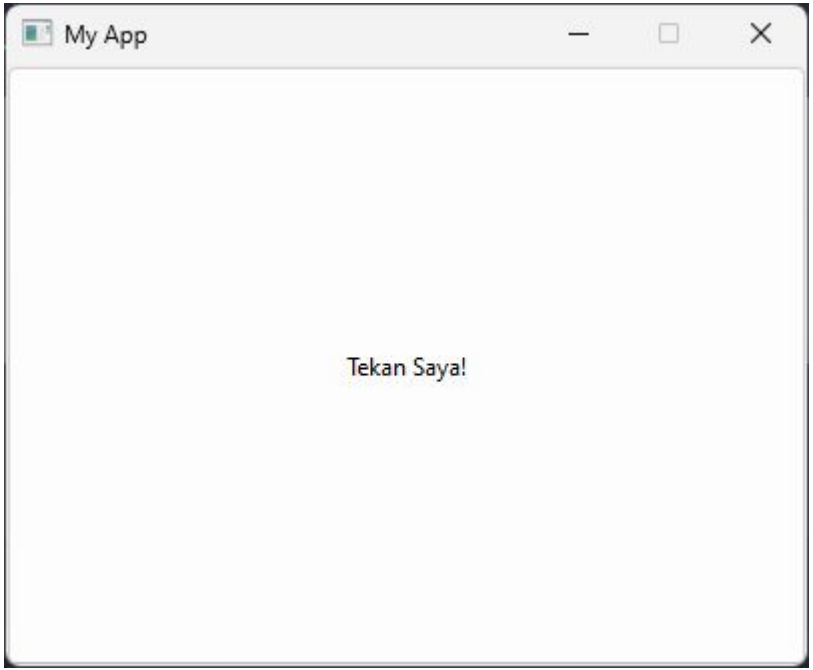
QMainWindow

- Dalam Qt, ukuran ditentukan menggunakan objek QSize.
- Object tersebut menerima parameter lebar dan tinggi.
- Misalnya, berikut ini akan membuat jendela berukuran tetap 400x300 piksel.

QMainWindow

```
import sys
from PyQt6.QtCore import QSize, Qt
from PyQt6.QtWidgets import
QApplication, QMainWindow,
QPushButton

# Subclass QMainWindow to customize
your application's main window
class MainWindow(QMainWindow):
    def __init__(self):
        super().__init__()
        self.setWindowTitle("My
                                App")
        button = QPushButton("Tekan
                             Saya!")
        self.setFixedSize(QSize(400,
                          300))  ①

        self.setCentralWidget(button)

app = QApplication(sys.argv)
window = MainWindow()
window.show()
app.exec()
```

QMainWindow



Signals & Slots

- Umumnya kita ingin aplikasi tersebut melakukan sesuatu.
- Yang kita perlukan adalah cara menghubungkan tindakan menekan tombol dengan membuat sesuatu terjadi.
- Di Qt, ini disediakan oleh sinyal dan slot.

Signals & Slots

- Sinyal adalah notifikasi yang dikeluarkan oleh widget ketika terjadi sesuatu. Sesuatu itu bisa berupa apa saja, mulai dari menekan tombol, mengubah teks kotak input, hingga mengubah teks jendela.
- Banyak sinyal yang dipicu oleh tindakan pengguna, namun hal ini bukanlah suatu aturan.
- Selain memberi tahu tentang sesuatu yang terjadi, sinyal juga dapat mengirimkan data untuk memberikan konteks tambahan tentang apa yang terjadi.

Signals & Slots

- Slot adalah nama yang digunakan Qt untuk penerima sinyal.
- Dalam Python, fungsi (atau metode) apa pun dalam aplikasi kita dapat digunakan sebagai slot hanya dengan menghubungkan sinyal ke slot tersebut.
- Jika sinyal mengirimkan data, maka fungsi penerima akan menerima data tersebut.
- Banyak widget Qt juga memiliki slot bawaannya sendiri, artinya kita dapat menghubungkan widget Qt secara langsung.

Sinyal QPushButton

```
from PyQt6.QtWidgets import(
     QApplication,
     QMainWindow,
     QPushButton,
)①

import sys

class MainWindow(QMainWindow):
   def __init__(self):
       super().__init__()②
       self.setWindowTitle("My App")
       button = QPushButton("Tekan saya!")
```

Sinyal QPushButton

```
        button.setCheckable(True)
        button.clicked.connect(self.tombol_telah_di_click)
        # Set the central widget of the Window.
        self.setCentralWidget(button)

    def tombol_telah_di_click(self):
        print("Telah di Click!")

app = QApplication(sys.argv)
window = MainWindow()
window.show()
app.exec()
```

Menerima data

- sinyal juga dapat mengirimkan data untuk memberikan lebih banyak informasi tentang apa yang baru saja terjadi.
- Sinyal yang diklik tidak terkecuali, juga menyediakan status dicentang (atau dialihkan) untuk tombol tersebut.

Contoh

```
from PyQt6.QtWidgets import(
     QApplication,
     QMainWindow,
     QPushButton,
)①

import sys
class MainWindow(QMainWindow):
   def __init__(self):
       super().__init__()②
       self.setWindowTitle("My App")
       button = QPushButton("Tekan saya!")
       button.setCheckable(True)
       button.clicked.connect(self.tombol_telah_di_click)
```

Contoh

```
        button.clicked.connect(self.tombol_telah_dialihkan)
        # Set the central widget of the Window.
        self.setCentralWidget(button)

    def tombol_telah_di_click(self):
        print("Telah di Click!")

    def tombol_telah_dialihkan(self, dicentang):
        print("dicentang ?", dicentang)

app = QApplication(sys.argv)
window = MainWindow()
window.show()
app.exec()
```

Menyimpan data

- Seringkali berguna untuk menyimpan status widget saat ini dalam variabel Python.
- Hal ini memungkinkan anda untuk bekerja dengan nilai-nilai seperti variabel Python lainnya dan tanpa mengakses widget asli.
- Kita dapat menyimpan nilai-nilai ini sebagai variabel individual atau menggunakan dictionary jika kita mau.

Contoh

```
class MainWindow(QMainWindow):
   def __init__(self):
       super().__init__()
       self.tombol_dicentang =True  ①

       self.setWindowTitle("My App")
       button = QPushButton("Tekan Saya!")
       button.setCheckable(True)
       button.clicked.connect(self.tombol_telah_dirubah)
       button.setChecked(self.tombol_dicentang) ②
       self.setCentralWidget(button)

   def tombol_telah_dirubah(self, dicentang):
       self.tombol_dicentang = dicentang  ③
       print(self.button_is_checked)
```