

BAB 8

PyQt



8.1 Tujuan

1. Dapat memahami konsep dan penggunaan PyQt
2. Dapat mengimplementasikan PyQt untuk membuat tampilan GUI sederhana

8.2 Pengantar

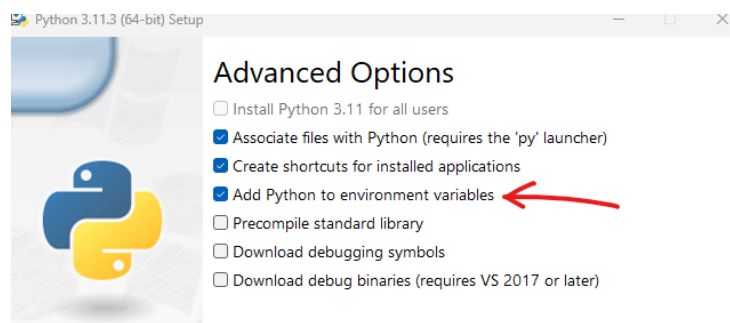
8.2.1 Pendahuluan PyQt

PyQt adalah set binding Python untuk pustaka Qt yang digunakan untuk membuat aplikasi GUI (Graphical User Interface). PyQt dikembangkan oleh Riverbank Computing Limited. Ada dua versi PyQt utama yang tersedia yaitu PyQt4 dan PyQt5. Versi ini merujuk pada versi Qt library yang digunakan. PyQt5 mendukung Qt5, dan PyQt4 mendukung Qt4

8.2.2 Instalasi dan Setup PyQt untuk Windows

Step 1. Instalasi Python

Instal Python dari situs web resmi Python di <https://www.python.org/downloads/>. Pastikan untuk mencentang opsi "Add Python to PATH" saat melakukan instalasi.



Step 2. Instalasi PyQt

Masuk ke Command Prompt kemudian ketikkan perintah berikut untuk melakukan instalasi

```
1. pip install pyqt5
```

Step 3. Verifikasi Instalasi

Setelah instalasi selesai ketikkan perintah berikut pada python interpreter

```
1. import PyQt5
```

Jika tidak ada error maka proses instalasi PyQt5 sudah selesai

```
C:\Users\AIO ASUS>python
Python 3.11.3 (tags/v3.11.3:f3909b8, Apr  4 2023, 23:49:59) [MSC v.1934 64
Type "help", "copyright", "credits" or "license" for more information.
>>> import PyQt5
>>> |
```

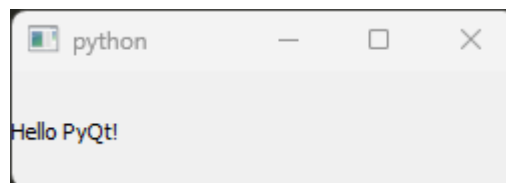
8.2.3 Konsep Dasar PyQt

Dalam PyQt, aplikasi GUI dibuat dengan membuat sebuah instance dari QApplication. Setelah itu, widget (seperti jendela, tombol, atau label) dibuat dan dimodifikasi sesuai kebutuhan. Setelah semua widget disiapkan, aplikasi memasuki loop event dengan memanggil metode `exec_()` dari QApplication. Loop event ini yang bertanggung jawab untuk menangani semua event seperti klik mouse atau ketukan keyboard.

Berikut adalah contoh sederhana dari aplikasi PyQt:

```
1. from PyQt5.QtWidgets import QApplication, QLabel
2.
3. app = QApplication([])
4. label = QLabel('Hello PyQt!')
5. label.show()
6. app.exec_()
```

Dalam contoh ini, sebuah aplikasi dan label dibuat. Label tersebut kemudian ditampilkan, dan aplikasi memasuki loop event. Ketika kode dijalankan maka akan menampilkan output seperti berikut



8.2.4 Struktur Aplikasi PyQt

Aplikasi PyQt biasanya memiliki struktur berikut:

- Import modul yang diperlukan: Melakukan import terhadap modul-modul yang diperlukan termasuk modul PyQt5.QtWidgets yang berisi kelas dasar yang diperlukan untuk membuat aplikasi GUI.

- Membuat aplikasi: Aplikasi dibuat dengan membuat instance dari QApplication.
- Membuat widget: Widget adalah elemen-elemen GUI seperti jendela, tombol, label, dan lainnya. Kita dapat membuat widget dengan membuat instance dari kelas-kelas seperti QWidget, QPushButton, QLabel, dan lainnya.
- Menyiapkan widget: Setelah widget dibuat, kita bisa mengubah propertinya seperti ukuran, posisi, dan teks. Kita juga bisa mengatur tata letak widget dengan menggunakan kelas-kelas seperti QVBoxLayout, QHBoxLayout, dan GridLayout.
- Menampilkan widget: Widget ditampilkan dengan memanggil metode show().
- Memasuki loop event: Aplikasi memasuki loop event dengan memanggil metode exec_() dari QApplication. Loop event ini yang bertanggung jawab untuk menangani semua event seperti klik mouse atau ketukan keyboard.

Berikut adalah contoh struktur kode PyQt5 menggunakan konsep pemrograman berorientasi objek (OOP):

```

1. # Import necessary modules from PyQt5
2. from PyQt5.QtWidgets import QApplication, QMainWindow, QLabel
3.
4. # Define a MainWindow class that inherits from QMainWindow
5. class MainWindow(QMainWindow):
6.     def __init__(self):
7.         # Call the constructor of the parent class
8.         super().__init__()
9.
10.        # Set some basic attributes for the window
11.        self.title = "Hello PyQt5"
12.        self.top = 100
13.        self.left = 100
14.        self.width = 400
15.        self.height = 300
16.
17.        # Call the method that initializes the window

```

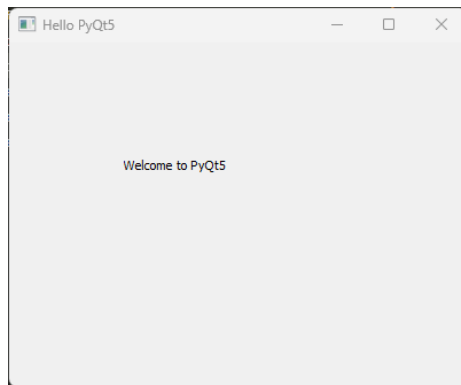
```

18.         self.initWindow()
19.
20.     def initWindow(self):
21.         # Create a QLabel widget and set it as the central widget o
f the window
22.         self.label = QLabel("Welcome to PyQt5", self)
23.         self.label.adjustSize() # Adjust the size of the label to
fit the text
24.         self.label.move(100, 100) # Move the label to the position
(100, 100)
25.
26.         # Set the title of the window
27.         self.setWindowTitle(self.title)
28.         # Set the position and size of the window
29.         self.setGeometry(self.top, self.left, self.width, self.heig
ht)
30.         # Show the window
31.         self.show()
32.
33. # Create an instance of QApplication
34. app = QApplication([])
35. # Create an instance of MainWindow
36. window = MainWindow()
37. # Start the event loop
38. app.exec_()

```

Dalam contoh ini, kita membuat kelas MainWindow yang mewarisi dari QMainWindow. Kelas ini memiliki metode `__init__()` di mana kita mengatur beberapa atribut dasar seperti judul, posisi, dan ukuran jendela. Kita juga memanggil metode `initWindow()` di mana kita mengatur lebih banyak properti jendela dan menampilkan jendela.

Kemudian, kita membuat instance dari QApplication dan MainWindow, dan memulai loop event dengan memanggil `app.exec_()`.



8.2.5 Widget dan Layout

8.2.5.1 QLabel

QLabel adalah widget yang digunakan untuk menampilkan teks atau gambar

```
1. from PyQt5.QtWidgets import QApplication, QMainWindow, QLabel
2.
3. class MainWindow(QMainWindow):
4.     def __init__(self):
5.         super().__init__()
6.
7.         self.label = QLabel('Hello PyQt!', self)
8.         self.label.adjustSize()
9.
10. app = QApplication([])
11. window = MainWindow()
12. window.show()
13. app.exec_()
```

8.2.5.2 QPushButton

QPushButton adalah tombol yang bisa diklik oleh pengguna

```
1. from PyQt5.QtWidgets import QApplication, QMainWindow, QPushButton
2.
3. class MainWindow(QMainWindow):
4.     def __init__(self):
5.         super().__init__()
6.
7.         self.button = QPushButton('Click Me!', self)
8.         self.button.clicked.connect(self.on_button_clicked)
```

```

9.
10.     def on_button_clicked(self):
11.         print('Button clicked!')
12.
13. app = QApplication([])
14. window = MainWindow()
15. window.show()
16. app.exec_()

```

8.2.5.3 QLineEdit

QLineEdit adalah kotak teks satu baris yang bisa digunakan pengguna untuk memasukkan teks..

```

1. from PyQt5.QtWidgets import QApplication, QMainWindow, QLineEdit
2.
3. class MainWindow(QMainWindow):
4.     def __init__(self):
5.         super().__init__()
6.
7.         self.lineEdit = QLineEdit(self)
8.         self.lineEdit.textChanged.connect(self.on_text_changed)
9.
10.    def on_text_changed(self, text):
11.        print('Text changed:', text)
12.
13. app = QApplication([])
14. window = MainWindow()
15. window.show()
16. app.exec_()

```

8.2.5.4 QTextEdit

QTextEdit adalah kotak teks multi-baris yang bisa digunakan pengguna untuk memasukkan teks.

```

1. from PyQt5.QtWidgets import QApplication, QMainWindow, QTextEdit
2.
3. class MainWindow(QMainWindow):
4.     def __init__(self):

```

```

5.         super().__init__()
6.
7.         self.textEdit = QTextEdit(self)
8.         self.textEdit.textChanged.connect(self.on_text_changed)
9.
10.    def on_text_changed(self):
11.        print('Text changed')
12.
13.app = QApplication([])
14.window = MainWindow()
15.window.show()
16.app.exec_()
17.

```

8.2.5.5 QCheckBox

QCheckBox adalah kotak centang yang bisa dicentang atau tidak dicentang oleh pengguna

```

1. from PyQt5.QtWidgets import QApplication, QMainWindow, QCheckBox
2. from PyQt5.QtCore import Qt
3.
4. class MainWindow(QMainWindow):
5.     def __init__(self):
6.         super().__init__()
7.
8.         self.checkBox = QCheckBox('Check Me', self)
9.         self.checkBox.stateChanged.connect(self.on_state_changed)
10.
11.    def on_state_changed(self, state):
12.        if state == Qt.Checked:
13.            print('Checked')
14.        else:
15.            print('Unchecked')
16.
17.app = QApplication([])
18.window = MainWindow()
19.window.show()
20.app.exec_()
21.

```


8.2.5.6 QRadioButton

QRadioButton adalah tombol radio yang biasanya digunakan dalam grup di mana pengguna bisa memilih satu dari beberapa pilihan

```
1. from PyQt5.QtWidgets import QApplication, QMainWindow, QRadioButto
   n
2.
3. class MainWindow(QMainWindow):
4.     def __init__(self):
5.         super().__init__()
6.
7.         self.radioButton = QRadioButton('Select Me', self)
8.         self.radioButton.toggled.connect(self.on_toggled)
9.
10.    def on_toggled(self, is_checked):
11.        if is_checked:
12.            print('Selected')
13.
14. app = QApplication([])
15. window = MainWindow()
16. window.show()
17. app.exec_()
18.
```

8.2.5.7 QComboBox

QComboBox adalah kotak kombinasi yang memungkinkan pengguna memilih satu dari beberapa pilihan

```
1. from PyQt5.QtWidgets import QApplication, QMainWindow, QComboBox
2.
3. class MainWindow(QMainWindow):
4.     def __init__(self):
5.         super().__init__()
6.
7.         self.comboBox = QComboBox(self)
8.         self.comboBox.addItem('Option 1')
9.         self.comboBox.addItem('Option 2')
10.        self.comboBox.addItem('Option 3')
11.        self.comboBox.currentIndexChanged.connect(self.on_index_cha
12.            nged)
13.
14.
15.
16.
```

```

11.     def on_index_changed(self, index):
12.         print('Selected option:', self.comboBox.itemText(index))
13.
14. app = QApplication([])
15. window = MainWindow()
16. window.show()
17. app.exec_()
18.

```

8.2.5.8 Qslider

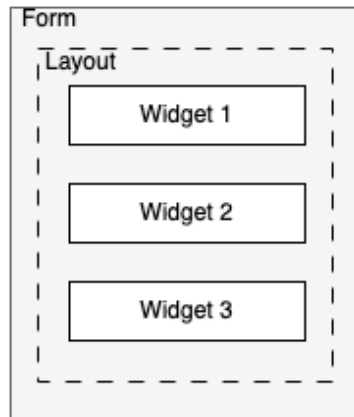
QSlider adalah slider yang memungkinkan pengguna memilih nilai dari rentang nilai dengan menggeser penanda.

```

1. from PyQt5.QtWidgets import QApplication, QMainWindow, QSlider
2. from PyQt5.QtCore import Qt
3.
4. class MainWindow(QMainWindow):
5.     def __init__(self):
6.         super().__init__()
7.
8.         self.slider = QSlider(Qt.Horizontal, self)
9.         self.slider.setMinimum(0)
10.        self.slider.setMaximum(100)
11.        self.slider.valueChanged.connect(self.on_value_changed)
12.
13.        def on_value_changed(self, value):
14.            print('Slider value:', value)
15.
16. app = QApplication([])
17. window = MainWindow()
18. window.show()
19. app.exec_()
20.

```

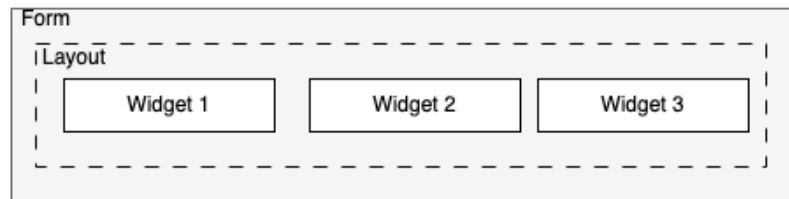
8.2.5.9 QVBoxLayout



`QVBoxLayout` adalah layout yang mengatur widget secara vertikal. Berikut adalah contoh penggunaannya:

```
1. from PyQt5.QtWidgets import QApplication, QWidget, QVBoxLayout, QPushButton
2.
3. class MainWindow(QWidget):
4.     def __init__(self):
5.         super().__init__()
6.
7.         layout = QVBoxLayout()
8.
9.         layout.addWidget(QPushButton('Button 1'))
10.        layout.addWidget(QPushButton('Button 2'))
11.        layout.addWidget(QPushButton('Button 3'))
12.
13.        self.setLayout(layout)
14.
15. app = QApplication([])
16. window = MainWindow()
17. window.show()
18. app.exec_()
```

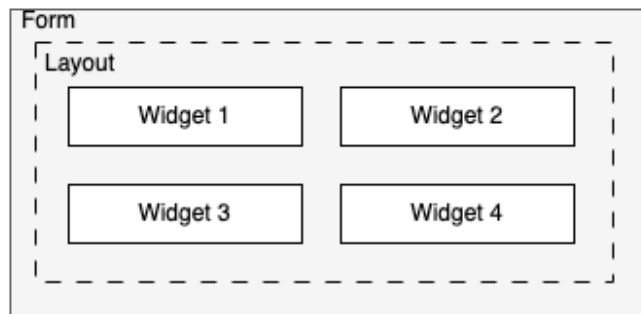
8.2.5.10 QHBoxLayout



QHBoxLayout adalah layout yang mengatur widget secara horizontal. Berikut adalah contoh penggunaannya:

```
1. from PyQt5.QtWidgets import QApplication, QWidget, QHBoxLayout, QPushButton
2.
3. class MainWindow(QWidget):
4.     def __init__(self):
5.         super().__init__()
6.
7.         layout = QHBoxLayout()
8.
9.         layout.addWidget(QPushButton('Button 1'))
10.        layout.addWidget(QPushButton('Button 2'))
11.        layout.addWidget(QPushButton('Button 3'))
12.
13.        self.setLayout(layout)
14.
15.app = QApplication([])
16.window = MainWindow()
17.window.show()
18.app.exec_()
```

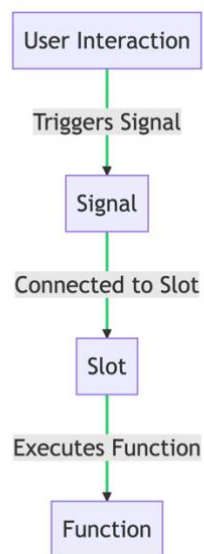
8.2.5.11 QGridLayout



QGridLayout adalah layout yang mengatur widget dalam bentuk grid. Berikut adalah contoh penggunaannya:

```
1. from PyQt5.QtWidgets import QApplication, QWidget, QGridLayout, QPushButton
2.
3. class MainWindow(QWidget):
4.     def __init__(self):
5.         super().__init__()
6.
7.         layout = QGridLayout()
8.
9.         layout.addWidget(QPushButton('Button 1'), 0, 0)
10.        layout.addWidget(QPushButton('Button 2'), 0, 1)
11.        layout.addWidget(QPushButton('Button 3'), 1, 0)
12.        layout.addWidget(QPushButton('Button 4'), 1, 1)
13.        layout.addWidget(QPushButton('Button 5'), 2, 0, 1, 2)
14.        self.setLayout(layout)
15.
16. app = QApplication([])
17. window = MainWindow()
18. window.show()
19. app.exec_()
```

8.2.6 Signal dan Slot



Signal dan Slot adalah fitur fundamental dari PyQt5 dan merupakan mekanisme yang memungkinkan komunikasi antara objek dalam aplikasi GUI.

Signal

Signal dalam PyQt5 adalah sebuah event yang bisa dipancarkan oleh objek tertentu. Event ini bisa berupa aksi pengguna seperti klik tombol, mengubah nilai slider, memilih item dari dropdown, dan lainnya. Signal juga bisa dipancarkan oleh program itu sendiri, misalnya ketika timer berakhir atau sebuah file selesai didownload.

Slot

Slot adalah fungsi atau metode yang dipanggil sebagai respons terhadap sebuah signal. Slot bisa berupa fungsi apa saja, termasuk fungsi yang dibuat oleh pengguna. Slot bisa memiliki parameter, dan parameter ini bisa dipasok oleh signal.

Mekanisme Penggunaan

Untuk menggunakan signal dan slot, kita perlu melakukan dua hal:

1. Membuat slot, yaitu fungsi yang akan dipanggil ketika signal dipancarkan.
2. Menghubungkan signal ke slot menggunakan metode `connect()`.

Berikut adalah contoh penggunaan signal dan slot dalam PyQt5:

```
1. from PyQt5.QtWidgets import QApplication, QMainWindow, QPushButton
2.
3. class MainWindow(QMainWindow):
4.     def __init__(self):
5.         super().__init__()
6.
7.         self.button = QPushButton('Click Me!', self)
8.         self.button.clicked.connect(self.on_button_clicked)
9.
10.    def on_button_clicked(self):
11.        print('Button clicked!')
12.
```

```

13.app = QApplication([])
14.window = MainWindow()
15.window.show()
16.app.exec_()

```

Dalam contoh ini, kita membuat sebuah QPushButton dan menghubungkan signal `clicked` dari tombol tersebut ke slot `on_button_clicked`. Ketika tombol diklik, signal `clicked` akan dipancarkan dan slot `on_button_clicked` akan dipanggil, mencetak 'Button clicked!' ke console.

8.3 Kegiatan Praktikum

8.3.1 Kegiatan 1 : Layout

1. Buat sebuah file program baru kemudian tulis kode program berikut ini

```

1. from PyQt5.QtWidgets import QApplication, QMainWindow, QLabel, QPush
   hButton, QLineEdit, QVBoxLayout, QHBoxLayout, QGridLayout, QWidget
2.
3. class MainWindow(QMainWindow):
4.     def __init__(self):
5.         super().__init__()
6.
7.         self.setWindowTitle('Layouts Example')
8.
9.         # Create some widgets
10.        label1 = QLabel('Label 1')
11.        label2 = QLabel('Label 2')
12.        label3 = QLabel('Label 3')
13.        button1 = QPushButton('Button 1')
14.        button2 = QPushButton('Button 2')
15.        button3 = QPushButton('Button 3')
16.        textbox1 = QLineEdit()
17.        textbox2 = QLineEdit()
18.        textbox3 = QLineEdit()
19.
20.        # Create a QVBoxLayout
21.        vbox_layout = QVBoxLayout()
22.        vbox_layout.addWidget(QLabel('Vertical Layout'))
23.        vbox_layout.addWidget(label1)
24.        vbox_layout.addWidget(button1)
25.        vbox_layout.addWidget(textbox1)
26.

```

```

27.         # Create a QHBoxLayout
28.         hbox_layout = QHBoxLayout()
29.         hbox_layout.addWidget(QLabel('Horizontal Layout'))
30.         hbox_layout.addWidget(label2)
31.         hbox_layout.addWidget(button2)
32.         hbox_layout.addWidget(textbox2)
33.
34.         # Create a QGridLayout
35.         grid_layout = QGridLayout()
36.         grid_layout.addWidget(QLabel('Grid Layout'), 0, 0, 1, 2)
37.         grid_layout.addWidget(label3, 1, 0)
38.         grid_layout.addWidget(button3, 1, 1)
39.         grid_layout.addWidget(textbox3, 2, 0, 1, 2)
40.
41.         # Create a QVBoxLayout for the main layout and add the other layouts
42.         main_layout = QVBoxLayout()
43.         main_layout.addLayout(vbox_layout)
44.         main_layout.addLayout(hbox_layout)
45.         main_layout.addLayout(grid_layout)
46.
47.         # Create a QWidget, set its layout, and set it as the central widget
48.         container = QWidget()
49.         container.setLayout(main_layout)
50.         self.setCentralWidget(container)
51.
52. app = QApplication([])
53. window = MainWindow()
54. window.show()
55. app.exec_()
56.

```

2. Amati hasilnya kemudian tulis analisis singkat mengenai kegiatan ini

8.3.2 Kegiatan 2 : Widget

1. Buat sebuah file program baru kemudian tulis kode program berikut ini

```

1. from PyQt5.QtWidgets import QApplication, QMainWindow, QLabel, QPushButton, QLineEdit, QVBoxLayout, QWidget
2.
3. class MainWindow(QMainWindow):

```



```

4.     def __init__(self):
5.         super().__init__()
6.
7.         # Set the window title
8.         self.setWindowTitle('My App')
9.
10.        # Create a QLabel, QPushButton, and QLineEdit
11.        self.label = QLabel()
12.        self.button = QPushButton('Show Text')
13.        self.textbox = QLineEdit()
14.
15.        # Connect the button's clicked signal to the slot
16.        self.button.clicked.connect(self.on_button_clicked)
17.
18.        # Create a QVBoxLayout and add the widgets
19.        layout = QVBoxLayout()
20.        layout.addWidget(self.textbox)
21.        layout.addWidget(self.button)
22.        layout.addWidget(self.label)
23.
24.        # Create a QWidget, set its layout, and set it as the central widget
25.        container = QWidget()
26.        container.setLayout(layout)
27.        self.setCentralWidget(container)
28.
29.        # Define the slot that will be called when the button is clicked
30.        def on_button_clicked(self):
31.            # Get the text from the QLineEdit and set it as the QLabel's text
32.            text = self.textbox.text()
33.            self.label.setText(text)
34.
35.        # Create a QApplication, create a MainWindow, show the main window, and start the event loop
36.        app = QApplication([])
37.        window = MainWindow()
38.        window.show()
39.        app.exec_()

```

2. Amati hasilnya kemudian tulis analisis singkat mengenai kegiatan ini

8.3.3 Kegiatan 3 : Signal dan Slot

1. Buat sebuah file program baru kemudian tulis kode program berikut ini

```
1. from PyQt5.QtWidgets import QApplication, QMainWindow, QLabel, QPush
   shButton, QLineEdit, QVBoxLayout, QWidget
2.
3. class MainWindow(QMainWindow):
4.     def __init__(self):
5.         super().__init__()
6.
7.         # Set the window title
8.         self.setWindowTitle('Kalkulator Sederhana')
9.
10.        # Create two QLineEdit widgets for the user to enter numbe
        rs
11.        self.first_number = QLineEdit()
12.        self.second_number = QLineEdit()
13.
14.        # Create a QLabel to display input and the result
15.        self.labelinput1 = QLabel("Masukkan Angka 1")
16.        self.labelinput2 = QLabel("Masukkan Angka 2")
17.        self.hasil_label = QLabel("Hasil Perhitungan ")
18.        self.result_label = QLabel()
19.
20.        # Create four QPushButton widgets for the mathematical ope
        rations
21.        self.add_button = QPushButton('Tambah')
22.        self.subtract_button = QPushButton('Kurang')
23.        self.multiply_button = QPushButton('Kali')
24.        self.divide_button = QPushButton('Bagi')
25.
26.        # Connect each button's clicked signal to the appropriate
        slot
27.        self.add_button.clicked.connect(self.add_numbers)
28.        self.subtract_button.clicked.connect(self.subtract_numbers
        )
29.        self.multiply_button.clicked.connect(self.multiply_numbers
        )
30.        self.divide_button.clicked.connect(self.divide_numbers)
31.
32.        # Create a QVBoxLayout and add the widgets
33.        layout = QVBoxLayout()
34.        layout.addWidget(self.labelinput1)
```

```

35.         layout.addWidget(self.first_number)
36.         layout.addWidget(self.labelinput2)
37.         layout.addWidget(self.second_number)
38.         layout.addWidget(self.add_button)
39.         layout.addWidget(self.subtract_button)
40.         layout.addWidget(self.multiply_button)
41.         layout.addWidget(self.divide_button)
42.         layout.addWidget(self.hasil_label)
43.         layout.addWidget(self.result_label)
44.
45.         # Create a QWidget, set its layout, and set it as the central widget
46.         container = QWidget()
47.         container.setLayout(layout)
48.         self.setCentralWidget(container)
49.
50.         # Define the slots that will be called when the buttons are clicked
51.         def add_numbers(self):
52.             # Add the numbers entered by the user and display the result
53.             result = float(self.first_number.text()) + float(self.second_number.text())
54.             self.result_label.setText(str(result))
55.
56.         def subtract_numbers(self):
57.             # Subtract the second number from the first and display the result
58.             result = float(self.first_number.text()) - float(self.second_number.text())
59.             self.result_label.setText(str(result))
60.
61.         def multiply_numbers(self):
62.             # Multiply the numbers entered by the user and display the result
63.             result = float(self.first_number.text()) * float(self.second_number.text())
64.             self.result_label.setText(str(result))
65.
66.         def divide_numbers(self):
67.             # Divide the first number by the second and display the result
68.             result = float(self.first_number.text()) / float(self.second_number.text())
69.             self.result_label.setText(str(result))
70.

```

```
71. # Create a QApplication, create a MainWindow, show the main window  
    , and start the event loop  
72. app = QApplication([])  
73. window = MainWindow()  
74. window.show()  
75. app.exec_()  
76.
```

2. Amati hasilnya kemudian tulis analisis singkat mengenai kegiatan ini

8.4 Tugas

Berdasarkan pada Praktikum Kegiatan 3, tambahkan button Modulus yang akan menghasilkan sisa pembagian, dan button Pangkat yang akan menghasilkan hasil pangkat dari angka pertama dengan pangkat angka ke dua.