BAB 9

PyQt

9.1 Tujuan

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

9.2 Pengantar

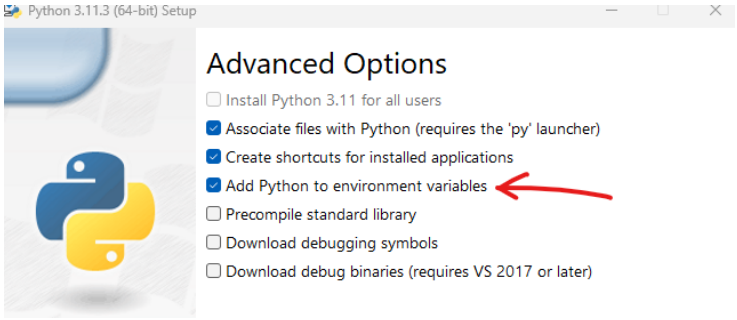
9.2.1 Pendahuluan PyQt

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 PyQt5 dan PyQt6. Versi ini merujuk pada versi Qt library yang digunakan. PyQt6 mendukung Qt6, dan PyQt5 mendukung Qt5

9.2.2 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 ketikan perintah berikut untuk melakukan instalasi

```
1. pip install pyqt6
```

```
C:\Users\AIO ASUS>pip install pyqt6
Collecting pyqt6
  Downloading PyQt6-6.9.0-cp39-abi3-win_amd64.whl (6.7 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 6.7/6.7 MB 8.8 MB/s eta 0:00:00
Collecting PyQt6-sip<14,>=13.8
  Downloading PyQt6_sip-13.10.0-cp311-cp311-win_amd64.whl (53 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 53.4/53.4 kB ? eta 0:00:00
Collecting PyQt6-Qt6<6.10.0,>=6.9.0
  Downloading PyQt6_Qt6-6.9.0-py3-none-win_amd64.whl (73.7 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 73.7/73.7 MB 5.2 MB/s eta 0:00:00
Installing collected packages: PyQt6-Qt6, PyQt6-sip, pyqt6
Successfully installed PyQt6-Qt6-6.9.0 PyQt6-sip-13.10.0 pyqt6-6.9.0

[notice] A new release of pip available: 22.3.1 -> 25.1.1
[notice] To update, run: python.exe -m pip install --upgrade pip
```

Step 3. Verifikasi Instalasi

Setelah instalasi selesai ketikkan perintah berikut pada python interpreter

```
1. import PyQt6
```

Jika tidak ada error maka proses instalasi PyQt6 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 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import PyQt6
>>>
```

9.2.3 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 PyQt6.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

9.2.4 Struktur Aplikasi PyQT

Aplikasi PyQt biasanya memiliki struktur berikut:

- Import modul yang diperlukan: Melakukan import terhadap modul-modul yang diperlukan termasuk modul PyQt6.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 QGridLayout.
- 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 PyQt6 menggunakan konsep pemrograman berorientasi objek (OOP):

```
# Import necessary modules from PyQt6

from PyQt6.QtWidgets import QApplication, QMainWindow, QLabel

# Define a MainWindow class that inherits from QMainWindow
class MainWindow(QMainWindow):
    def __init__(self):
        # Call the constructor of the parent class
        super().__init__()

        # Set some basic attributes for the window
        self.title = "Hello PyQt6"
        self.top = 100
        self.left = 100
        self.width = 400
        self.height = 300

        # Call the method that initializes the window
        self.initWindow()

    def initWindow(self):
        # Create a QLabel widget and set it as the central widget of the window
        self.label = QLabel("Welcome to PyQt6", self)
        self.label.adjustSize()  # Adjust the size of the label to fit the text
        self.label.move(100, 100)  # Move the label to the position (100, 100)

        # Set the title of the window
        self.setWindowTitle(self.title)
        # Set the position and size of the window
        self.setGeometry(self.top, self.left, self.width, self.height)
        # Show the window
        self.show()

# Create an instance of QApplication
app = QApplication([])
# Create an instance of MainWindow
window = MainWindow()
```

```
# Start the event loop
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().

9.2.5 Widget dan Layout

9.2.5.1 Qlabel

QLabel adalah widget yang digunakan untuk menampilkan teks atau gambar

```
from PyQt6.QtWidgets import QApplication, QMainWindow, QLabel

class MainWindow(QMainWindow):
    def __init__(self):
        super().__init__()

        self.label = QLabel('Hello PyQt!', self)
        self.label.adjustSize()

app = QApplication([])
window = MainWindow()
window.show()
app.exec()
```

9.2.5.2 QPushButton

QPushButton adalah tombol yang bisa diklik oleh pengguna

.

```
1. from PyQt6.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()
```

9.2.5.3 QLineEdit

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

```
1. from PyQt6.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()
```

9.2.5.4 QTextEdit

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

```
1. from PyQt6.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.
```

9.2.5.5 QCheckBox

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

```
1. from PyQt6.QtWidgets import QApplication, QMainWindow, QCheckBox
2. from PyQt6.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:
```

```
            print('Unchecked')

app = QApplication([])
window = MainWindow()
window.show()
app.exec()

```

9.2.5.6 QRadioButton

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

```
from PyQt6.QtWidgets import QApplication, QMainWindow, QRadioButto
n

class MainWindow(QMainWindow):
    def __init__(self):
        super().__init__()

        self.radioButton = QRadioButton('Select Me', self)
        self.radioButton.toggled.connect(self.on_toggled)

    def on_toggled(self, is_checked):
        if is_checked:
            print('Selected')

app = QApplication([])
window = MainWindow()
window.show()
app.exec()

```

9.2.5.7 QcomboBox

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

```
from PyQt6.QtWidgets import QApplication, QMainWindow, QComboBox

class MainWindow(QMainWindow):
```

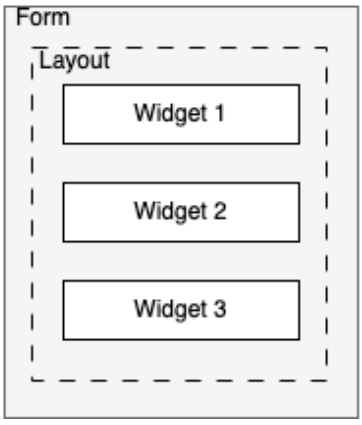
```
    def __init__(self):
        super().__init__()

        self.comboBox = QComboBox(self)
        self.comboBox.addItems(['Option 1', 'Option 2', 'Option 3'])
        self.comboBox.currentIndexChanged.connect(self.on_index_changed)

    def on_index_changed(self, index):
        print('Selected option:', self.comboBox.itemText(index))

app = QApplication([])
window = MainWindow()
window.show()
app.exec()

```

9.2.5.8 Qslider

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

```
from PyQt6.QtWidgets import QApplication, QMainWindow, QSlider
from PyQt6.QtCore import Qt

class MainWindow(QMainWindow):
    def __init__(self):
        super().__init__()

        self.slider = QSlider(Qt.Horizontal, self)
        self.slider.setMinimum(0)
        self.slider.setMaximum(100)
        self.slider.valueChanged.connect(self.on_value_changed)

    def on_value_changed(self, value):
        print('Slider value:', value)

app = QApplication([])
window = MainWindow()
window.show()
app.exec()

```

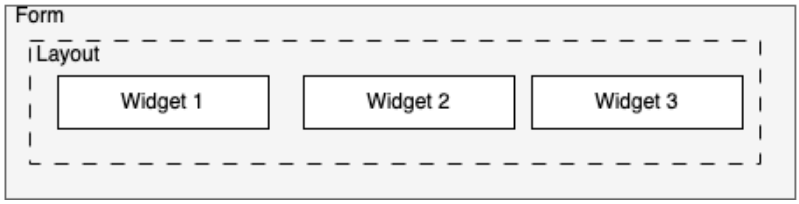
9.2.5.9 QVBoxLayout



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

```
1. from PyQt6.QtWidgets import QApplication, QWidget, QVBoxLayout, QPu
   shButton
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()
```

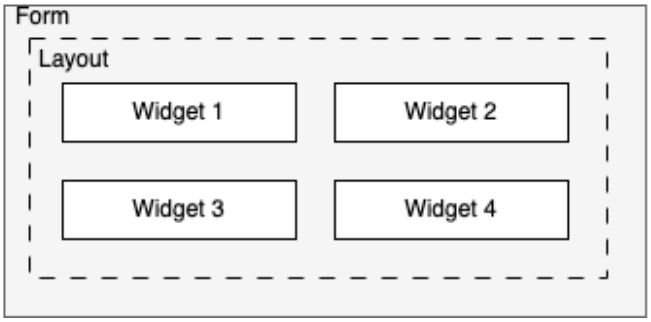
9.2.5.10 QHBoxLayout



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

```
1. from PyQt6.QtWidgets import QApplication, QWidget, QHBoxLayout, QPu
   shButton
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()
```

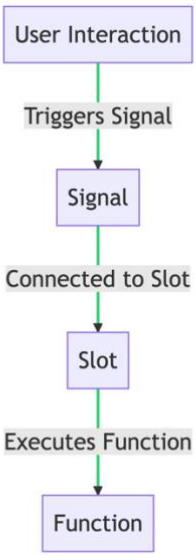
9.2.5.11 QGridLayout



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

```
1. from PyQt6.QtWidgets import QApplication, QWidget, QGridLayout, QPu
   shButton
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()
```

9.2.6 Signal dan Slot



Signal dan Slot adalah fitur fundamental dari PyQt6 dan merupakan mekanisme yang memungkinkan komunikasi antara objek dalam aplikasi GUI.

Signal

Signal dalam PyQt6 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 PyQt6:

```
1. from PyQt6.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.
```

```
app = QApplication([])
window = MainWindow()
window.show()
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.

9.3 Kegiatan Praktikum

9.3.1 Kegiatan 1 : Layout

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

```
from PyQt6.QtWidgets import QApplication, QMainWindow, QLabel, QPushButton, QLineEdit, QVBoxLayout, QHBoxLayout, QGridLayout, QWidget

class MainWindow(QMainWindow):
    def __init__(self):
        super().__init__()

        self.setWindowTitle('Layouts Example')

        # Create some widgets
        label1 = QLabel('Label 1')
        label2 = QLabel('Label 2')
        label3 = QLabel('Label 3')
        button1 = QPushButton('Button 1')
        button2 = QPushButton('Button 2')
        button3 = QPushButton('Button 3')
        textbox1 = QLineEdit()
        textbox2 = QLineEdit()
        textbox3 = QLineEdit()

        # Create a QVBoxLayout
        vbox_layout = QVBoxLayout()
        vbox_layout.addWidget(QLabel('Vertical Layout'))
        vbox_layout.addWidget(label1)
        vbox_layout.addWidget(button1)
        vbox_layout.addWidget(textbox1)

```

```
        # Create a QHBoxLayout
        hbox_layout = QHBoxLayout()
        hbox_layout.addWidget(QLabel('Horizontal Layout'))
        hbox_layout.addWidget(label2)
        hbox_layout.addWidget(button2)
        hbox_layout.addWidget(textbox2)

        # Create a QGridLayout
        grid_layout = QGridLayout()
        grid_layout.addWidget(QLabel('Grid Layout'), 0, 0, 1, 2)
        grid_layout.addWidget(label3, 1, 0)
        grid_layout.addWidget(button3, 1, 1)
        grid_layout.addWidget(textbox3, 2, 0, 1, 2)

        # Create a QVBoxLayout for the main layout and add the othe
    r layouts
        main_layout = QVBoxLayout()
        main_layout.addLayout(vbox_layout)
        main_layout.addLayout(hbox_layout)
        main_layout.addLayout(grid_layout)

        # Create a QWidget, set its layout, and set it as the centr
    al widget
        container = QWidget()
        container.setLayout(main_layout)
        self.setCentralWidget(container)

app = QApplication([])
window = MainWindow()
window.show()
app.exec()

```

2. Amati hasilnya kemudian tulis analisis singkat mengenai kegiatan ini

9.3.2 Kegiatan 2 : Widget

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

```
from PyQt6.QtWidgets import QApplication, QMainWindow, QLabel, QPus
    hButton, QLineEdit, QVBoxLayout, QWidget

class MainWindow(QMainWindow):
```

```
    def __init__(self):
        super().__init__()

        # Set the window title
        self.setWindowTitle('My App')

        # Create a QLabel, QPushButton, and QLineEdit
        self.label = QLabel()
        self.button = QPushButton('Show Text')
        self.textbox = QLineEdit()

        # Connect the button's clicked signal to the slot
        self.button.clicked.connect(self.on_button_clicked)

        # Create a QVBoxLayout and add the widgets
        layout = QVBoxLayout()
        layout.addWidget(self.textbox)
        layout.addWidget(self.button)
        layout.addWidget(self.label)

        # Create a QWidget, set its layout, and set it as the central widget
        container = QWidget()
        container.setLayout(layout)
        self.setCentralWidget(container)

    # Define the slot that will be called when the button is clicked
    def on_button_clicked(self):
        # Get the text from the QLineEdit and set it as the QLabel's text
        text = self.textbox.text()
        self.label.setText(text)

# Create a QApplication, create a MainWindow, show the main window, and start the event loop
app = QApplication([])
window = MainWindow()
window.show()
app.exec()
```

2. Amati hasilnya kemudian tulis analisis singkat mengenai kegiatan ini

9.3.3 Kegiatan 3 : Signal dan Slot

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

```
1. from PyQt6.QtWidgets import QApplication, QMainWindow, QLabel, QPu
   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)
```

```
        layout.addWidget(self.first_number)
        layout.addWidget(self.labelinput2)
        layout.addWidget(self.second_number)
        layout.addWidget(self.add_button)
        layout.addWidget(self.subtract_button)
        layout.addWidget(self.multiply_button)
        layout.addWidget(self.divide_button)
        layout.addWidget(self.hasil_label)
        layout.addWidget(self.result_label)

        # Create a QWidget, set its layout, and set it as the central widget
        container = QWidget()
        container.setLayout(layout)
        self.setCentralWidget(container)

    # Define the slots that will be called when the buttons are clicked
    def add_numbers(self):
        # Add the numbers entered by the user and display the result
        result = float(self.first_number.text()) + float(self.second_number.text())
        self.result_label.setText(str(result))

    def subtract_numbers(self):
        # Subtract the second number from the first and display the result
        result = float(self.first_number.text()) - float(self.second_number.text())
        self.result_label.setText(str(result))

    def multiply_numbers(self):
        # Multiply the numbers entered by the user and display the result
        result = float(self.first_number.text()) * float(self.second_number.text())
        self.result_label.setText(str(result))

    def divide_numbers(self):
        # Divide the first number by the second and display the result
        result = float(self.first_number.text()) / float(self.second_number.text())
        self.result_label.setText(str(result))

```

```
# Create a QApplication, create a MainWindow, show the main window
    , and start the event loop
app = QApplication([])
window = MainWindow()
window.show()
app.exec()
```

2. Amati hasilnya kemudian tulis analisis singkat mengenai kegiatan ini

9.4Tugas

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.