PySide6 Development

Developing Application with Object-Oriented Design

Object-Oriented Programming 2/2567

@wichit2s

https://wichit2s.github.io/courses/oop/

Hourly Agenda

Hour 1: Introduction to PySide6

Hour 2: Widgets and Layouts

Hour 3: Events, Signals, and Slots

Hour 4: Mini-Project

Developing Application with Object-Oriented Design
Object-Oriented Programming 2/2567

@wichit2s
https://wichit2s.github.io/courses/oop/

Objective:
Understand the basics of PySide6 and its ecosystem

Hour 1

Introduction to PySide6

Content:
1. What is PySide6?
2. Setting up the environment:
    1. Python installation
    2. Installing PySide6
3. Overview of PySide6 modules
4. First PySide6 Program
5. Anatomy of a PySide6 application

Hands-on Activity:
Write and run a basic PySide6 application.

Developing game with Object-Oriented Design
Object-Oriented Programming 2/2567

@wichit2s
https://wichit2s.github.io/courses/oop/

What is PySide6?

https://wiki.qt.io/Qt_for_Python

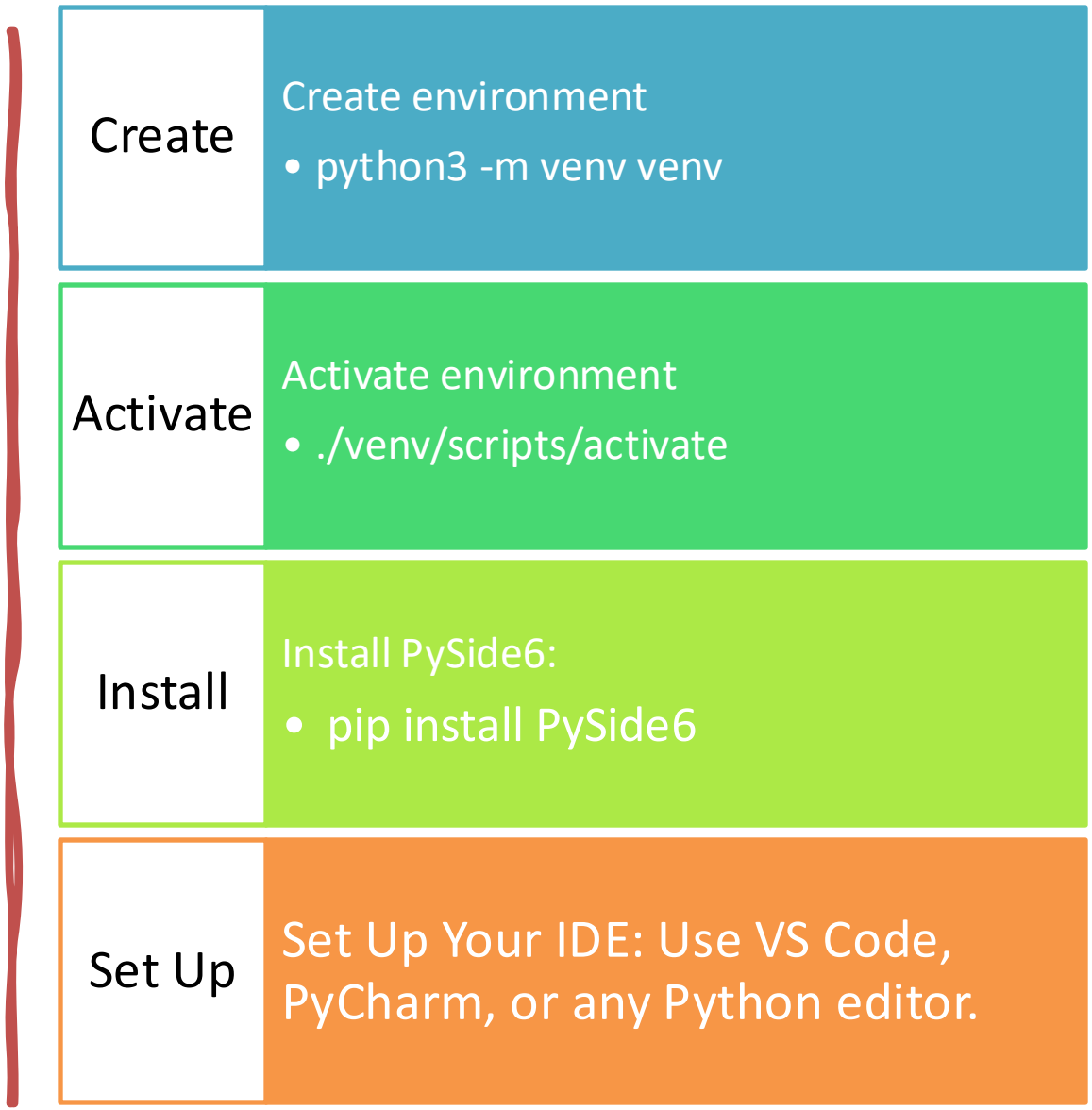
Definition:
PySide6 is the official Python binding for **Qt6**, a powerful framework for building cross-platform applications with graphical user interfaces (GUIs).

Key Features:
- Cross-platform compatibility (Windows, macOS, Linux, etc.).
- Extensive widgets and tools for GUI development.
- Support for modern features like animations, data visualization, and multimedia.

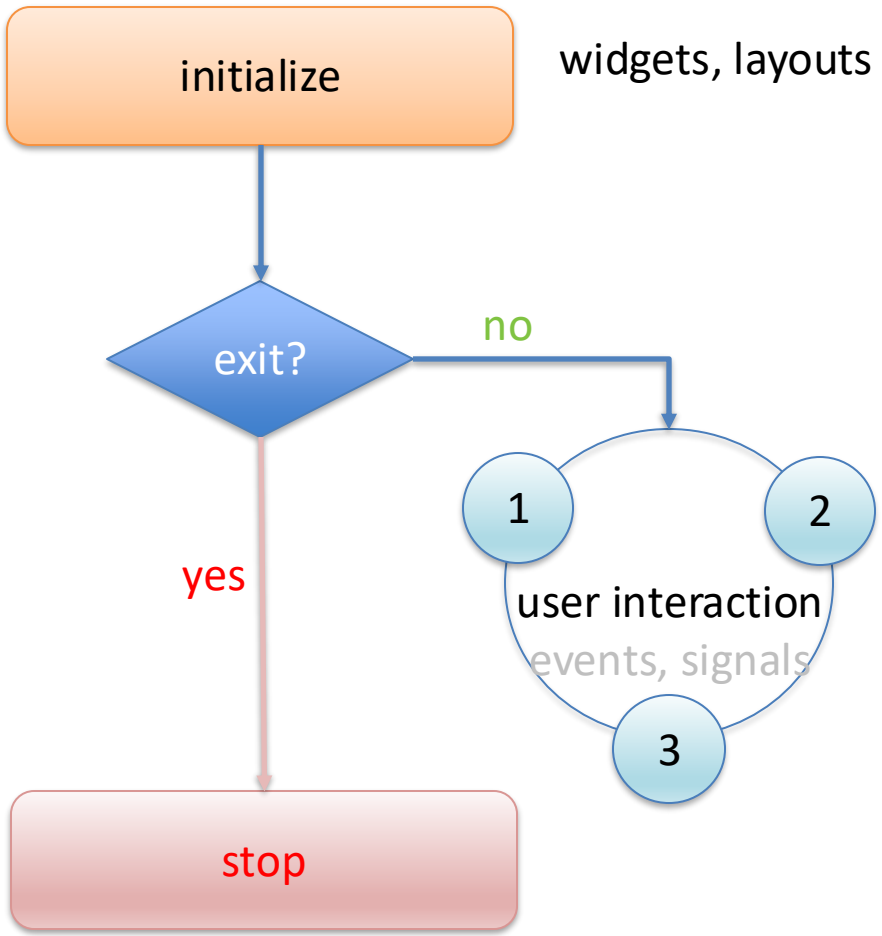
Why Learn PySide6?
- Easy-to-use API for Python developers.
- Create professional-grade **desktop applications**.
- Backed by a strong Qt ecosystem.
- Growing demand for custom GUI applications.

https://doc.qt.io/qtforpython-6/

Installation and Setup



Window Application Life Cycle Basics

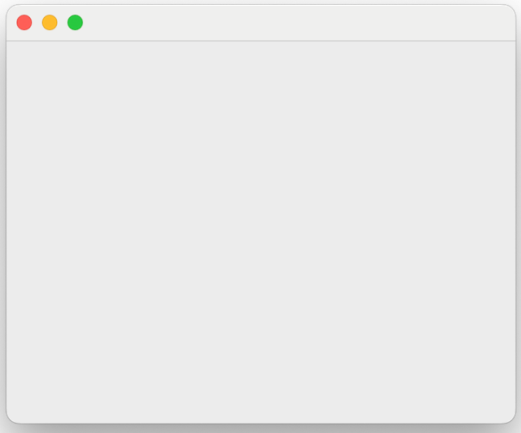


Application Loop

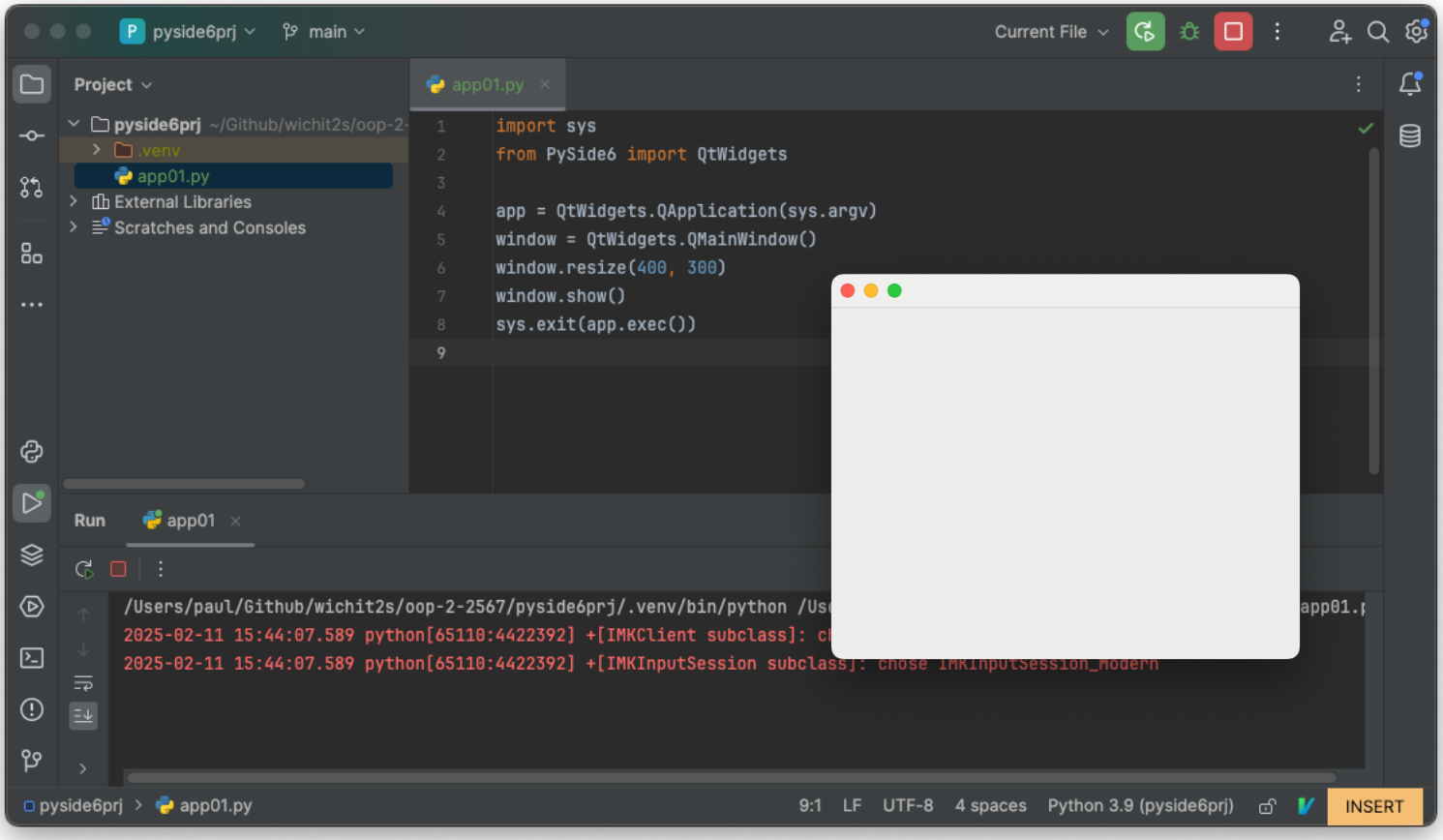
```
import sys
from PySide6.QtWidgets import *

app = QApplication(sys.argv)
window = QMainWindow()
window.resize(800, 600)
window.show()

sys.exit(app.exec())
```



Hands-on Activity



Objective:
Learn to use PySide6 widgets and layouts effectively.

Hour 2

Widgets and Layouts

Content:

1. Commonly used widgets:

   QPushButton, QLabel, QLineEdit, QComboBox

2. Layout management:

   QVBoxLayout, QHBoxLayout, QGridLayout

3. Nesting layouts and managing complex UIs.

4. Adding functionality to widgets (signals and slots).

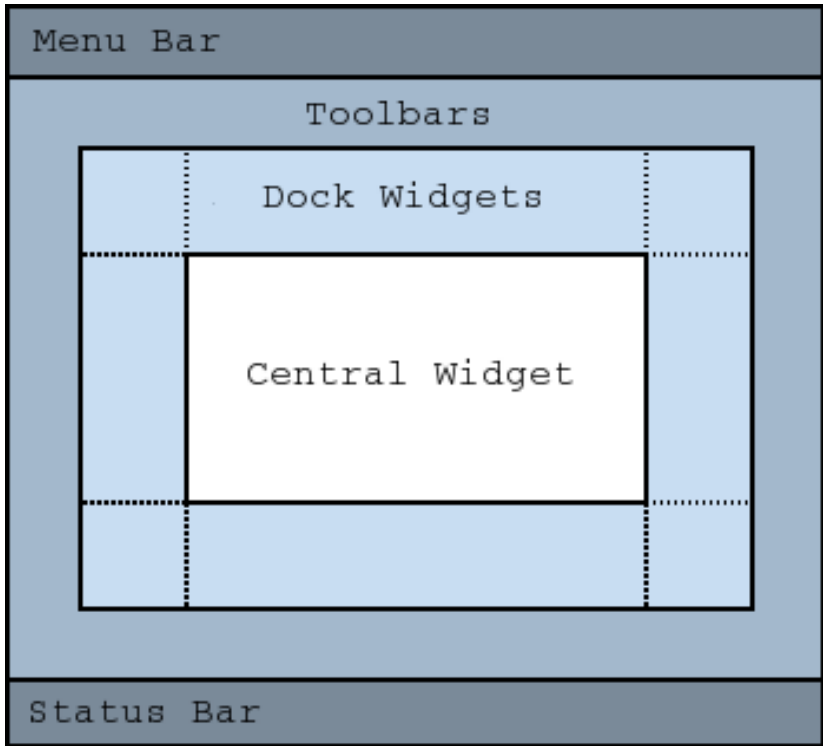
Hands-on Activity:
Build a simple form with labels, text inputs, and a submit button.

Developing game with Object-Oriented Design
Object-Oriented Programming 2/2567

@wichit2s
https://wichit2s.github.io/courses/oop/

QtWidgets QMainWindow

- Provides a framework for building standard GUI applications.
  - **Menu Bar**: for application menus
  - **Tool Bar**: for quick-access tools.
  - **Dock Widgets**: dockable widgets
  - **Central Widget**: content area of the windwo
  - **Status Bar**: for displaying feedback messages.



https://doc.qt.io/qtforpython-6.5/PySide6/QtWidgets/QMainWindow.html

OOPChat

```
class OOPChat(QMainWindow):
  def __init__(self):
    super().__init__()
    self.resize(800, 600)
    self.setWindowTitle('OOP Chat’)
    # create central widget
    logo = QPixmap('whitehaticon.jpg’)
    central_widget = QLabel(pixmap=logo)
    self.setCentralWidget(central_widget)

if __name__ == "__main__":
  app = QApplication(sys.argv)
  window = OOPChat()
  window.show()
  sys.exit(app.exec())
```

OOPChat

```
class OOPChat(QMainWindow):
  def __init__(self):
    super().__init__()
    self.resize(800, 600)
    self.setWindowTitle('OOP Chat’)
    self.create_central_widget()

  def create_central_widget(self): pass #???
  def create_menubar(self): pass
  def create_toolbar(self): pass
  def create_statusbar(self): pass

if __name__ == "__main__":
  app = QApplication(sys.argv)
  window = OOPChat()
  window.show()
  sys.exit(app.exec())
```

Widgets

QLineEdit

QPushButton

QTextEdit

QPlainTextEdit

QLabel

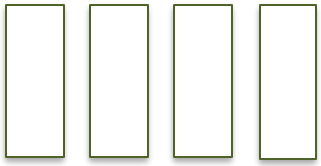
QCheckBox

2 QSpinBox

QRadioButton

QDial

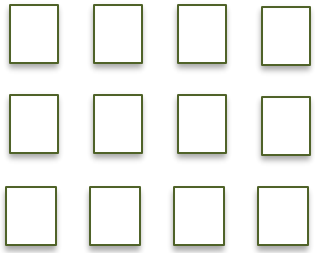
Layouts



QHBoxLayout

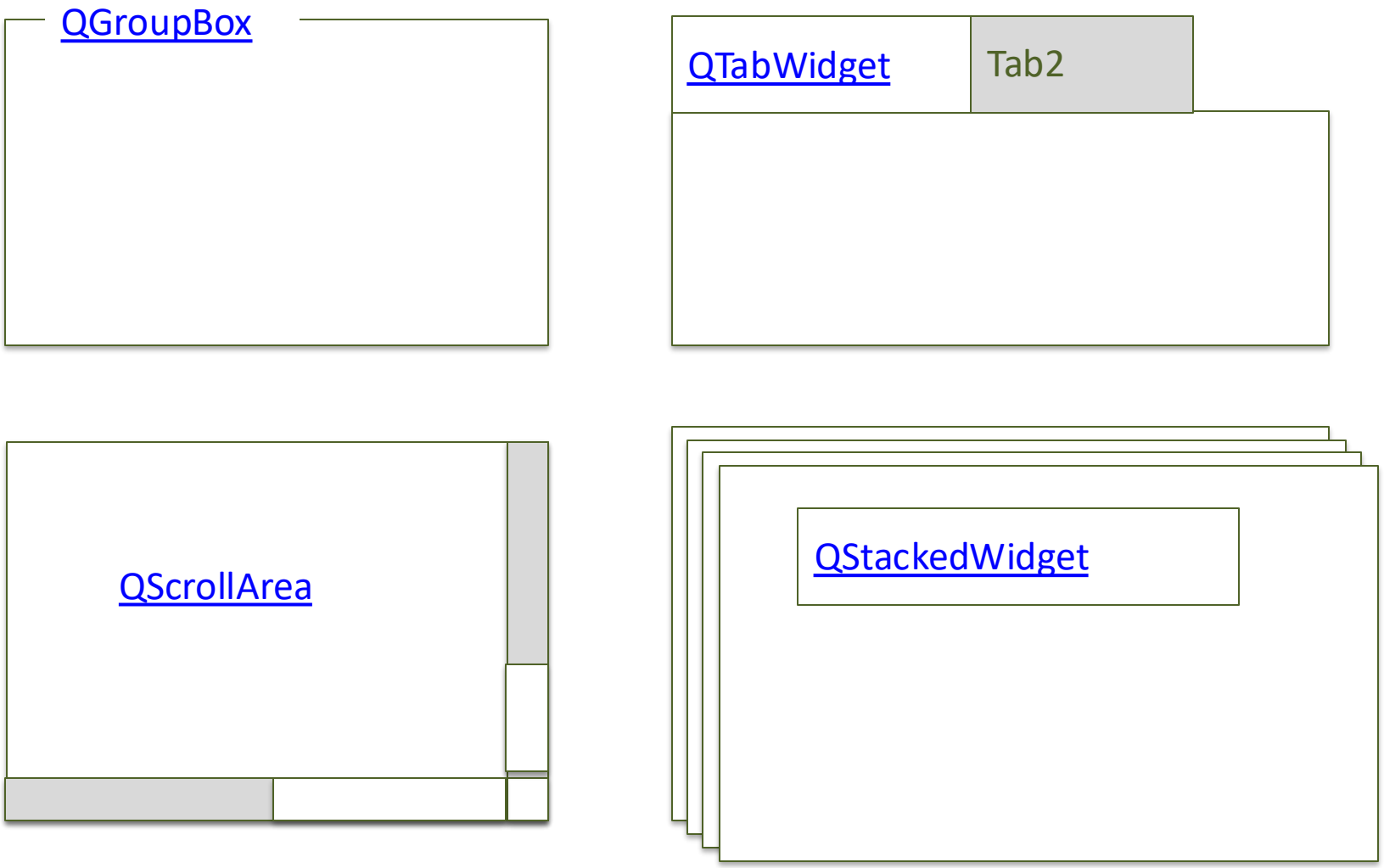
QFormLayout

QVBoxLayout

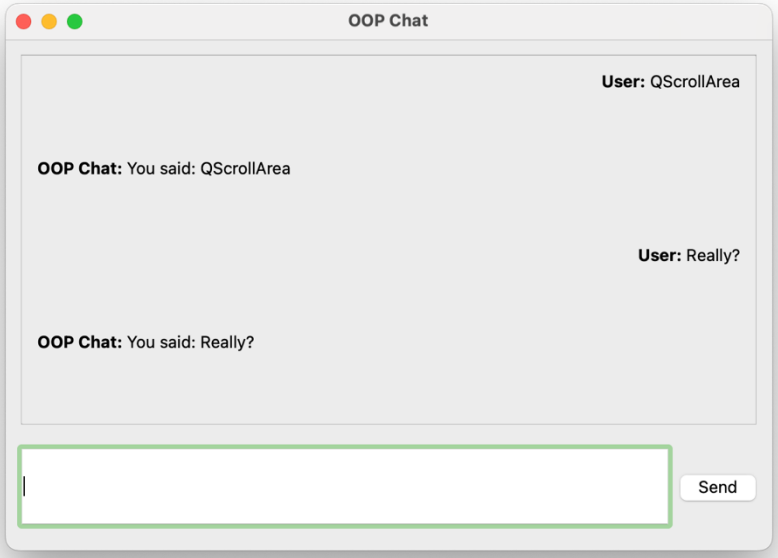


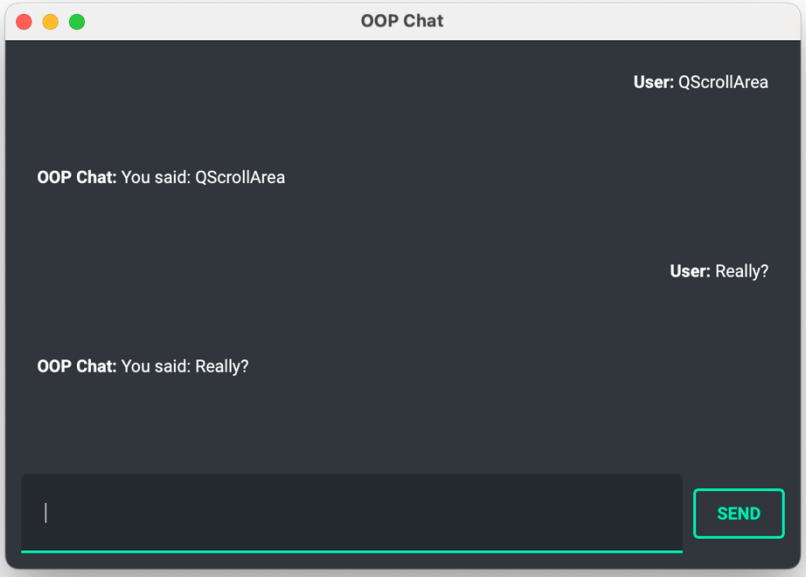
QGridLayout

Containers



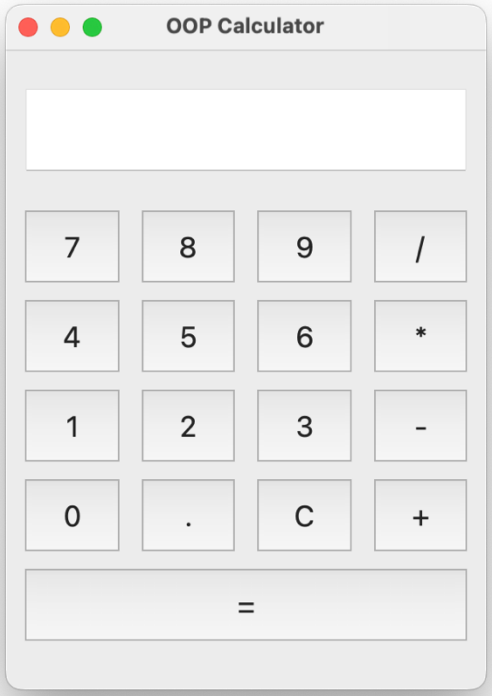
Hands-on Activity



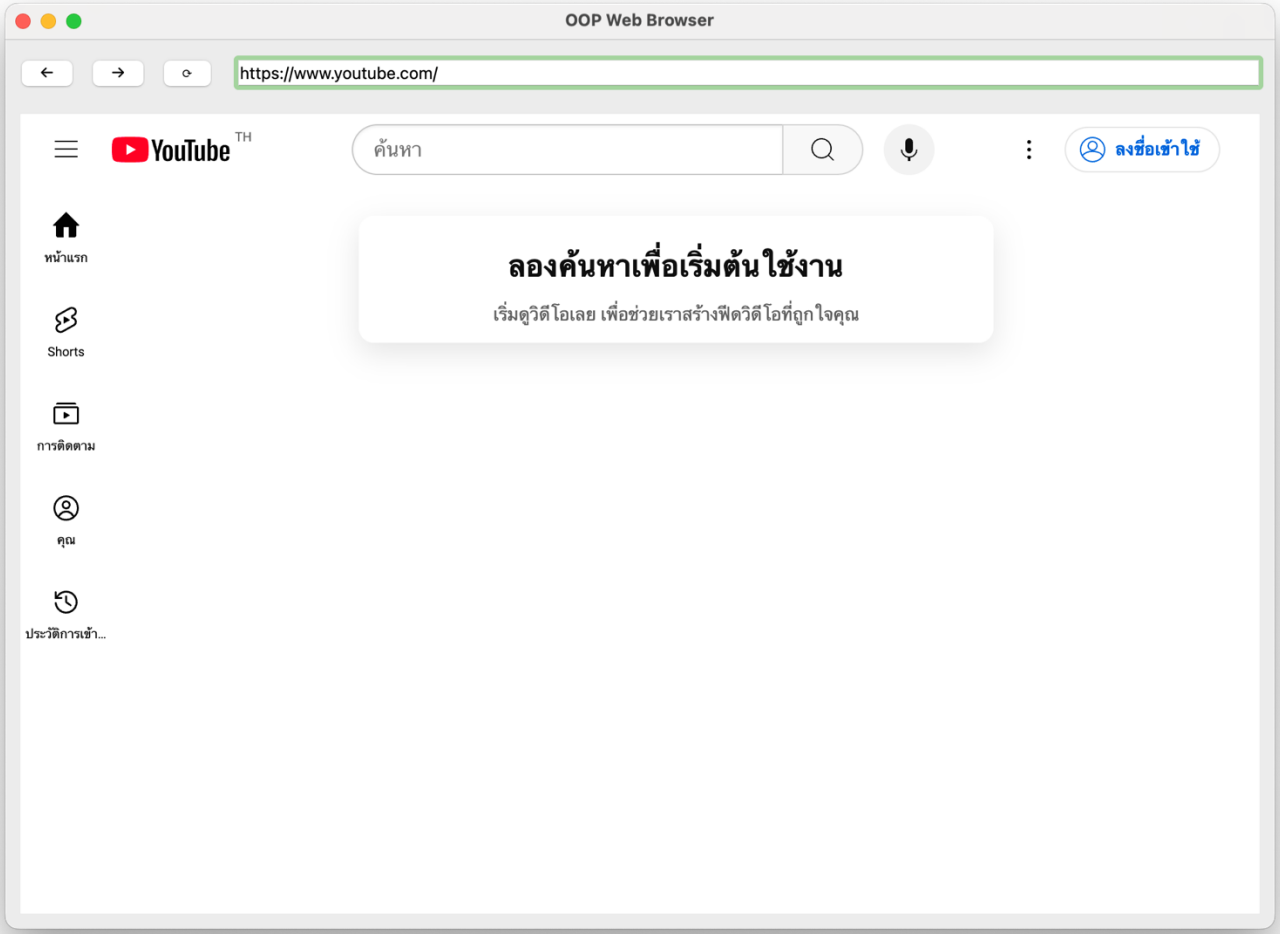


https://qt-material.readthedocs.io/en/latest/index.html

Hands-on Activity



Hands-on Activity



Objective:
Understand the event system in PySide6 and how to connect signals to slots.

Content:
1. What are signals and slots?
2. Built-in signals and creating custom signals.
3. Handling user inputs and interactions
4. Event handling with custom event filters.

Hour 3

Events, Signals, and Slots

Hands-on Activity:
Create an interactive application where a button click updates a label.

Developing game with Object-Oriented Design
Object-Oriented Programming 2/2567

@wichit2s
https://wichit2s.github.io/courses/oop/

What?

- **What is an Event?**
  - Events are user interactions such as **mouse clicks, key presses, and window actions**.
  - PySide6 processes events using an **event-driven** mechanism.

- **What are Signals and Slots?**
  - **Signals** are emitted when an event occurs (e.g., button click).
  - **Slots** are functions that handle these signals (e.g., updating UI).
  - Signals and slots provide a **flexible** and **declarative** way to handle interactions.

- **Why Are They Important?**
  - Enable **separation of concerns** between UI components.
  - Allow **reactive programming** in PySide6.
  - **Decouple** components (e.g., a button doesn't need to know how a label updates).

```
button.clicked.connect(self.on_button_clicked)
```

Qt Event System

What are Qt events?

- Mouse events (mousePressEvent, mouseMoveEvent)
- Keyboard events (keyPressEvent, keyReleaseEvent)
- Window events (resizeEvent, closeEvent)
- Custom events (user-defined interactions)

```
from PySide6.QtWidgets import QApplication, QWidget
from PySide6.QtGui import QMouseEvent

class MyWidget(QWidget):
    def mousePressEvent(self, event: QMouseEvent):
        print(f"Mouse clicked at: {event.pos()}")

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

Handling Event

How does it work?

- Every widget in PySide6 has an `event()` method that gets called whenever an event occurs.
- If an event is not handled, it is passed to the **default event handler** (superclass).
- It receives all events before specific event handlers like mousePressEvent() or keyPressEvent().

```
class MyWidget(QWidget):
  def event(self, event):
    if event.type() == QEvent.KeyPress:
      print(f"Key pressed: {event.key()}")
    elif event.type() == QEvent.MouseButtonPress:
      print(f"Mouse clicked at: {event.pos()}")
    return super().event(event)
```

Handling Mouse Event

```python
from PySide6.QtWidgets import QApplication, QWidget
from PySide6.QtGui import QMouseEvent

class MyWidget(QWidget):
    def mousePressEvent(self, event: QMouseEvent):
        print(f"{event.button()} @ {event.pos()}")
        b = event.button()
        print(b == Qt.LeftButton)
        print(b == Qt.RightButton)
    def mouseReleaseEvent(self, event):
        pass
    def mouseMoveEvent(self, event):
        pass
app = QApplication([])
window = MyWidget()
window.show()
app.exec()
```

Handling Keyboard Event

```python
from PySide6.QtWidgets import QApplication, QWidget
from PySide6.QtGui import QKeyEvent

class MyWidget(QWidget):
    def keyPressEvent(self, event: QKeyEvent):
        print(f"{event.key()} : {event.text()}")

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

Signals and Slots

- **Signals**: Messages emitted when an event occurs (e.g., button click).
- **Slots**: Functions that respond to signals (e.g., updating text, closing a window).
- **Purpose**: Enables **communication** between objects in a decoupled way.

Widget	Signal	Description
QPushButton	clicked()	Triggered when clicked
QLineEdit	textChanged(str)	Emitted when text changes
QSlider	valueChanged(int)	Emitted when slider moves
QCheckBox	toggled(bool)	Triggered when checkbox state changes

Button Clicked → Signal Emitted → Slot Executed → UI Updated

Objective:
Advanced Topics and Mini-Project

Hour 4

Mini-Project

Content:
1. Using QtDesigner for rapid GUI prototyping.
2. Working with .ui files
3. Introduction to styling
4. Mini-project: Build a simple To-Do List application

Hands-on Activity:
Guide participants through the mini-project step-by-step.

Developing game with Object-Oriented Design
Object-Oriented Programming 2/2567

@wichit2s
https://wichit2s.github.io/courses/oop/

pyside6-designer