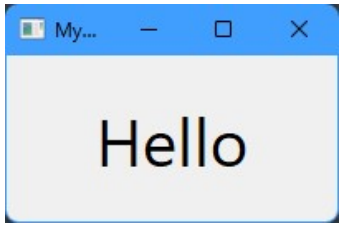
Pemrograman Desktop 3

Yonathan F. Hendrawan

Qlabel: One-line piece of text to show information

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

class MainWindow(QMainWindow):
    def __init__(self):
        super().__init__()
        self.setWindowTitle("My App")
        widget = QLabel("Hello")
        font = widget.font()
        font.setPointSize(30)
        widget.setFont(font)
        widget.setAlignment(Qt.AlignmentFlag.AlignHCenter | Qt.AlignmentFlag.AlignVCenter)
        self.setCentralWidget(widget)

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



Horizontal & Vertical Alignment

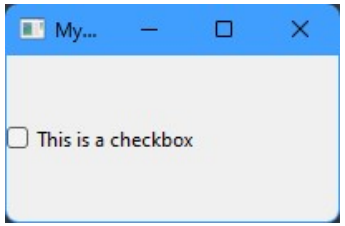
Flag	Behavior
`Qt.AlignmentFlag.AlignLeft`	Aligns with the left edge.
`Qt.AlignmentFlag.AlignRight`	Aligns with the right edge.
`Qt.AlignmentFlag.AlignHCenter`	Centers horizontally in the available space.
`Qt.AlignmentFlag.AlignJustify`	Justifies the text in the available space.

Flag	Behavior
`Qt.AlignmentFlag.AlignTop`	Aligns with the top.
`Qt.AlignmentFlag.AlignBottom`	Aligns with the bottom.
`Qt.AlignmentFlag.AlignVCenter`	Centers vertically in the available space.

Flag	Behavior
`Qt.AlignmentFlag.AlignCenter`	Centers horizontally **and** vertically

QCheckBox: presents a checkable box to the user

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

class MainWindow(QMainWindow):
    def __init__(self):
        super().__init__()
        self.setWindowTitle("My App")
        widget = QCheckBox("This is a checkbox")
        widget.setCheckState(Qt.CheckState.Checked)
        # For tristate: widget.setCheckState(Qt.PartiallyChecked)
        # Or: widget.setTristate(True)
        widget.stateChanged.connect(self.show_state)
        self.setCentralWidget(widget)

    def show_state(self, s):
        print(Qt.CheckState(s) == Qt.CheckState.Checked)
        print(s)

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

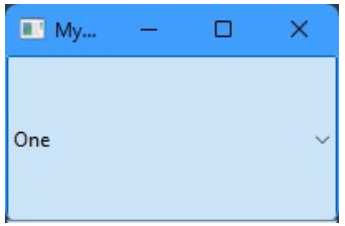


setCheckState()

Flag	Behavior
`Qt.CheckState.Checked`	Item is checked
`Qt.CheckState.Unchecked`	Item is unchecked
`Qt.CheckState.PartiallyChecked`	Item is partially checked

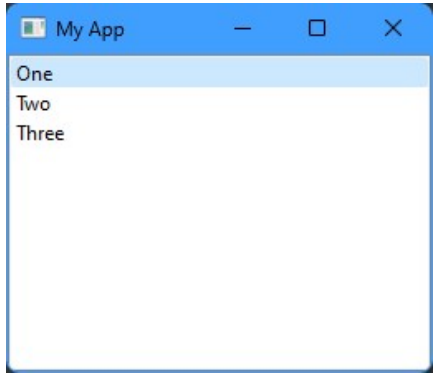
QComboBox: a drop-down list, closed by default with an arrow to open it

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

class MainWindow(QMainWindow):
    def __init__(self):
        super().__init__()
        self.setWindowTitle("My App")
        widget = QComboBox()
        widget.addItems(["One", "Two", "Three"])
        widget.currentIndexChanged.connect(self.index_changed)
        widget.currentTextChanged.connect(self.text_changed)
        self.setCentralWidget(widget)
    def index_changed(self, i): # i is an int
        print(i)
    def text_changed(self, s): # s is a str
        print(s)

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



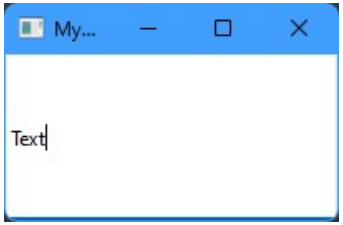
QListWidget: similar to QComboBox, except options are presented as a scrollable list

```
import sys
from PyQt6.QtWidgets import QApplication, QListWidget, QMainWindow

class MainWindow(QMainWindow):
    def __init__(self):
        super().__init__()
        self.setWindowTitle("My App")
        widget = QListWidget()
        widget.addItems(["One", "Two", "Three"])
        widget.currentItemChanged.connect(self.index_changed)
        widget.currentTextChanged.connect(self.text_changed)
        self.setCentralWidget(widget)
    def index_changed(self, i): # Not an index, i is a QListItem
        print(i.text())
    def text_changed(self, s): # s is a str
        print(s)

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



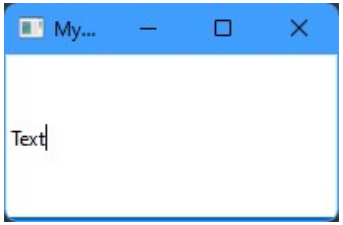
QLineEdit: single-line input text editing box

```
import sys
from PyQt6.QtWidgets import QApplication, QLineEdit,
QMainWindow


class MainWindow(QMainWindow):
    def __init__(self):
        super().__init__()
        self.setWindowTitle("My App")
        widget = QLineEdit()
        widget.setMaxLength(10)
        widget.setPlaceholderText("Enter your text")
        # widget.setReadOnly(True) # uncomment this to make
readonly
        widget.returnPressed.connect(self.return_pressed)
        widget.selectionChanged.connect(self.selection_changed)
        widget.textChanged.connect(self.text_changed)
        widget.textEdited.connect(self.text_edited)
        self.setCentralWidget(widget)
```



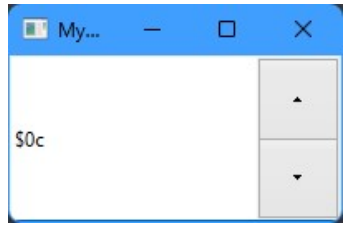
QLineEdit: single-line input text editing box

```
def return_pressed(self):
        print("Return pressed!")
        self.centralWidget().setText("BOOM!")
    def selection_changed(self):
        print("Selection changed")
        print(self.centralWidget().selectedText())
    def text_changed(self, s):
        print("Text changed...")
        print(s)
    def text_edited(self, s):
        print("Text edited...")
        print(s)

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



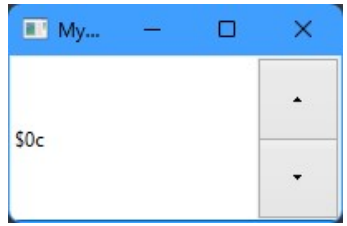
QSpinBox: a small numerical input box with arrows to increase and decrease the value

```
import sys
from PyQt6.QtWidgets import QApplication, QMainWindow, QSpinBox

class MainWindow(QMainWindow):
    def __init__(self):
        super().__init__()
        self.setWindowTitle("My App")
        widget = QSpinBox()
        # Or: widget = QDoubleSpinBox()
        widget.setMinimum(-10)
        widget.setMaximum(3)
        # Or: widget.setRange(-10,3)
        widget.setPrefix("$")
        widget.setSuffix("c")
        widget.setSingleStep(3) # Or e.g. 0.5 for QDoubleSpinBox
        widget.valueChanged.connect(self.value_changed)
        widget.textChanged.connect(self.value_changed_str)
        self.setCentralWidget(widget)
```



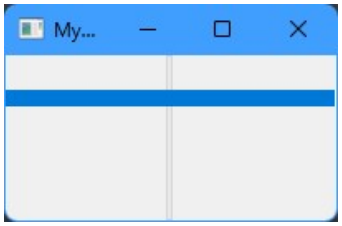
QSpinBox: a small numerical input box with arrows to increase and decrease the value

```
def value_changed(self, i):
        print(i)
    def value_changed_str(self, s):
        print(s)


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



QSlider: a slide-bar widget

```
import sys
from PyQt6.QtWidgets import QApplication, QMainWindow, QSlider

class MainWindow(QMainWindow):
    def __init__(self):
        super().__init__()
        self.setWindowTitle("My App")
        widget = QSlider()
        widget.setMinimum(-10)
        widget.setMaximum(3)
        # Or: widget.setRange(-10,3)
        widget.setSingleStep(3)
        widget.valueChanged.connect(self.value_changed)
        widget.sliderMoved.connect(self.slider_position)
        widget.sliderPressed.connect(self.slider_pressed)
        widget.sliderReleased.connect(self.slider_released)
        self.setCentralWidget(widget)
```



QSlider: a slide-bar widget

```
def value_changed(self, i):
        print(i)
    def slider_position(self, p):
        print("position", p)
    def slider_pressed(self):
        print("Pressed!")
    def slider_released(self):
        print("Released")

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