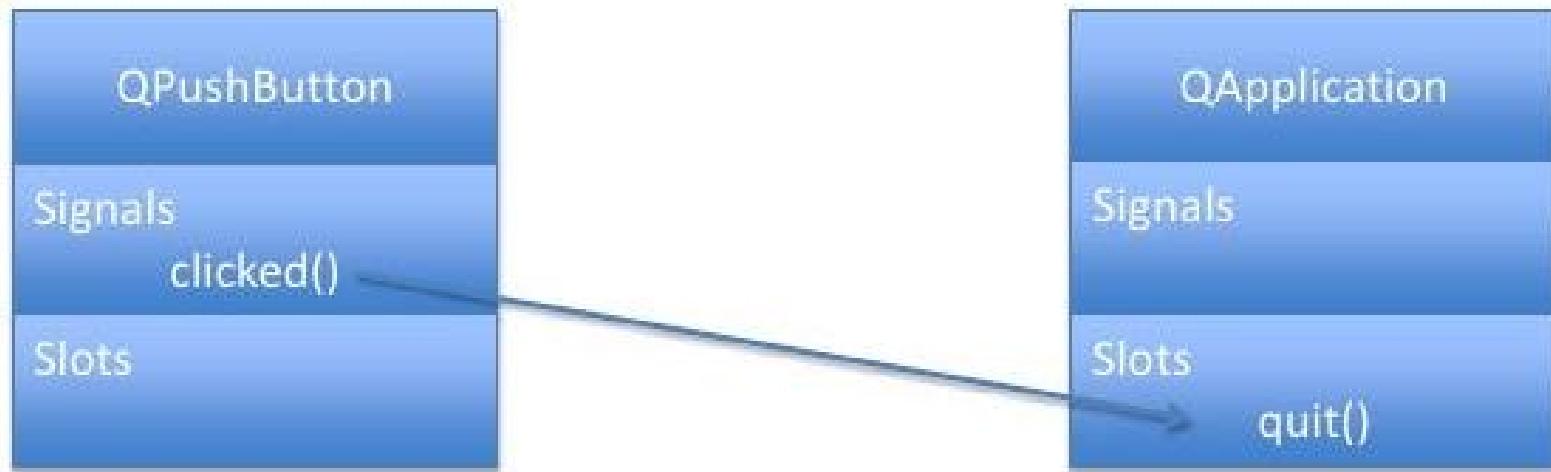


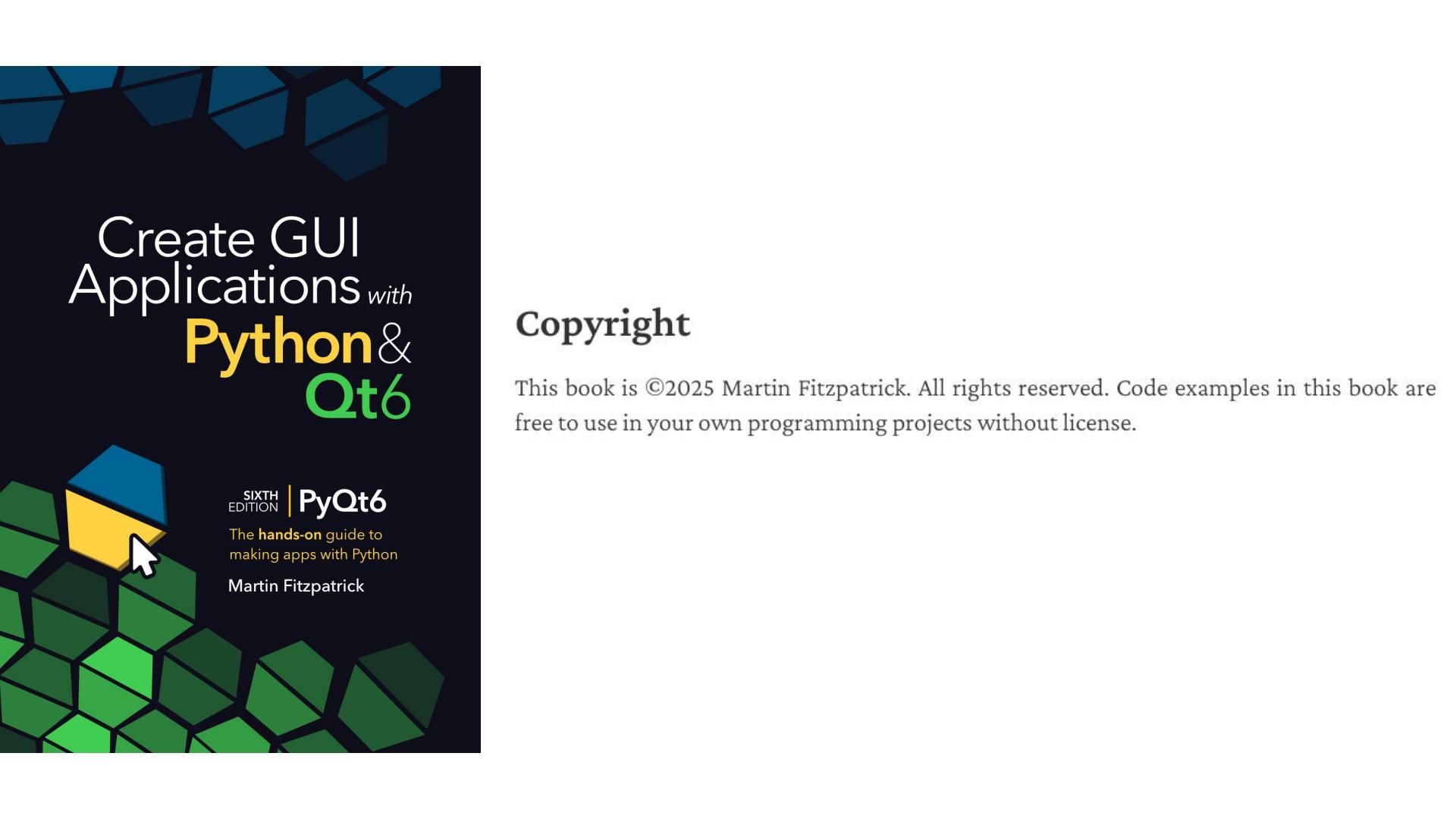
pyQt6



Signals & Slots

Signals are *notifications* emitted by widgets when something happens.





Create GUI Applications *with* **Python &** **Qt6**

SIXTH EDITION | **PyQt6**

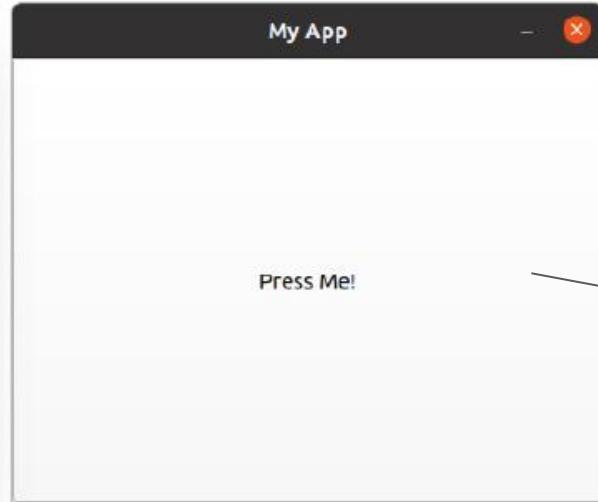
The **hands-on** guide to
making apps with Python

Martin Fitzpatrick

Copyright

This book is ©2025 Martin Fitzpatrick. All rights reserved. Code examples in this book are free to use in your own programming projects without license.

Sample



```
import sys

from PyQt6.QtCore import QSize
from PyQt6.QtWidgets import QApplication, QMainWindow, QPushButton

class MainWindow(QMainWindow):
    def __init__(self):
        super().__init__() ②

        self.setWindowTitle("My App")

        button = QPushButton("Press Me!")
        button.clicked.connect(self.the_button_was_clicked)

        # Set the central widget of the Window.
        self.setCentralWidget(button)

    def the_button_was_clicked(self):
        print("Clicked!")

app = QApplication(sys.argv)

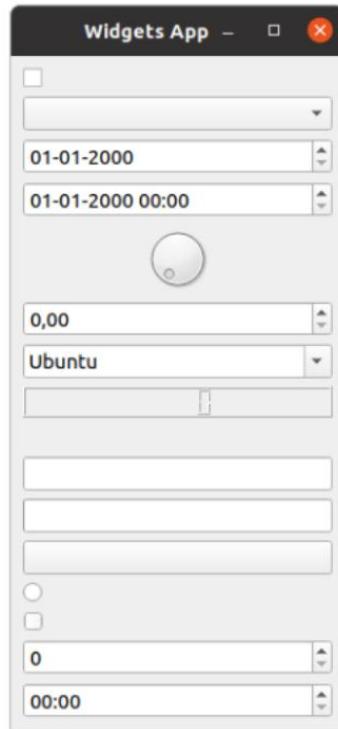
window = MainWindow()
window.show()

app.exec()
```

The code above is a Python script using the PyQt6 library to create a simple application window. It defines a class `MainWindow` that inherits from `QMainWindow`. In its constructor, it sets the window title to "My App", creates a button with the text "Press Me!", and connects the button's `clicked` signal to a slot named `the_button_was_clicked`. This slot prints the string "Clicked!" to the console. Finally, it initializes an `QApplication` object, creates an instance of `MainWindow`, shows the window, and enters the application's event loop with `exec`.

Clicked!
Clicked!
Clicked!

Widgets



Widget	What it does
QCheckbox	A checkbox
QComboBox	A dropdown list box
QDateEdit	For editing dates
QDateTimeEdit	For editing dates and datetimes
QDial	Rotatable dial
QDoubleSpinBox	A number spinner for floats
QFontComboBox	A list of fonts
QLCDNumber	A quite ugly LCD display
QLabel	Just a label, not interactive
QLineEdit	Enter a line of text
QProgressBar	A progress bar
QPushButton	A button
QRadioButton	A group with only one active choice
QSlider	A slider
QSpinBox	An integer spinner
QTimeEdit	For editing times

Layouts

Layout	Behavior
QHBoxLayout	Linear horizontal layout
QVBoxLayout	Linear vertical layout
QGridLayout	An (x&y) indexable grid
QStackedLayout	Stacked (z) in front of one another
QFormLayout	Linear 2-column layout for labels & fields

Nesting layouts

For more complex layouts you can nest layouts inside one another using `.addLayout` on a layout. Below we add a `QVBoxLayout` into the main `QHBoxLayout`. If we add some widgets to the `QVBoxLayout`, they'll be arranged vertically in the first slot of the parent layout.

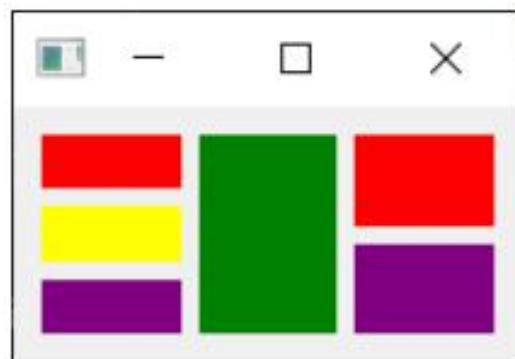


Figure 26. Nested `QHBoxLayout` and `QVBoxLayout` layouts.

```
import sys

from layout_colorwidget import Color

from PyQt6.QtWidgets import (
    QApplication,
    QHBoxLayout,
    QMainWindow,
    QVBoxLayout,
    QWidget,
)

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

        self.setWindowTitle("My App")

        layout1 = QHBoxLayout()
        layout2 = QVBoxLayout()
        layout3 = QVBoxLayout()

        layout2.addWidget(Color("red"))
        layout2.addWidget(Color("yellow"))
        layout2.addWidget(Color("purple"))

        layout1.addLayout(layout2)
        layout1.addWidget(Color("green"))

        layout3.addWidget(Color("red"))
        layout3.addWidget(Color("purple"))

        layout1.addLayout(layout3)

        widget = QWidget()
        widget.setLayout(layout1)
        self.setCentralWidget(widget)

app = QApplication(sys.argv)

window = MainWindow()
window.show()

app.exec()
```

QGridLayout is a 2-dimensional layout, allowing you to lay widgets out horizontally and vertically using grid positions. Importantly, widgets in the same rows and columns will have their edges aligned. If widgets are different sizes, empty space will be added to the layout to keep this alignment.

0,0	0,1	0,2	0,3
1,0	1,1	1,2	1,3
2,0	2,1	2,2	2,3
3,0	3,1	3,2	3,3

```
import sys  
  
from layout_colorwidget import Color  
  
from PyQt6.QtWidgets import (  
    QApplication,  
    QGridLayout,  
    QMainWindow,  
    QWidget,  
)
```

```
class MainWindow(QMainWindow):  
    def __init__(self):  
        super().__init__()  
  
        self.setWindowTitle("My App")  
  
        layout = QGridLayout()  
  
        layout.addWidget(Color("red"), 0, 0)  
        layout.addWidget(Color("green"), 1, 0)  
        layout.addWidget(Color("blue"), 1, 1)  
        layout.addWidget(Color("purple"), 2, 1)
```

```
widget = QWidget()  
widget.setLayout(layout)  
self.setCentralWidget(widget)  
  
app = QApplication(sys.argv)  
  
window = MainWindow()  
window.show()  
  
app.exec()
```

