COMPSYS302: PyQt5

1. Basics

Ho Seok AHN

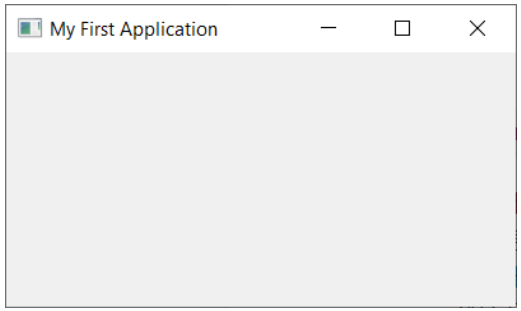
hs.ahn@auckland.ac.nz

THE UNIVERSITY OF AUCKLAND
FACULTY OF ENGINEERING

1.1 Windows

THE UNIVERSITY OF AUCKLAND
FACULTY OF ENGINEERING

Opening Windows



- PyQt5 support the windows as a basic of GUI.
- You can adjust the size or control windows.

- Call basic modules

  - Ex) 1_01_Windows.py

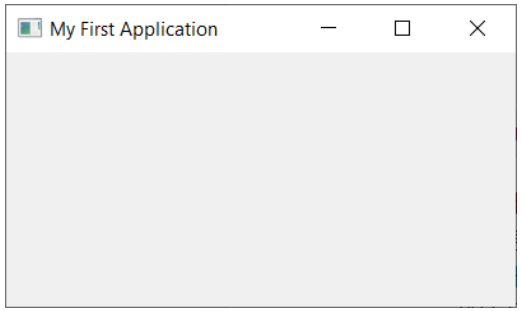
```
import sys
from PyQt5.QtWidgets import QApplication, QWidget
```

➔ Almost basic UI elements are included in "PyQt5.QtWidgets" module.

1.1 Windows

THE UNIVERSITY OF AUCKLAND
FACULTY OF ENGINEERING

Opening Windows



- Ex) 1_01_Windows.py

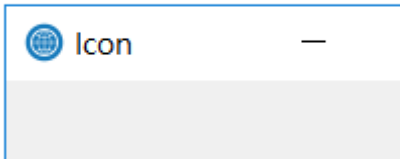
```
self.setWindowTitle('My First Application')
self.move(300, 300)
self.resize(400, 200)
self.show()
```

- ➔ "self" means the MyApp object.
- ➔ **setWindowTitle**(): Set title of window
- ➔ **move**(x, y): position of the window (unit: pixel)
- ➔ **resize**(width, height): size of the window (unit: pixel)
- ➔ **show**(): make it visible

1.2 Icon

THE UNIVERSITY OF AUCKLAND
FACULTY OF ENGINEERING

☐ Add Icon (QIcon)



- Can add an icon on the window.
- You should have the image on your PC.
- Remember the directory where your image is.

  e.g. 'web.png' under 'D:\ workspace\ COMPSYS302_PyQt5'

  - Ex) 1_02_Icon.py

```
self.setWindowIcon(QIcon('D:\ workspace\ COMPSYS302_PyQt5\ web.png'))
```

➔ Almost basic UI elements are included in "PyQt5.QtWidgets" module.

```
self.setGeometry(x, y, 300, 200)
```

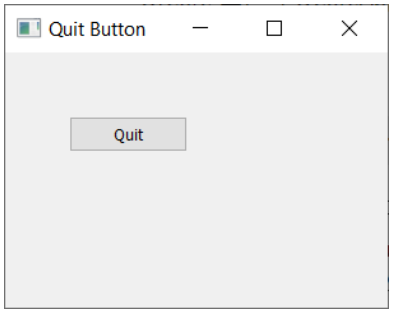
➔ **setGeometry**(x, y, width, height): position and size of the window (unit: pixel)

➔ It is a mixture of **move**(x, y) and **resize**(width, height)

1.3 Button

THE UNIVERSITY OF AUCKLAND
FACULTY OF ENGINEERING

Add Button (QPushButton)



- Can add buttons on the window.
- Let's add one button to quit the app using the "Quit" button.
  - Ex) 1_03_Quit.py

btn = **QPushButton**('text', self)

➔ Add a button, which is an instance of QPushButton class

btn.**clicked**.connect(QCoreApplication.instance().quit)

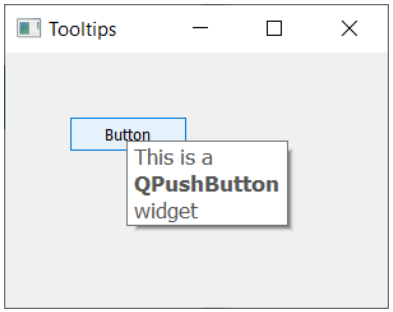
➔ Add an event for button.

➔ We will learn more about event later.

1.4 ToolTip

THE UNIVERSITY OF AUCKLAND
FACULTY OF ENGINEERING

Add ToolTip (QToolTip)



- Can add ToolTip for an object on the window.
- It shows a text when the mouse curser is on the object.

  - Ex) 1_04_ToolTip.py

```
QToolTip.setFont(QFont('Sansserif', 10))
self.setToolTip('This is a <b>QWidget</b> widget')
```

➔ Set the font for ToolTip

```
btn.setToolTip('This is a <b>QPushButton</b> widget')
```

➔ Use the defined ToolTip, and add it to a button

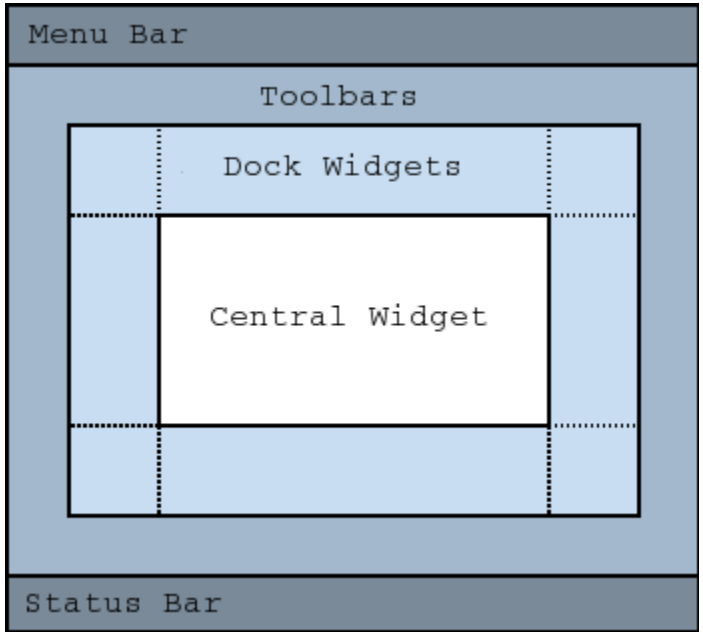
```
btn.resize(btn.sizeHint())
```

➔ Set appropriate size of button

1.5 Status Bar

Add Status Bar (QStatusBar)

- Main window has its own layout for QMenuBar, QToolBar, QDockWidget, QStatusBar
- Any type of widgets can be placed on the 'Central Widget' area.

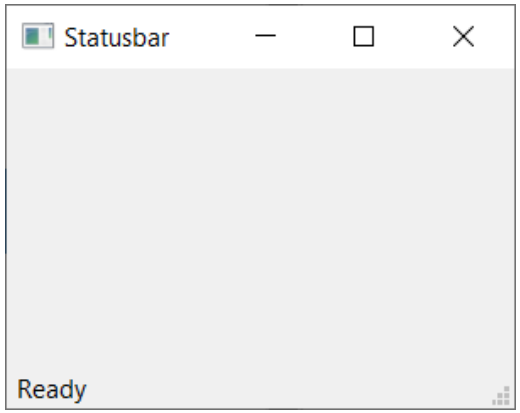


1.5 Status Bar

THE UNIVERSITY OF AUCKLAND
FACULTY OF ENGINEERING

Add Status Bar

- Let's make a status bar.



- Ex) 1_05_StatusBar.py

```
self.statusBar().showMessage('Ready')
```

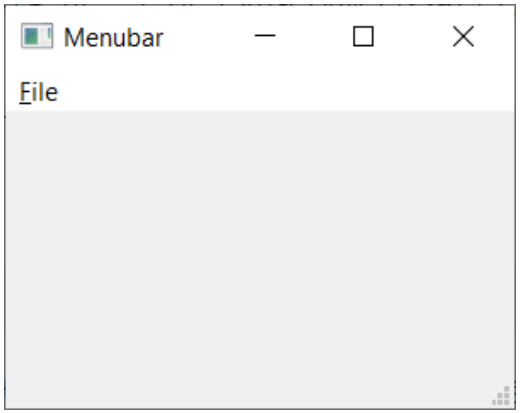
➔ Use statusBar() method of QMainWindow class

➔ showMessage(): show the message for the status bar

1.6 Menu Bar

THE UNIVERSITY OF AUCKLAND
FACULTY OF ENGINEERING

Add Menu Bar (QMenuBar)



- Can add menu bar on the window.
- Can add icon as well.

  - Ex) 1_06_MenuBar.py

```
exitAction = QAction(QIcon('D:\ workspace\ COMPSYS302_PyQt5\ exit.png'), 'Exit', self)
exitAction.setShortcut('Ctrl+Q')
exitAction.setStatusTip('Exit application’)
```

➔ Make a menu with icon, shortcut and message

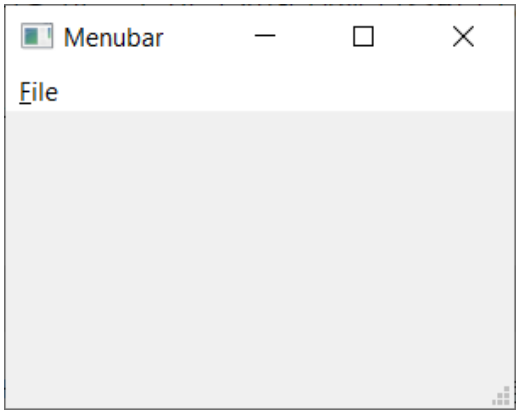
```
exitAction.triggered.connect(qApp.quit)
```

➔ Quit app with menu bar

1.6 Menu Bar

THE UNIVERSITY OF AUCKLAND
FACULTY OF ENGINEERING

Add Menu Bar (QMenuBar)



- Can add menu bar on the window.
  - Ex) 1_06_MenuBar.py

```
menubar = self.menuBar()
fileMenu = menubar.addMenu('&File')
fileMenu.addAction(exitAction)
```

➔ Add menubar

➔ &File: simpler version of **setShortcut()** ➔ 'Alt+F'

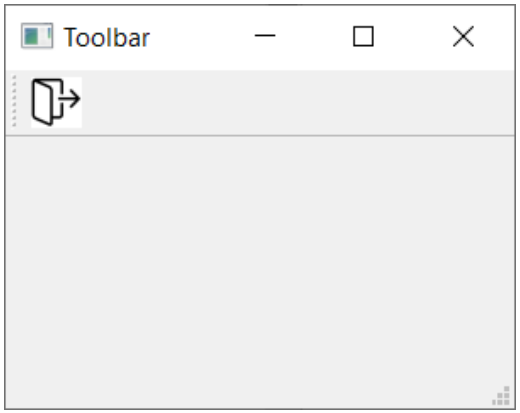
➔ Add 'exitAction' to file menu

1.7 Tool Bar

THE UNIVERSITY OF AUCKLAND
FACULTY OF ENGINEERING

Add Tool Bar (QToolBar)

- Can add tool bar for easy usage on the window.



- Ex) 1_07_ToolBar.py
  - Same function with the 1_06_MenuBar.py

```
self.toolbar = self.addToolBar('Exit')
self.toolbar.addAction(exitAction)
```

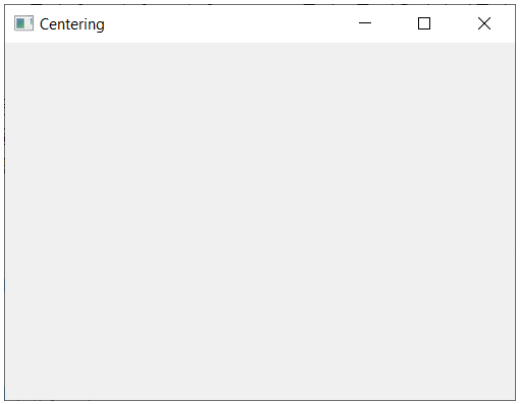
➔ Add toolbar

➔ Add 'exitAction' to file menu

1.8 Position

THE UNIVERSITY OF AUCKLAND
FACULTY OF ENGINEERING

Adjust position of window



- Let's show the window at the centre of the monitor.
  - Ex) 1_08_Centering.py

    qr = self.**frameGeometry**()

    ➔ Get window information

    cp = QDesktopWidget().availableGeometry().**center**()

    ➔ Get the monitor information; get the center position of the monitor

    qr.**moveCenter**(cp)

    ➔ Set the centre position of the monitor to the variable 'qr'

    self.**move**(qr.topLeft())

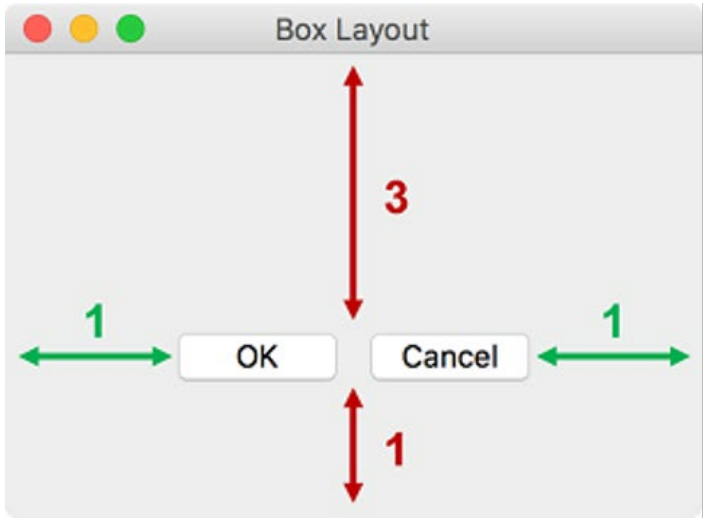
    ➔ Move the window to the centre position of the monitor

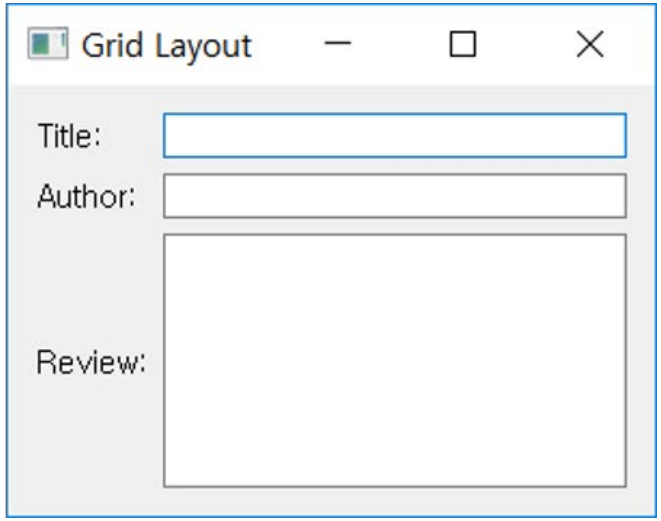
1.9 Layout

THE UNIVERSITY OF AUCKLAND
FACULTY OF ENGINEERING

Three types of layouts

- Layout is away to put widgets on the application window.
- Three types of layouts
  - Absolute positioning
  - Box layout
  - Grid layout

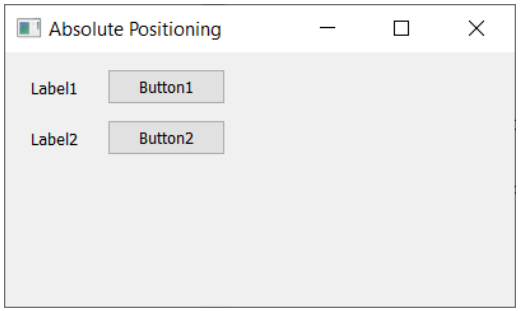




1.9 Layout

THE UNIVERSITY OF AUCKLAND
FACULTY OF ENGINEERING

Absolute positioning



- It set all the details separately.
- Widgets are not changed at all.
- It looks different from different applications.
- You may need adjustment even it you change its fonts.

  - Ex) 1_09_AbsolutePositioning.py

```
label1 = QLabel('Label1', self)
label1.move(x, y)
```

➔ Make a label1, then position it to (x, y)

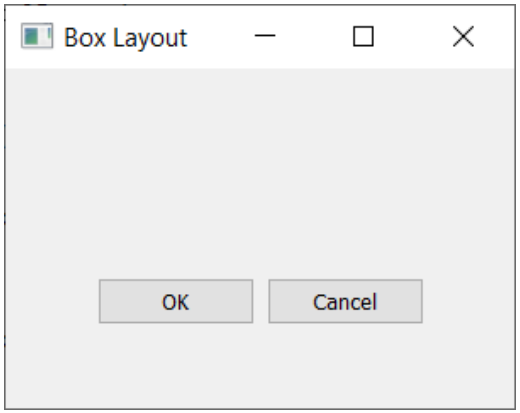
```
btn1 = QPushButton('Button1', self)
btn1.move(x, y)
```

➔ Make a button1, then position it to (x, y)

1.9 Layout

THE UNIVERSITY OF AUCKLAND
FACULTY OF ENGINEERING

Box Layout (QBoxLayout)



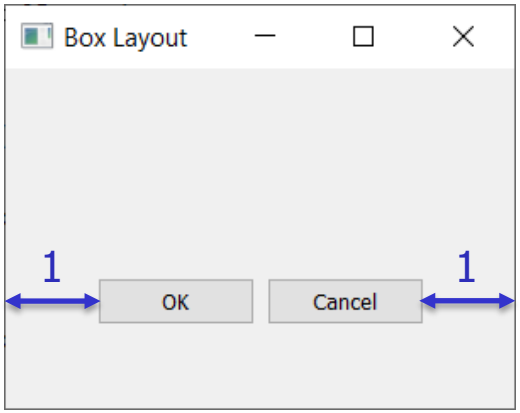
- It aligns widgets horizontally and vertically.
- It dynamically adjust the positions of widgets depends on the window size.
- Use **QHBoxLayout**, **QVBoxLayout** for adjusting the positions of widgets.

- Ex) 1_10_BoxLayout.py

okButton = **QPushButton**('text')

➔ Make a button with 'text'

Box Layout (QBoxLayout)



- Ex) 1_10_BoxLayout.py

```
hbox = QHBoxLayout()
hbox.addStretch(1)
hbox.addWidget(okButton)
hbox.addWidget(cancelButton)
hbox.addStretch(1)
```

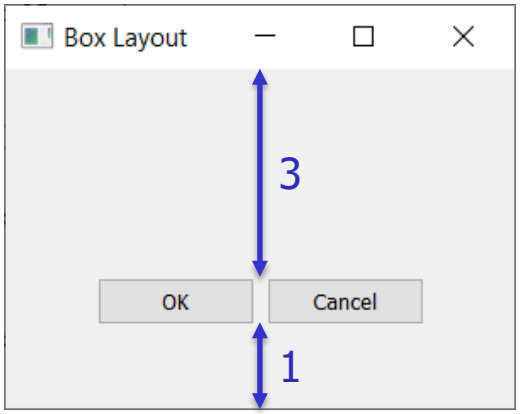
➔ **QHBoxLayout**(): Make a horizontal box.
➔ **addStretch**(n): Add a space with the stretch factor as n
➔ **addWidget**(widget): Add a widget

➔ Here, we have two **addStretch**(1) in the line 2 and the line 5:
It makes both spaces as same size when the size of window is changed.

1.9 Layout

THE UNIVERSITY OF AUCKLAND
FACULTY OF ENGINEERING

Box Layout (QBoxLayout)



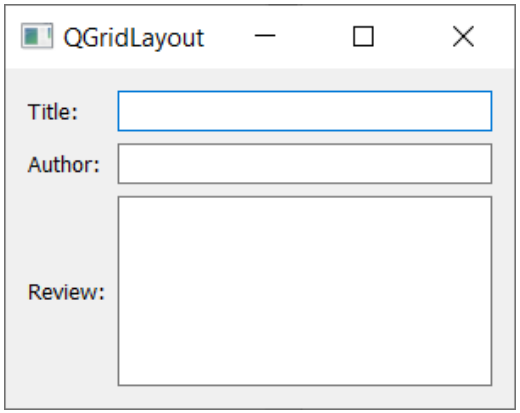
- Ex) 1_10_BoxLayout.py

```
vbox = QVBoxLayout()
vbox.addStretch(3)
vbox.addLayout(hbox)
vbox.addStretch(1)
```

➔ **QVBoxLayout**(): Make a vertical box.
➔ **addStretch**(n): Add a space with the stretch factor as n
➔ **addLayout**(hbox): Add the "hbox"

➔ Here, we have **addStretch**(3) in the line 2 and another **addStretch**(1) in the line 4: It makes the size of top and bottom spaces are always 3:1 when the size of window is changed.
➔ But it does not mean changing the size of widgets.

Grid Layout (QGridLayout)



- It separates the space as row and column.
- Use **QGridLayout** for setting grids.
  - Ex) 1_11_GridLayout.py

```
grid = QGridLayout()
self.setLayout(grid)
```

➔ Make a grid and set it as the layout of the window.

```
grid.addWidget(widget, row, column)
```

➔ Add a widget to specific row and column.

```
grid.addWidget(QLineEdit(), row, column)
grid.addWidget(QTextEdit(), row, column)
```

➔ **QLineEdit**(): widget with one line

➔ **QTextEdit**(): widget with multi-line