Python PyQt6 Class 2

**QRadioButton 單選按鈕**

建立 PyQt6 視窗物件後，**透過QtWidgets.QRadioButton(widget)方法，就能在指定的元件中建立單選按鈕**，加入兩個 QRadioButton 按鈕 ，並使用 setText() 方法加入文字。注意，**放在同樣元件裡的 QRadioButton 視為同一個群組，會套用「單選」的規則**，兩個Button都放在 Form 裡，所以只能擇一選擇。

rb\_a = QtWidgets.QRadioButton(Form) # 單選按鈕 A

rb\_a.setGeometry(30, 30, 100, 20)

rb\_a.setText('A')

rb\_b = QtWidgets.QRadioButton(Form) # 單選按鈕 B

rb\_b.setGeometry(30, 60, 100, 20)

rb\_b.setText('B')

如果有「多組」QRadioButton，**則可以使用QtWidgets.QButtonGroup(widget)方法建立按鈕群組，然後將歸類為同一組的 QRadioButton 加入同一個 QButtonGroup**，就能分別進行單選的動作。

rb\_c = QtWidgets.QRadioButton(Form) # 單選按鈕 C

rb\_c.setGeometry(150, 30, 100, 20)

rb\_c.setText('C')

rb\_d = QtWidgets.QRadioButton(Form) # 單選按鈕 D

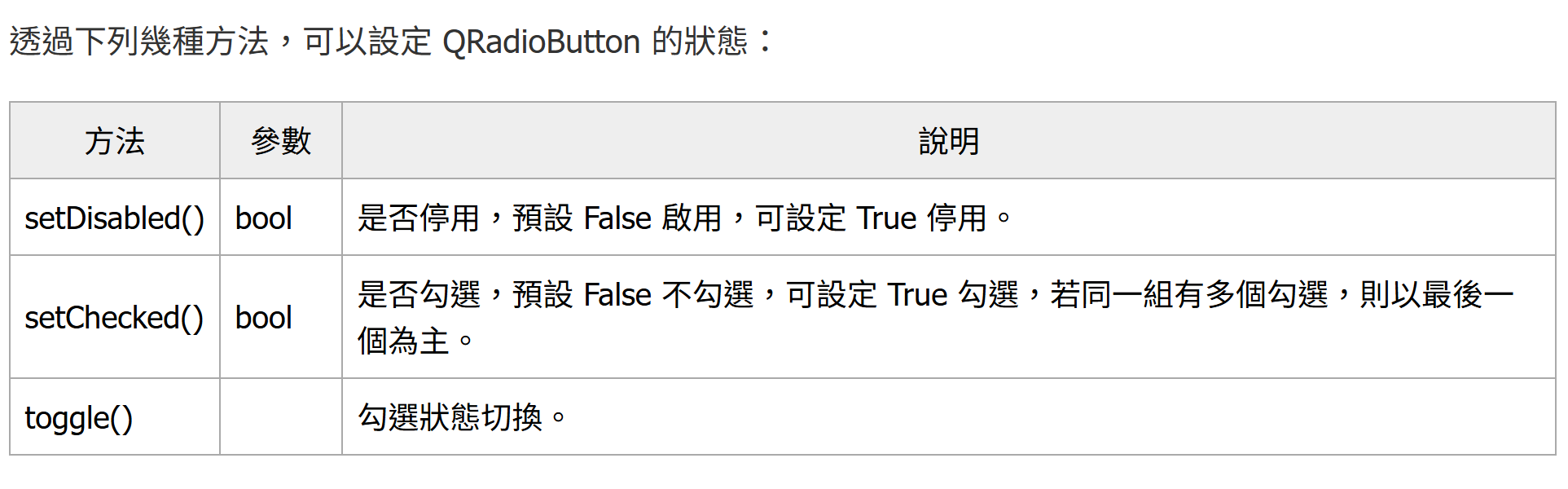
rb\_d.setGeometry(150, 60, 100, 20)

rb\_d.setText('D')

group2 = QtWidgets.QButtonGroup(Form) # 按鈕群組

group2.addButton(rb\_c) # 加入單選按鈕 C

group2.addButton(rb\_d) # 加入單選按鈕 D



rb\_b = QtWidgets.QRadioButton(Form)

rb\_b.setGeometry(30, 60, 100, 20)

rb\_b.setText('B')

rb\_b.setDisabled(True) # 停用

rb\_c = QtWidgets.QRadioButton(Form)

rb\_c.setGeometry(30, 90, 100, 20)

rb\_c.setText('C')

rb\_c.setChecked(True) # 預先勾選

rb\_a = QtWidgets.QRadioButton(Form)

rb\_a.setGeometry(30, 30, 100, 20)

rb\_a.setText('A')

套用CSS

# 設定按鈕 A 的樣式

rb\_a.setStyleSheet('''

QRadioButton {

color: #00f;

}

QRadioButton:hover {

color:#f00;

}

''')

**QRadioButton 點擊事件**

如果要偵測勾選了哪個 QRadioButton，有兩種常用的方法，第一種是透過 QButtonGroup() 將 QRadioButton 包裝成同一個群組，使用 addButton() 添加按鈕時，**可設定第二個按鈕的 ID 參數，設定後使用buttonClicked.connect(fn)方法，就能偵測是否勾選按鈕，並能夠過函式，執行 checkedId() 取得勾選按鈕的 ID**，下方的程式碼執行後，會在勾選不同按鈕時，透過 QLabel 顯示對應的勾選按鈕的 ID。

def show():

label.setText(str(group.checkedId())) # 設定 label 文字為按鈕群組中勾選按鈕的 ID

group = QtWidgets.QButtonGroup(Form)

group.addButton(rb\_a, 1) # 添加 QRadioButton A，ID 設定為 1

group.addButton(rb\_b, 2) # 添加 QRadioButton B，ID 設定為 2

group.buttonClicked.connect(show) # 綁定點擊事件

label = QtWidgets.QLabel(Form)

label.setGeometry(30, 30, 100, 20)

**複選按鈕 QCheckBox**

cb\_a = QtWidgets.QCheckBox(Form) # 複選按鈕 A

cb\_a.move(30, 60)

cb\_a.setText('A')

cb\_b = QtWidgets.QCheckBox(Form) # 複選按鈕 B

cb\_b.move(80, 60)

cb\_b.setText('B')

cb\_c = QtWidgets.QCheckBox(Form) # 複選按鈕 C

cb\_c.setGeometry(130, 60, 50, 20)

cb\_c.setText('C')

cb\_d = QtWidgets.QCheckBox(Form) # 複選按鈕 D

cb\_d.setGeometry(180, 60, 50, 20)

cb\_d.setText('D')

# 設定 QCheckBox

style = '''

QCheckBox {

color: #00f;

}

QCheckBox:hover {

color: #f00;

}

QCheckBox:checked {

color: #fff;

background: #000;

}

'''

套用CSS

cb\_a = QtWidgets.QCheckBox(Form)

cb\_a.move(30, 60)

cb\_a.setText('A')

cb\_a.setStyleSheet(style) # 套用 style

cb\_b = QtWidgets.QCheckBox(Form)

cb\_b.move(80, 60)

cb\_b.setText('B')

cb\_b.setStyleSheet(style) # 套用 style

cb\_c = QtWidgets.QCheckBox(Form)

cb\_c.move(130, 60)

cb\_c.setText('C')

cb\_c.setStyleSheet(style) # 套用 style

**QCheckBox 點擊事件**

如果要偵測勾選了哪個 QCheckBox，可以**使用 clicked() 的方法，將函式與各個按鈕綁定**，接著就能透過 text() 取得按鈕文字，透過 isChecked() 取得按鈕勾選狀態，下方的程式碼執行後，勾選 QCheckBox 時，就會將勾選的按鈕文字組合，透過 QLabel 輸出顯示。

arr = ['']\*3 # 先新增一個串列放入文字

def show(cb, i):

global a

if cb.isChecked():

arr[i] = cb.text() # 如果該按鈕是勾選狀態，在串列的指定位置放入文字

else:

arr[i] = '' # 如果該按鈕是勾選狀態，在串列的指定位置放入空字串

output = ''.join(arr) # 組合串列內容為文字

label.setText(output) # label 顯示文字

label = QtWidgets.QLabel(Form)

label.setGeometry(30, 30, 100, 30)

cb\_a = QtWidgets.QCheckBox(Form)

cb\_a.move(30, 60)

cb\_a.setText('A')

cb\_a.clicked.connect(lambda:show(cb\_a, 0)) # 點擊按鈕時，回傳兩個參數給 show 函式

cb\_b = QtWidgets.QCheckBox(Form)

cb\_b.move(80, 60)

cb\_b.setText('B')

cb\_b.clicked.connect(lambda:show(cb\_b, 1)) # 點擊按鈕時，回傳兩個參數給 show 函式

cb\_c = QtWidgets.QCheckBox(Form)

cb\_c.move(130, 60)

cb\_c.setText('C')

cb\_c.clicked.connect(lambda:show(cb\_c, 2)) # 點擊按鈕時，回傳兩個參數給 show 函式

**QGraphicsView 顯示圖片**

建立 PyQt6 視窗物件後，**透過QtWidgets.QGraphicsView(widget)方法，就能在指定的元件中建立顯示圖形元件**，QGraphicsView 建立後，需**再使用QtWidgets.QGraphicsScene()建立場景元件，再透過QtGui.QPixmap()於場景中加入圖片，最後將場景加入 QGraphicsView 就可以顯示圖片**，如果場景大小超過顯示區域，會自動出現捲軸。

grview = QtWidgets.QGraphicsView(Form) # 加入 QGraphicsView

grview.setGeometry(20, 20, 260, 200) # 設定 QGraphicsView 位置與大小

scene = QtWidgets.QGraphicsScene() # 加入 QGraphicsScene

scene.setSceneRect(0, 0, 300, 400) # 設定 QGraphicsScene 位置與大小

img = QtGui.QPixmap('mona.jpg') # 加入圖片

scene.addPixmap(img) # 將圖片加入 scene

grview.setScene(scene) # 設定 QGraphicsView 的場景為 scene

**改變圖片尺寸**

使用 QtGui.QPixmap() 建立圖片後，就能透過scaled(w, h)方法調整圖片大小，下方的程式碼執行後，會顯示縮小後的圖片。

img = img.scaled(120,160) # 調整圖片大小為 120x160

位置設定

grview = QtWidgets.QGraphicsView(Form)

gw = 260

gh = 200

grview.setGeometry(20, 20, gw, gh) # QGraphicsView 的長寬改成變數

scene = QtWidgets.QGraphicsScene()

img = QtGui.QPixmap('mona.jpg')

img\_w = 120 # 顯示圖片的寬度

img\_h = 160 # 顯示圖片的高度

img = img.scaled(img\_w, img\_h)

x = 20 # 左上角 x 座標

y = 20 # 左上角 y 座標

dx = int((gw - img\_w) / 2) - x # 修正公式

dy = int((gh - img\_h) / 2) - y

scene.setSceneRect(dx, dy, img\_w, img\_h)

scene.addPixmap(img)

grview.setScene(scene)

**顯示多張圖片**

如果要加入多張圖片，就要使用 QItem 的做法，下方的程式碼執行後，會在場景裡放入兩個圖片尺寸不同的 QItem。

grview = QtWidgets.QGraphicsView(MainWindow) # 加入 QGraphicsView

grview.setGeometry(0, 0, 300, 300) # 設定 QGraphicsView 位置與大小

scene = QtWidgets.QGraphicsScene() # 加入 QGraphicsScene

scene.setSceneRect(0, 0, 200, 200) # 設定 QGraphicsScene 位置與大小

img = QtGui.QPixmap('mona.jpg') # 建立圖片

img1 = img.scaled(200,50) # 建立不同尺寸圖片

qitem1 = QtWidgets.QGraphicsPixmapItem(img1) # 設定 QItem，內容是 img1

img2 = img.scaled(100,150) # 建立不同尺寸圖片

qitem2 = QtWidgets.QGraphicsPixmapItem(img2) # 設定 QItem，內容是 img2

scene.addItem(qitem1) # 場景中加入 QItem

scene.addItem(qitem2) # 場景中加入 QItem

grview.setScene(scene) # 設定 QGraphicsView 的場景為 scene