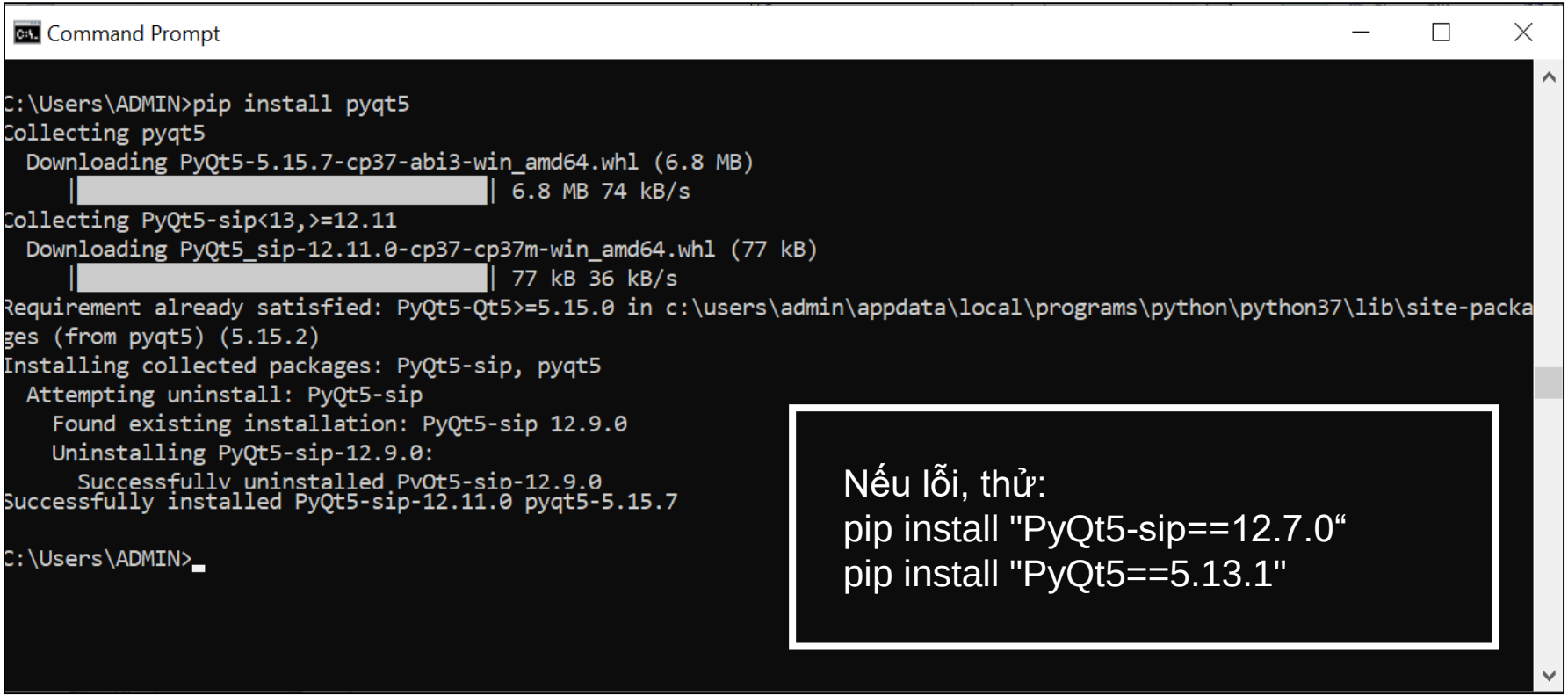
PHỤ LỤC
Sử dụng QtDesigner và PyQt5

PHẦN 1
Sử dụng PyQt5

- **Cài đặt PyQt5**: tại dấu nhắc lệnh thực thi: pip install PyQt5



2. Sơ lược PyQt5

- PyQt là Python interface của Qt, dùng để phát triển các ứng dụng trên Desktop, hệ thống nhúng, mobile…
- PyQt kết hợp ngôn ngữ Python và thư viện Qt
- PyQt bao gồm:
  - Các thành phần giao diện điều khiển (widgets, graphical control elements).
  - Các module chứa các classes và functions hỗ trợ thiết kế các giao diện người dùng đồ họa (GUI-Grapic User Interface).

3. Các class của PyQt5

- QtCore: phần lõi dùng để làm việc với thời gian, file và thư mục, các loại dữ liệu, streams, URLs, mime type, threads hoặc processes.
- QtGui: dùng cho việc lập trình giao diện, event handling, 2D graphics, fonts, text…
- QtWidgets: gồm các widget như: button, hộp thoại… để tạo nên giao diện người dùng.
- QtMultimedia: sử dụng âm thanh, hình ảnh, camera…
- QtBluetooth: dùng cho việc tìm kiếm và kết nối với các thiết bị giao tiếp.
- QtNetwork: dùng cho việc lập trình mạng, lập trình TCP/IP và UDP client, server...
- QtPositioning: hỗ trợ định vị.
- Enginio: giúp các client truy cập các Cloud Services của Qt.

3. Các class của PyQt5

- QtWebSockets: cung cấp các công cụ cho WebSocket protocol.
- QtWebKit: dùng cho làm việc với các trình duyệt Web, dựa trên thư viện WebKit2.
- QtWebKitWidgets: các widget cho WebKit.
- QtXml: dùng cho làm việc với XML file.
- QtSvg: dùng cho hiển thị các thành phần của SVG file.
- QtSql: dùng cho việc làm việc với dữ liệu.
- QtTest: cung cấp các công cụ cho phép test các đơn vị của ứng dụng với PyQt5.

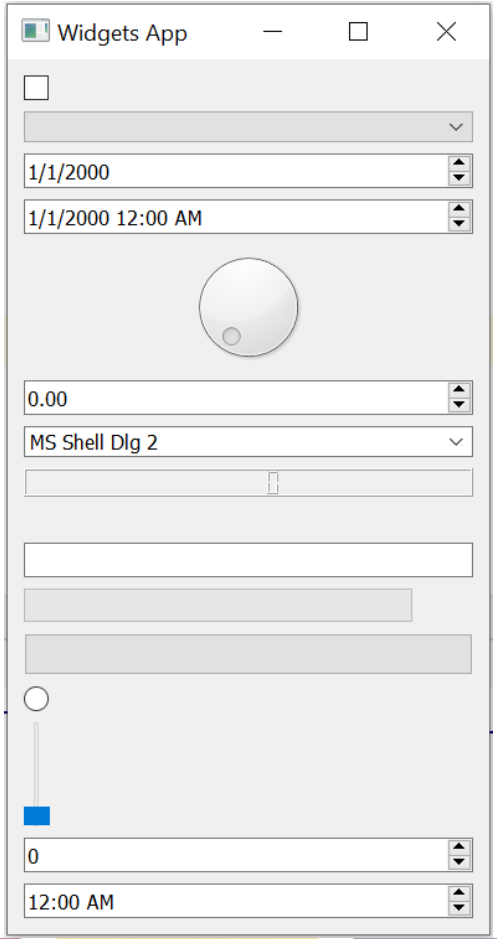
4. Các khối cơ bản xây dựng các ứng dụng GUI của PyQT

- Widgets
- Layout managers
- Dialogs
- Main windows
- Applications
- Event loops
- Signals and slots.

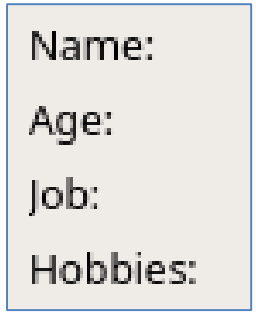
- **Widgets**
  - Là các thành phần để xây dựng GUI.
  - Widgets có các attributes và methods.
  - Widgets nhận các sự kiện (event) từ: mouse clicks, keypresses của người dùng, của hệ thống và từ nhiều nguồn khác nữa.
  - Mỗi lần Widget bắt được event, nó đưa ra các signal (tín hiệu) để thông báo sự thay đổi trạng thái. PyQt5 có các widgets phục vụ cho nhiều mục đích khác nhau:

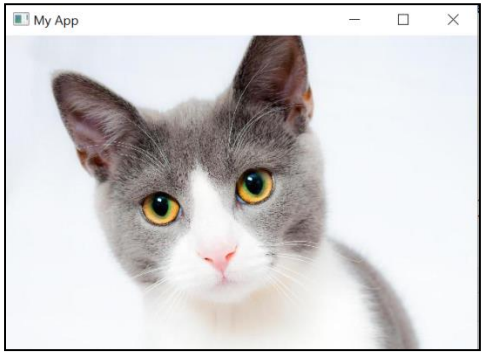
4. Các khối cơ bản xây dựng các ứng dụng GUI của PyQT: Widgets

- QCheckbox → Hộp Check box
- QComboBox → Hộp danh sách dropdown
- QDateEdit → Soạn thảo kiểu ngày (date)
- QDateTimeEdit → Soạn thảo kiểu date và datetimes
- QDial → Điều khiển xoay để thay đổi giá trị
- QDoubleSpinbox → Điều khiển thay đổi giá trị kiểu float
- QFontComboBox → Hộp chọn danh sách font
- QLCDNumber → Màn hình LCD
- QLabel → Hiển thị text hoặc hình ảnh
- QLineEdit → Nhập dữ liệu text
- QProgressBar → Thanh tiến trình
- QPushButton → Nút nhấn
- QRadioButton → Một/nhóm điều khiển có thể chọn
- QSlider → Thanh trượt
- QSpinBox → Điều khiển (xoay) thay đổi giá trị kiểu int
- QTimeEdit → Soạn thảo kiểu time



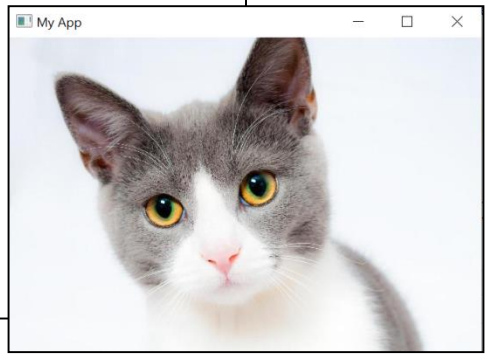
- **1. QLabel**
  - QLabel giúp hiển thị các thông tin ở dạng văn bản hoặc hình ảnh.
  - Tạo: widget = QLabel(`"Hello"`)
  - Thay đổi text: widget.setText(`"2"`)
  - Hiển thị ảnh: widget.setPixmap(QPixmap(`"img.jpg"`))
  - Kéo dãn hình ảnh cho vừa với kích thước Qlabel: widget.setScaledContents(`True`)
  - Căn lề
    - Qt.AlignLeft → Căn lề trái; Qt.AlignRight → Căn lề phải.
    - Qt.AlignHCenter → Căn giữa; Qt.AlignJustify → Căn đều trái và phải.
    - Qt.AlignTop → Căn trên; Qt.AlignBottom→ Căn dưới.
    - Qt.AlignVCenter→ Căn giữa dòng
  - Kết hợp bằng ký hiệu “|”
  - Ví dụ: align_top_left = Qt.AlignLeft | Qt.AlignTop



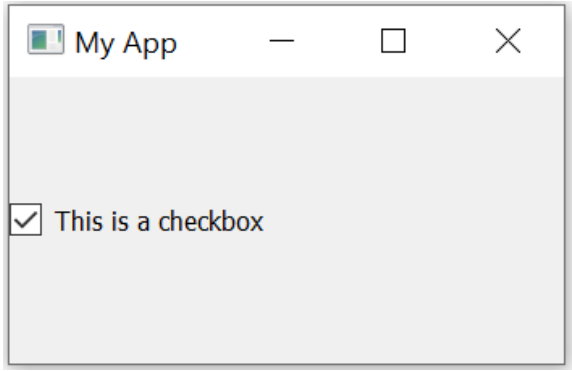


- Ví dụ về QLabel

```
import sys
from PyQt5.QtGui import QPixmap
from PyQt5.QtWidgets import QApplication, QLabel, QMainWindow

class MainWindow(QMainWindow):
    def __init__(self):
        super().__init__()
        self.setWindowTitle("My App")
        widget = QLabel("Hello")
        widget.setPixmap(QPixmap("chu_meo.jpg"))
        widget.setScaledContents(True)
        self.setCentralWidget(widget)

if __name__ == "__main__":
    app = QApplication(sys.argv)
    window = MainWindow()
    window.show()
    app.exec_()
```



- **2. QCheckBox**
  - Tạo Check box, cho phép người dùng có thể đánh dấu "tích" để kiểm tra, chọn …
  - Tạo: widget = QCheckBox("This is a checkbox")
  - Flag
    - Qt.Checked → Phần tử được chọn → trả về giá trị 2
    - Qt.Unchecked → Phần tử không được chọn → trả về giá trị 0
    - Qt.PartiallyChecked → Hộp trạng thái có 3 trạng thái để chọn
  - Signal: stateChanged → Khi trạng thái của hộp check box thay đổi
  - Ví dụ: widget.stateChanged.connect(self.show_state)



- **Signals and Slots**
  - Mỗi widget có thể bắt một số sự kiện (event) như: mouse clicks, keypresses…
  - Để phản hồi các sự kiện đó widgets đưa ra signal (tín hiệu) là một loại message để thông báo sự thay đổi trạng thái và đồng thời thực hiện một hàm/phương thức nào đó.
  - Hàm hoặc phương thức gọi là Slot.
  - Để kết nối signal với slot, dùng cú pháp

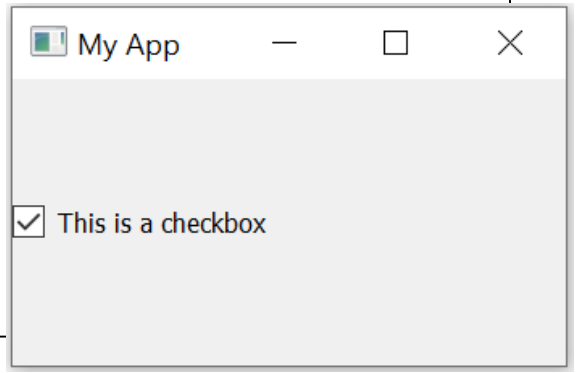
    widget.signal.connect(Hàm/phương thức)

  - Mỗi một widget có một kiểu signal khác nhau
  - Hàm/phương thức do người dùng tạo ra và gắn với một widget để thực hiện một công việc nào đó

4. Các khối cơ bản xây dựng các ứng dụng GUI của PyQT: QCheckBox

- Ví dụ về QCheckBox

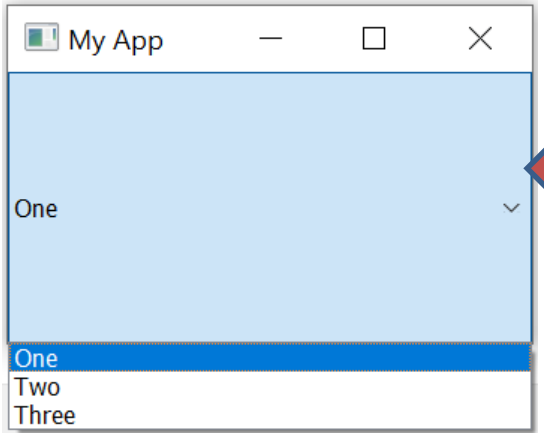
```
import sys
from PyQt5.QtWidgets import QApplication, QCheckBox, QMainWindow
class MainWindow(QMainWindow):
  def __init__(self):
      super().__init__()
      self.setWindowTitle("My App")
      widget = QCheckBox("This is a checkbox")
      widget.stateChanged.connect(self.show_state)
      self.setCentralWidget(widget)

  def show_state(self, s):
      print(s)

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



- **1.3. QComboBox**
  - Chứa danh sách các phần tử
  - Cho phép người dùng chọn các phần tử trong danh sách
  - Tạo: widget = QComboBox()
    widget.addItems(["phan_tu_1", "phan_tu_2", "phan_tu_3"])
  - Flag
    - QComboBox.NoInsert
    - QComboBox.InsertAtTop
    - QComboBox.InsertAtCurrent
    - QComboBox.InsertAlphabetically
    - QComboBox.InsertAtBottom
    - QComboBox.InsertAfterCurrent
    - QComboBox.InsertBeforeCurrent
  - Để chèn thì dùng cú pháp: widget.setInsertPolicy(QComboBox.InsertAlphabetically)
  - Signal:
    - currentIndexChanged Ví dụ: widget.currentIndexChanged.connect(self.Phương_thức)
    - currentTextChanged widget.currentTextChanged.connect(self.Phương_thức)

4. Các khối cơ bản xây dựng các ứng dụng GUI của PyQT: QComboBox

- Ví dụ QComboBox

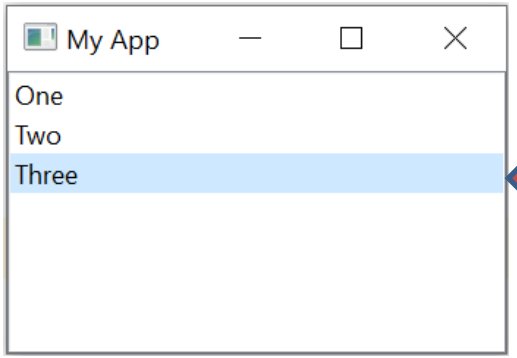



```
import sys
from PyQt5.QtWidgets import QApplication, QComboBox, QMainWindow
class MainWindow(QMainWindow):
  def __init__(self):
    super().__init__()
    self.setWindowTitle("My App")
    widget = QComboBox()
    widget.addItems(["One", "Two", "Three"])
    widget.currentIndexChanged.connect(self.index_changed)
    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 string
    print(s)

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

- **4. QListWidget**
  - Giống như QComboBox.
  - Chứa danh sách các phần tử
  - Tạo: widget = QListWidget()

    widget.addItems(["phan_tu_1", "phan_tu_2", "phan_tu_3"])
  - Signal:
    - currentIndexChanged
    - currentTextChanged
  - Ví dụ:
    - widget.currentIndexChanged.connect(self.Phương_thức)
    - widget.currentTextChanged.connect(self.Phương_thức)

- Vid dụ QListWidget

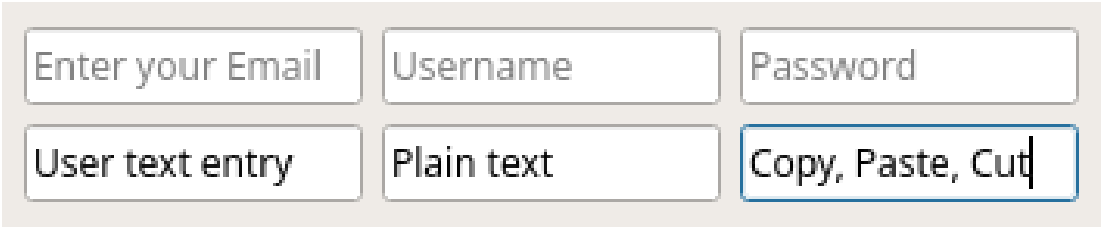



```
import sys
from PyQt5.QtWidgets import QApplication, QListWidget, QMainWindow
class MainWindow(QMainWindow):
    def __init__(self):
        super().__init__()
        self.setWindowTitle("My App")
        widget = QListWidget()
        widget.addItems(["One", "Two", "Three"])

        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_()
```

- **5. QLineEdit**
  - Cho phép người dùng nhập vào dữ liệu dạng text.
  - Tạo: widget = QLineEdit()

```
widget.setMaxLength(10) # Độ dài tối đa dữ liệu nhập
widget.setPlaceholderText("Enter your text")   # Hiển thị kèm theo chuỗi
widget.setInputMask('000.000.000.000;_')  # Thiết lập thuộc tính mặt nạ
```

  - Signal:
    - returnPressed
    - selectionChanged
    - textChanged
    - textEdited

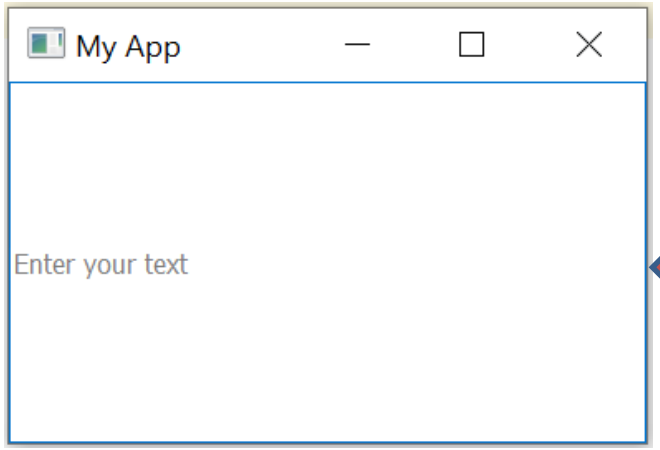


- **1.5. QLineEdit (tiếp theo)**
  - Ví dụ:
    - widget.returnPressed.connect(self.Phương_thức)
    - widget.selectionChanged.connect(self.Phương_thức)
    - widget.textChanged.connect(self.Phương_thức)
    - widget.textEdited.connect(self.Phương_thức)

- Vì dụ QLineEdit

```
import sys
from PyQt5.QtCore import Qt
from PyQt5.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.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)
```

- Ví dụ QLineEdit (tiếp theo)



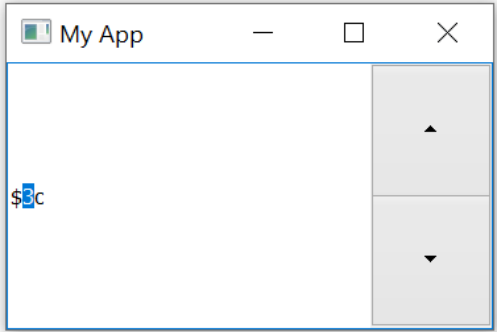

```
  def return_pressed(self):
    print("Return pressed!")
    self.centralWidget().setText("BOOM!")
  def selection_changed(self):
    print("Selection changed")
    print(self.centralWidget().selectedText())
  def text_changed(self, s):
    print("Text changed...")
    print(s)
  def text_edited(self,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

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

- **6. QSpinBox và QDoubleSpinBox**
  - Cung cấp hộp cùng với các mũi tên để cho phép người dùng tăng, giảm dữ liệu.
  - QSpinBox hỗ trợ số nguyên
  - QDoubleSpinBox hỗ trợ số thực
  - Tạo: widget = QSpinBox()

```
widget.setMinimum(-10)   # Giá trị nhỏ nhất
widget.setMaximum(3)     # Giá trị lơn nhất
widget.setSingleStep(3)  # Giá trị ở mỗi bước thay đổi
```

  - Signal:
    - valueChanged → được kích hoạt khi giá trị thay đổi và gửi tín hiệu là giá trị kiểu int hoặc float
    - valueChanged[str] → được kích hoạt khi giá trị thay đổi và gửi tín hiệu là kiểu string

- Vì dụ QSpinBox và QDoubleSpinBox




```
import sys
from PyQt5.QtWidgets import QApplication, QMainWindow, QSpinBox
class MainWindow(QMainWindow):
  def __init__(self):
    super().__init__()
    self.setWindowTitle("My App")

    widget = QSpinBox()
    widget.setMinimum(-10)
    widget.setMaximum(3)
    widget.setSingleStep(3)
    widget.valueChanged.connect(self.value_changed)
    widget.valueChanged[str].connect(self.value_changed_str)
    self.setCentralWidget(widget)
  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_()
```

- **7. QSlider**
  - QSlider cung cấp một tiện ích thanh trượt, hoạt động giống như một QDoubleSpinBox.
  - Thay vì hiển thị giá trị hiện tại bằng số, nó là được thể hiện bằng vị trí của tay cầm thanh trượt dọc/ngang theo chiều dài.
  - Ứng dụng phổ biến nhất sử dụng loại tiện ích này là để điều khiển âm lượng
  - Tạo:

```
widget = QSlider()                  # Mặc định theo chiều dọc
widget = QSlider(Qt.Horizontal) # Theo chiều dọc
widget = QSlider(Qt.Vertical)     # Theo chiều ngang

widget.setMinimum(-10)   # Giá trị nhỏ nhất
widget.setMaximum(3)     # Giá trị lơn nhất
widget.setSingleStep(3)   # Giá trị ở mỗi bước thay đổi
```

- **7. Qslider (tiếp theo)**
  - Signal:
    - valueChanged → được kích hoạt khi giá trị thay đổi
    - sliderMoved → được kích hoạt khi di chuyển
    - sliderPressed → được kích hoạt khi nhấn chuột trái
    - sliderReleased → được kích hoạt khi nhả chuột trái

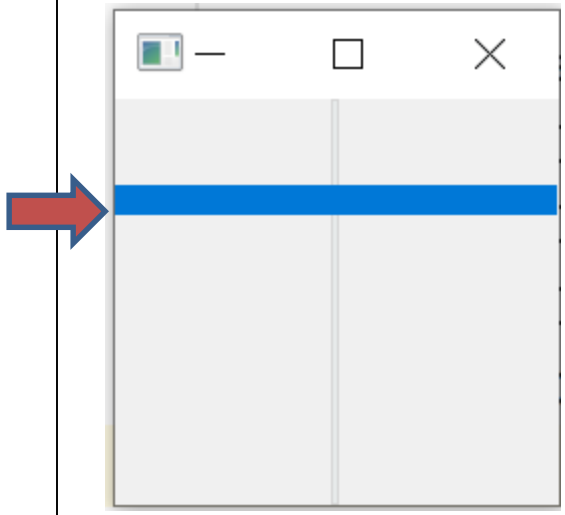
4. Các khối cơ bản xây dựng các ứng dụng GUI của PyQT: QSlider

- Ví dụ QSlider

```
import sys
from PyQt5.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)
        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)
```

- Ví dụ Qslider (tiếp theo)

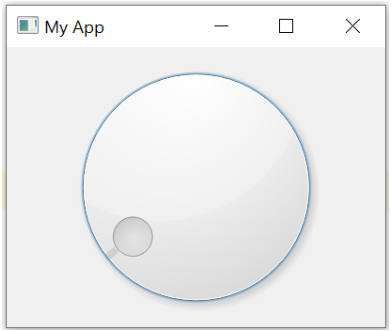
```
    def value_changed(self, i):
        print(i)
    def slider_position(self, p):
        print("position", p)
    def slider_pressed(self):
        print("Pressed!")
    def slider_released(self):
        print("Released")

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



- **8. QDial**
  - QDial là một tiện ích có thể xoay
  - Hoạt động giống như thanh trượt, nhưng xuất hiện dưới dạng quay số
  - Chúng thường được sử dụng trong các ứng dụng liên quan điều khiển âm lượng
  - Tạo: widget = QDial()

    widget.setRange(-10, 100) # Vùng giá trị điều chỉnh

    widget.setSingleStep(0.5) # Giá trị ở mỗi bước thay đổi
  - Signal:
    - valueChanged → được kích hoạt khi giá trị thay đổi
    - sliderMoved → được kích hoạt khi di chuyển
    - sliderPressed → được kích hoạt khi nhấn chuột trái
    - sliderReleased → được kích hoạt khi nhả chuột trái

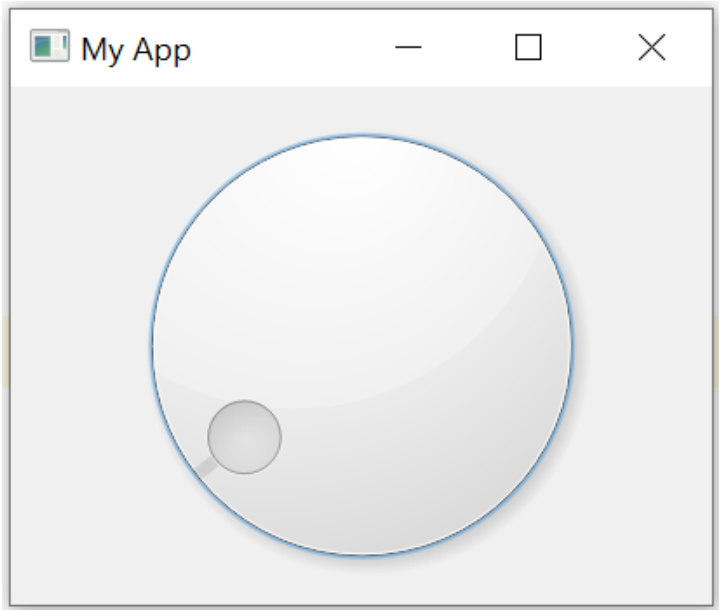


- Ví dụ QDial

```
import sys
from PyQt5.QtWidgets import QApplication, QDial, QMainWindow
class MainWindow(QMainWindow):
    def __init__(self):
        super().__init__()

        self.setWindowTitle("My App")

        widget = QDial()
        widget.setRange(-10, 100)
        widget.setSingleStep(1)
        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)
```

- Ví dụ Qdial (tiếp theo)

```
    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_()
```



- **9. QPushButton**
  - QPushButton là một tiện ích nút nhấn
  - Tạo: button = QPushButton(“Nội dung hiển thị trên nút nhấn")
  - Signal:
    - clicked → được kích hoạt khi click chuột trái
    - released → được kích hoạt khi thả chuột
  - Thuộc tính:
    - isChecked() → Trả về giá trị True nếu nút được nhấn
    - setCheckable() → Nếu có giá trị True thì nhận biết trạng thái nhấn và nhả
    - text() → Thiết lập nội dung hiển thị trên nút nhấn
    - setIcon() → Thiết lập icon trên nút nhấn
    - setEnabled() → Khi có giá trị False, nút sẽ bị tắt

```
def btnstate(self):
    If self.b1.isChecked():
        print "button pressed"
    else:
        print "button released"
```

```
b2.setIcon(QIcon(QPixmap("python.gif")))
```

4. Các khối cơ bản xây dựng các ứng dụng GUI của PyQT: QPushButton

- Ví dụ QPushButton

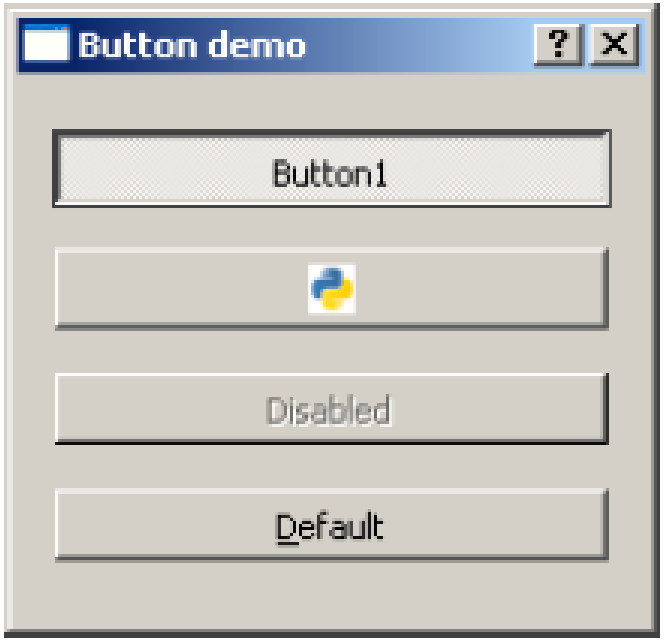
```
import sys
from PyQt4.QtCore import *
from PyQt4.QtGui import *
class Form(QDialog):
    def __init__(self, parent=None):
        super(Form, self).__init__(parent)
        layout = QVBoxLayout()
        self.b1 = QPushButton("Button1")
        self.b1.setCheckable(True)
        self.b1.toggle()
        self.b1.clicked.connect(lambda:self.whichbtn(self.b1))
        self.b1.clicked.connect(self.btnstate)
```

- Ví dụ QPushButton

```
        layout.addWidget(self.b1)
        self.b2 = QPushButton()
        self.b2.setIcon(QIcon(QPixmap("python.gif")))
        self.b2.clicked.connect(lambda:self.whichbtn(self.b2))

        layout.addWidget(self.b2)
        self.setLayout(layout)
        self.b3 = QPushButton("Disabled")
        self.b3.setEnabled(False)

        layout.addWidget(self.b3)
        self.b4 = QPushButton("&Default")
        self.b4.setDefault(True)
        self.b4.clicked.connect(lambda:self.whichbtn(self.b4))
```

4. Các khối cơ bản xây dựng các ứng dụng GUI của PyQT: QPushButton

- Ví dụ QPushButton



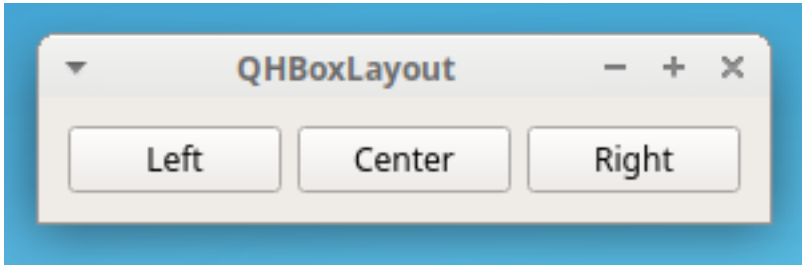

```
        layout.addWidget(self.b4)
        self.setWindowTitle("Button demo")
    def btnstate(self):
        if self.b1.isChecked():
            print "button pressed"
        else: print "button released"
    def whichbtn(self,b):
        print "clicked button is "+b.text()
def main():
    app = QApplication(sys.argv)
    ex = Form() ex.show()
    sys.exit(app.exec_())
if __name__ == '__main__':
    main()
```

5. Layout Managers

- Để sắp xếp các widgets trong một GUI ➔ sử dụng layout manager ➔ giúp tăng hiệu suất và cải thiện khả năng bảo trì code (maintainability).
- Layout managers là các classes cho phép định kích thước và vị trí của widgets để trên application form. Layout managers tự động điều chỉnh để thay đổi kích thước events và nội dung. Các widgets trong layout sẽ tự động được thay đổi kích thước bất cứ khi bào form thay đổi kích thước.
- PyQt cung cấp 4 basic layout manager classes:
  - QHBoxLayout
  - QVBoxLayut
  - QGridLayout
  - QFormLayout

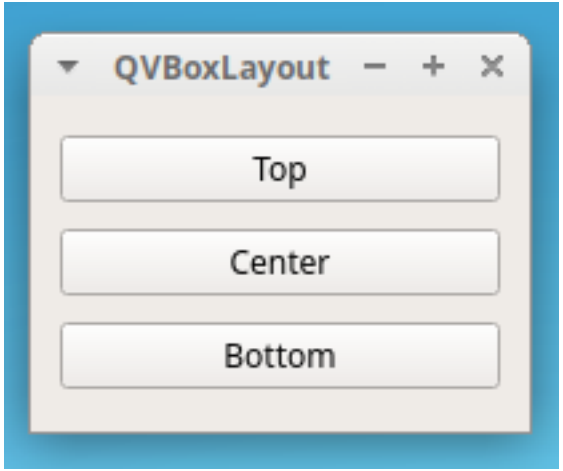
- Ví dụ 1: Sử dụng QHBoxLayout sắp xếp các widgets theo chiều ngang từ trái qua phải:

```
"""Horizontal layout example."""
import sys
from PyQt5.QtWidgets import QApplication
from PyQt5.QtWidgets import QHBoxLayout
from PyQt5.QtWidgets import QPushButton
from PyQt5.QtWidgets import QWidget

app = QApplication(sys.argv)
window = QWidget()
window.setWindowTitle('QHBoxLayout')

# tạo QHBoxLauout object
layout = QHBoxLayout()

# thêm các buttons vào layout với .addWidget()
layout.addWidget(QPushButton('Left'))
layout.addWidget(QPushButton('Center'))
layout.addWidget(QPushButton('Right'))

# set layout là window's layout
window.setLayout(layout)

window.show()
sys.exit(app.exec_())
```



- Ví dụ 2: Sử dụng QVBoxLayout sắp xếp các widgets theo chiều dọc:

```
"""Vertical layout example."""
import sys

from PyQt5.QtWidgets import QApplication
from PyQt5.QtWidgets import QVBoxLayout
from PyQt5.QtWidgets import QPushButton
from PyQt5.QtWidgets import QWidget

app = QApplication(sys.argv)
window = QWidget()
window.setWindowTitle('QVBoxLayout')

# tạo QVBoxLayout object
layout = QVBoxLayout()

# thêm 3 buttons vào layout
layout.addWidget(QPushButton('Top'))
layout.addWidget(QPushButton('Center'))
layout.addWidget(QPushButton('Bottom'))

# set layout cho window
window.setLayout(layout)

window.show()
sys.exit(app.exec_())
```



- Ví dụ 3: Sử dụng QGridLayout sắp xếp các widgets theo dạng grid:

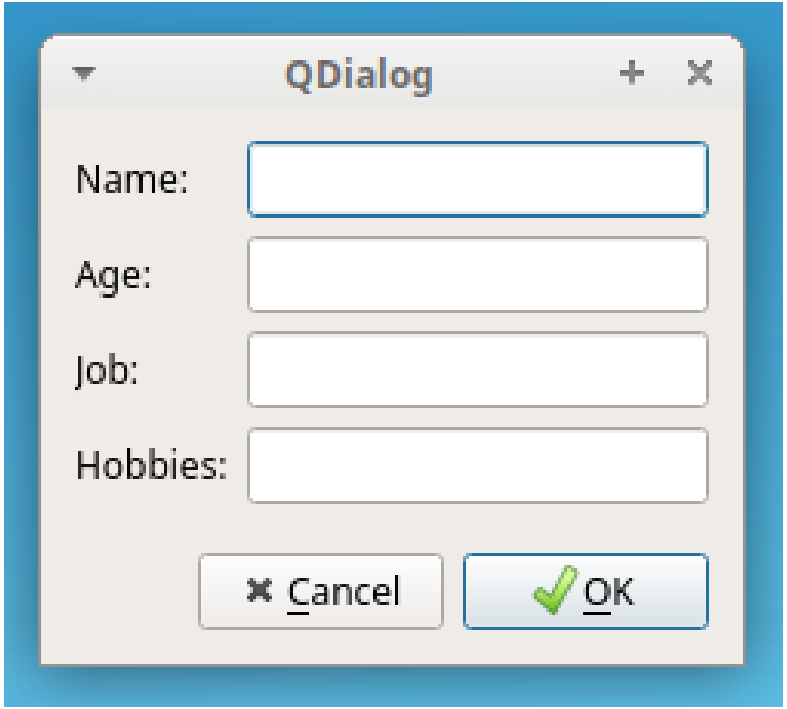
```
"""Grid layout example."""
import sys

from PyQt5.QtWidgets import QApplication
from PyQt5.QtWidgets import QGridLayout
from PyQt5.QtWidgets import QPushButton
from PyQt5.QtWidgets import QWidget

app = QApplication(sys.argv)
window = QWidget()
window.setWindowTitle('QGridLayout')

# tạo QGridLayout object
layout = QGridLayout()

# thêm các widgets vào thông qua .addWidget(), tham
# số thứ hai và ba là row và column => xác định vị trí
# widget

layout.addWidget(QPushButton('Button (0, 0)'), 0, 0)
layout.addWidget(QPushButton('Button (0, 1)'), 0, 1)
layout.addWidget(QPushButton('Button (0, 2)'), 0, 2)
layout.addWidget(QPushButton('Button (1, 0)'), 1, 0)
layout.addWidget(QPushButton('Button (1, 1)'), 1, 1)
layout.addWidget(QPushButton('Button (1, 2)'), 1, 2)
layout.addWidget(QPushButton('Button (2, 0)'), 2, 0)
layout.addWidget(QPushButton('Button (2, 1) + 2
                                    Columns Span'), 2, 1, 1, 2)

# set layout cho window
window.setLayout(layout)
window.show()
sys.exit(app.exec_())
```

- Ví dụ 4: Sử dụng QFormLayout sắp xếp các widgets theo dạng form:

```
"""Form layout example."""

import sys

from PyQt5.QtWidgets import QApplication
from PyQt5.QtWidgets import QFormLayout
from PyQt5.QtWidgets import QLineEdit
from PyQt5.QtWidgets import QWidget

app = QApplication(sys.argv)
window = QWidget()
window.setWindowTitle('QFormLayout')

# tạo QFormLayout object
layout = QFormLayout()
```

```
# thêm các widgets theo hàng, một hàng có 2 cột
# tương ứng 2 widgets

layout.addRow('Name:', QLineEdit())
layout.addRow('Age:', QLineEdit())
layout.addRow('Job:', QLineEdit())
layout.addRow('Hobbies:', QLineEdit())

# set layout cho window
window.setLayout(layout)

window.show()
sys.exit(app.exec_())
```

- Để phát triển Ứng dụng kiểu Dialog thì cần tạo lớp GUI class kế thừa từ Qdialog
- Dialogs chứa các Widget, các nút OK, Cancel…



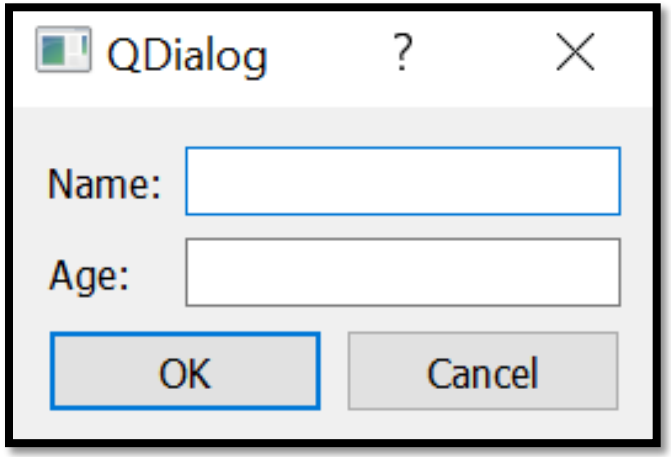
- Ví dụ: Sử dụng QDialog để tạo hộp hội thoại

```
"""Dialog-Style application."""

import sys

from PyQt5.QtWidgets import QApplication
from PyQt5.QtWidgets import QDialog
from PyQt5.QtWidgets import QDialogButtonBox
from PyQt5.QtWidgets import QFormLayout
from PyQt5.QtWidgets import QLineEdit
from PyQt5.QtWidgets import QVBoxLayout
```

```
# xây dựng full class dialog cho GUI, kết thừa từ QDialog
class Dialog(QDialog):
    """Dialog."""
    def __init__(self, parent=None):
        """Initializer."""
        super().__init__(parent)
        self.setWindowTitle('QDialog')

        # tạo 2 object layout
        dlgLayout = QVBoxLayout()
        formLayout = QFormLayout()

# thêm các widgets vào formLayout thông qua .addRow()
        formLayout.addRow('Name:', QLineEdit())
        formLayout.addRow('Age:', QLineEdit())
```

```
# sử dụng dlgLayout để sắp xếp các widgets trên biểu mẫu (form)
    # layout trong layout
    dlgLayout.addLayout(formLayout)

    # object để đặt dialog buttons
    btns = QDialogButtonBox()

    # thêm 2 buttons chuẩn Ok và Cancel
    btns.setStandardButtons(QDialogButtonBox.Cancel
                            | QDialogButtonBox.Ok)


    dlgLayout.addWidget(btns)
    self.setLayout(dlgLayout
```

```
if __name__ == '__main__':
    app = QApplication(sys.argv)
    dlg = Dialog()
    dlg.show()
    sys.exit(app.exec_())
```



5. Main Windows

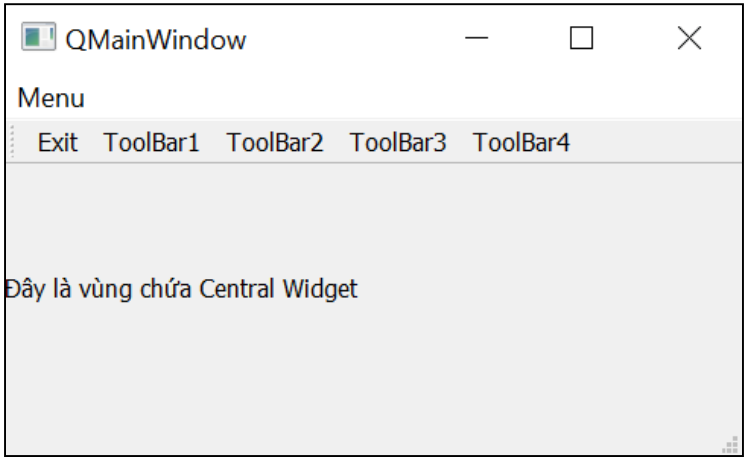
- Hầu hết các ứng dụng GUI có kiểu Main window.
- Một ứng dụng Main Windows gồm:
  - Menu bar chứa menu chính của ứng dụng
  - Toolbars thích hợp để chứa các buttons và một số loại widgets khác như QComboBox, QSpinBox…
  - Widget trung tâm có thể là bất kỳ loại nào hoặc có thể là một widget tổng hợp.
  - Dock widgets là các windows nhỏ và có thể di chuyển.
  - Status bar hiển thị thông tin về trạng thái của ứng dụng.
- Ngoài ra, các ứng dụng sẽ có một vài Hộp hội thoại dialog để thực hiện tác vụ thứ cấp phụ thuộc vào đầu vào của người dùng (user input)

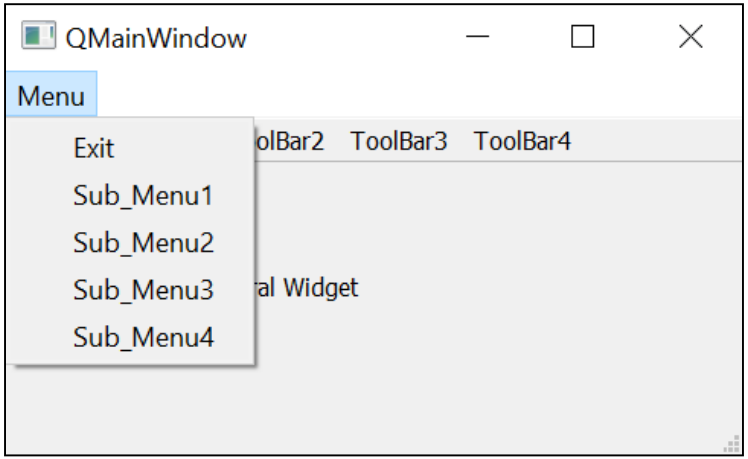
Ví dụ: Thiết kế giao diện ứng dụng Main Windows

```
"""Main Window-Style application."""
import sys

from PyQt5.QtWidgets import QApplication
from PyQt5.QtWidgets import QLabel
from PyQt5.QtWidgets import QMainWindow
from PyQt5.QtWidgets import QStatusBar
from PyQt5.QtWidgets import QToolBar

# tạo class window kế thừa từ QMainWindow
class Window(QMainWindow):
    """Main Window."""
    def __init__(self, parent=None):
        """Initializer."""
        super().__init__(parent)
        # Thiết lập tiêu đề cửa sổ
        self.setWindowTitle('QMainWindow')
        # Thiết lập central widget là QLabel
        self.setCentralWidget(QLabel("Đây là vùng chứa Central Widget"))
        # các private methods để tạo các thành phần ứng dụng GUI
        self._createMenu()
        self._createToolBar()
        self._createStatusBar()
```

```
def _createMenu(self):
    self.menu = self.menuBar().addMenu("&Menu")
    self.menu.addAction('&Exit', self.close)
    self.menu.addAction('&Sub_Menu1')
    self.menu.addAction('&Sub_Menu2')
    self.menu.addAction('&Sub_Menu3')
    self.menu.addAction('&Sub_Menu4')


def _createToolBar(self):
    tools = QToolBar()
    self.addToolBar(tools)
    tools.addAction('Exit', self.close)
    tools.addAction('ToolBar1')
    tools.addAction('ToolBar2')
    tools.addAction('ToolBar3')
    tools.addAction('ToolBar4')


def _createStatusBar(self):
    status = QStatusBar()
    status.showMessage("Đây là Status Bar")
    self.setStatusBar(status)
```

```
if __name__ == '__main__':
    app = QApplication(sys.argv)
    win = Window()
    win.show()
    sys.exit(app.exec_())
```

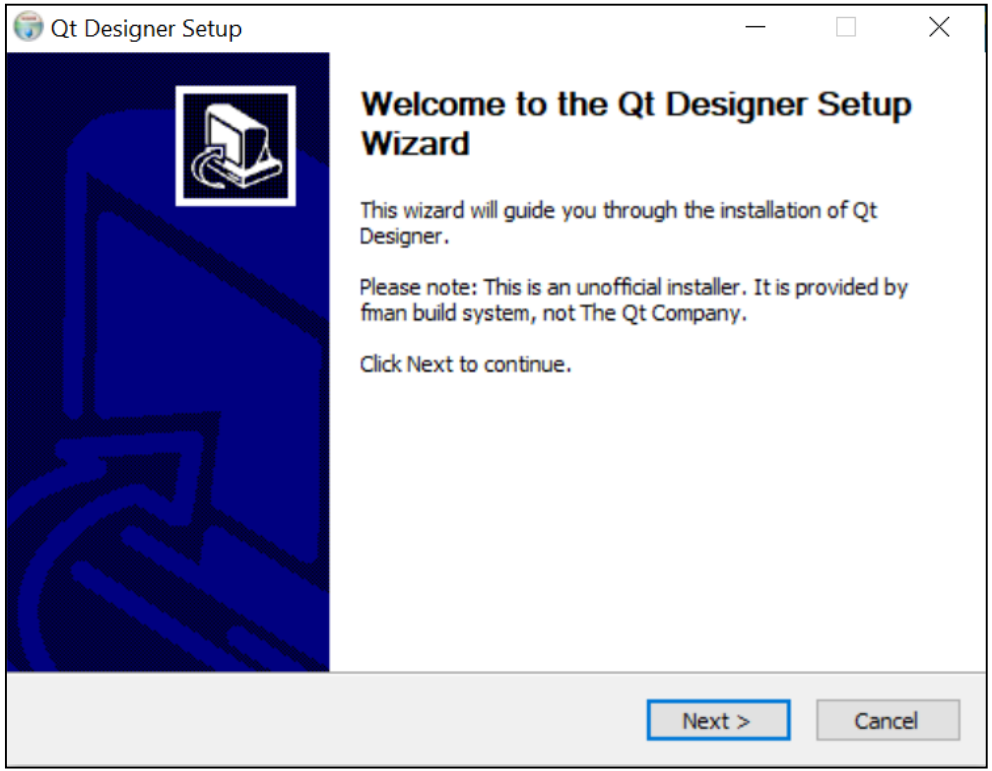




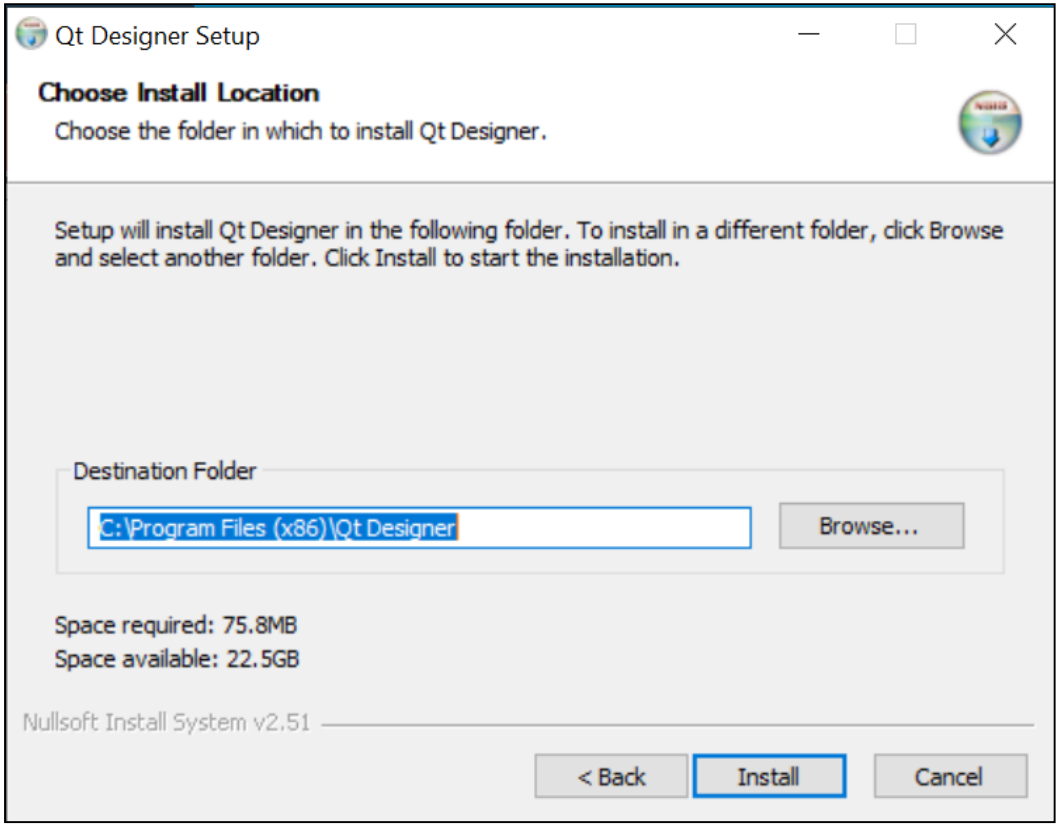
PHẦN 2
Sử dụng QtDesigner

Bước 1: Cài đặt QtDesigner

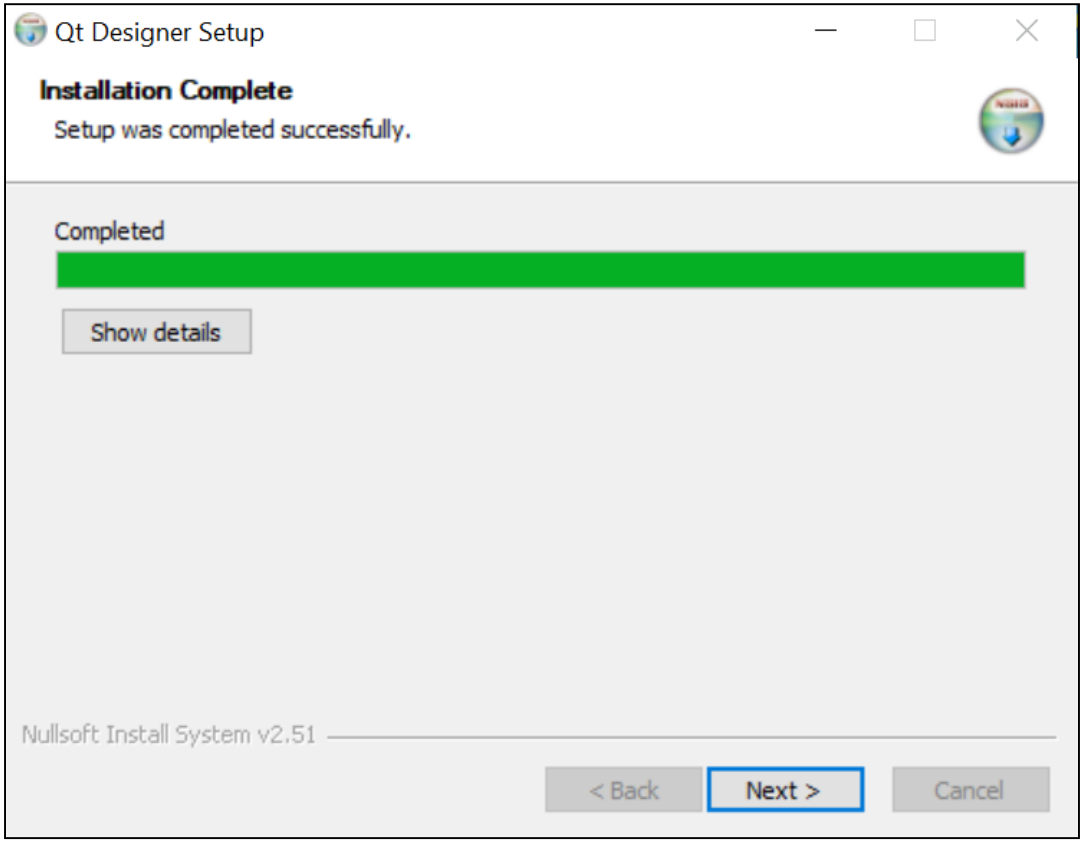
- Tải về phần mềm Qt Designer tại đường dẫn sau: https://build-system.fman.io/qt-designer-download.
- Cài đặt:
- B1: Double Click vào file Qt Designer Setup.exe
- B2: Click nút Next



- B3: Chọn đường dẫn hoặc để mặc định rồi Click nút Install

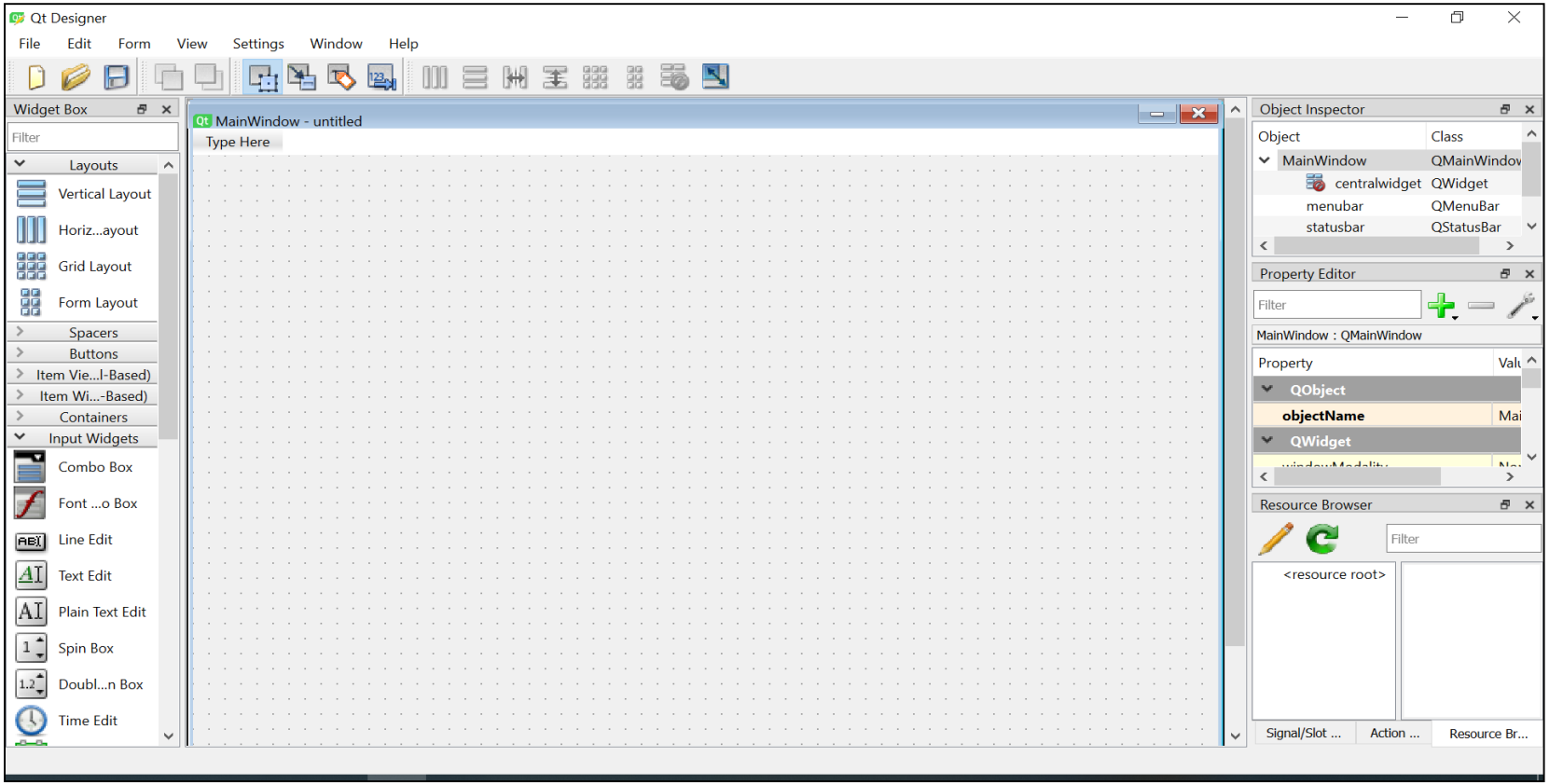


- B4: Clilck nút Next ➔ Finish



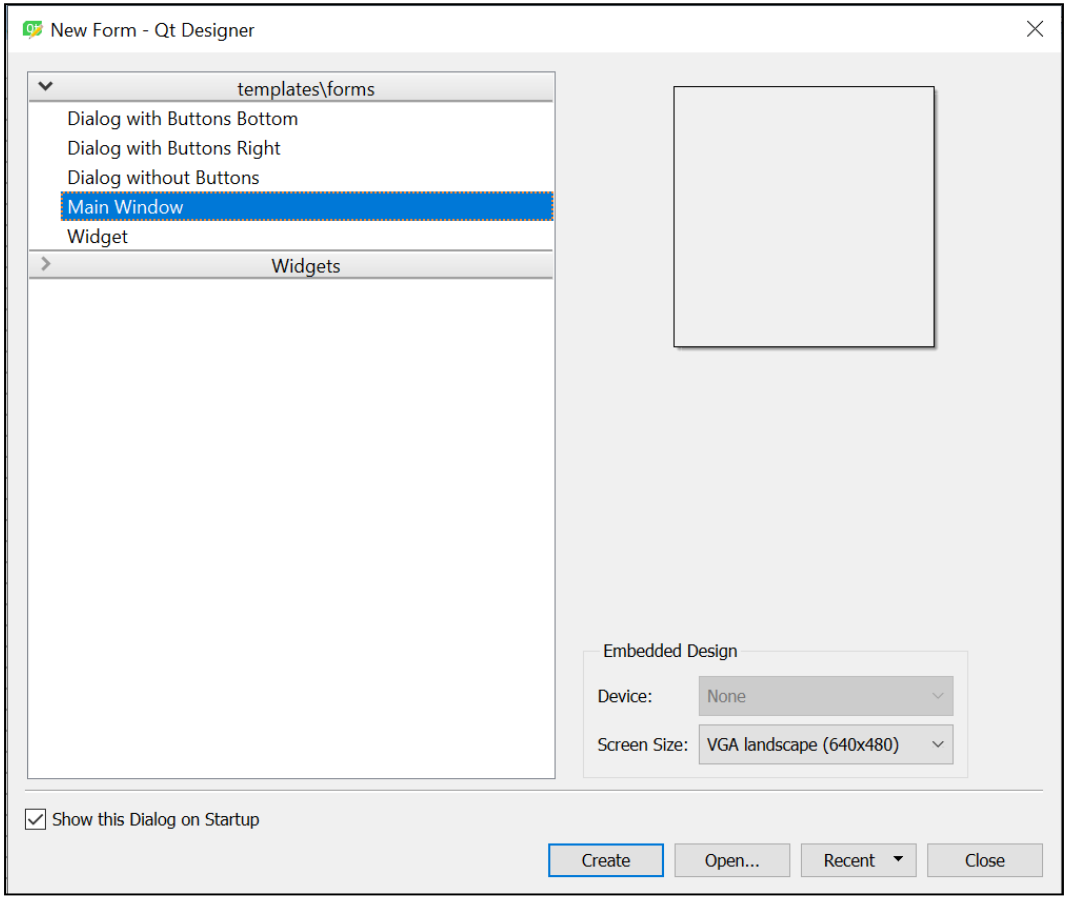
Bước 2: Khởi động QtDesigner và tao file *.UI

- Khởi động: Click nút Start ➔ Qt Designer



Bước 2: Khởi động QtDesigner và tao file *.UI (tiếp theo)

- B1: Click File ➔ New
- B2: Click kiểu form
- B3: Click nút Create
- Lưu ý: Nhấn Ctrl-R để xem kết quả



- Cách 1: Dịch file thiết kế (*.UI) sang file Python (*.Py)
  - Tại terminal của IDE (ví dụ pycharm) gõ: pyuic5 <file *.ui> -o <file *.py> -x
  - Sau đó biên soạn lại code cho file *.py để tạo ra chương trình ứng dụng hoàn thiện
  - Bổ sung đoạn code:

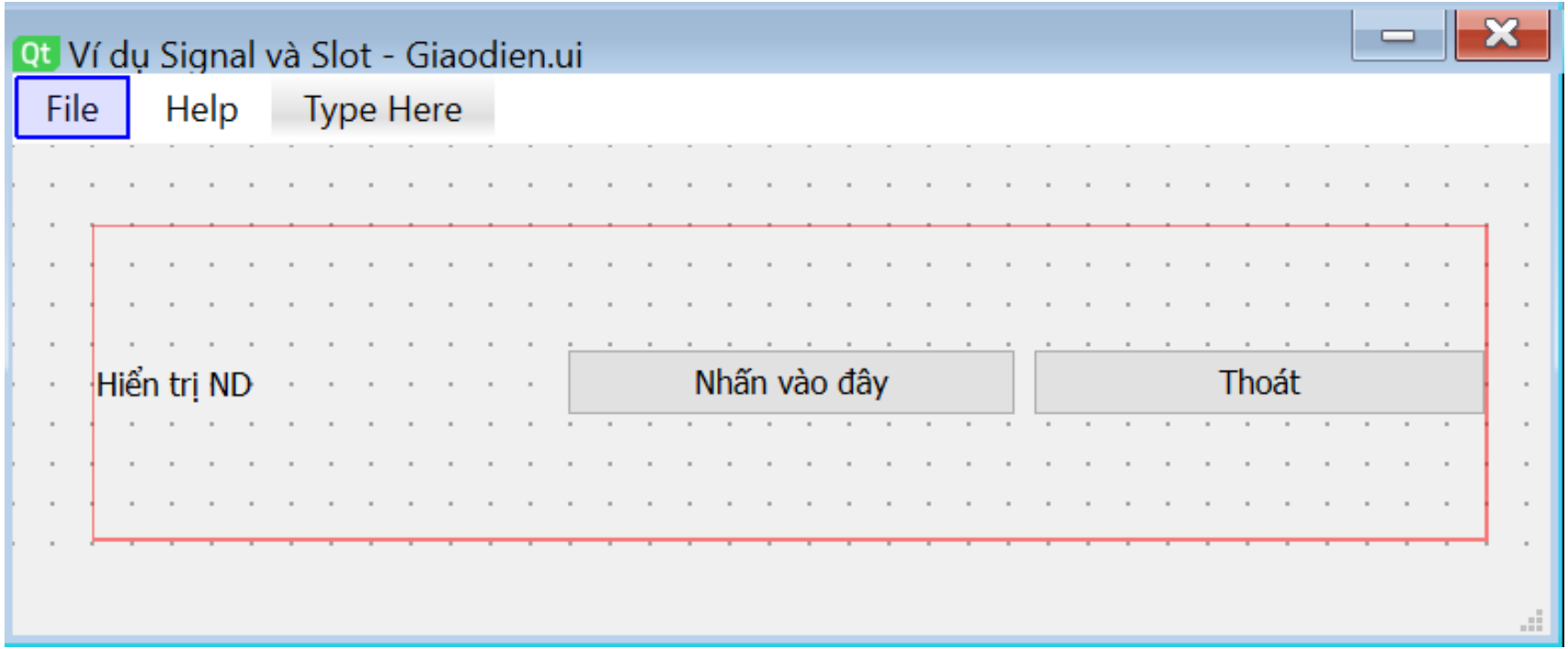
```
if __name__ == "__main__":
    import sys
    app = QtWidgets.QApplication(sys.argv)
    MainWindow = QtWidgets.QMainWindow()
    ui = Ui_MainWindow()
    ui.setupUi(MainWindow)
    MainWindow.show()
    sys.exit(app.exec_())
```

  - Thực thi file *.py

- Cách 2: Nạp file thiết kế (*.UI) bên trong file python (*.py). Tức là tạo file python:

```
from PyQt5 import uic
from PyQt5.QtWidgets import QApplication
Form, Window = uic.loadUiType("file *.ui") # Nạp file thiết kế
app = QApplication([])
window = Window()
form = Form()
form.setupUi(window)
window.show()
app.exec()
```

- **Bài 1.**
  - Bước 1: Khởi động QT Designer và thiết kế giao diện sau



- **Bài 1.**
  - Yêu cầu:
    - Menu File có 2 sub menu: Open và Exit
    - Menu Help có 1 sub menu: About
    - Nút nhấn 1 có tên là: button1 và hiển thị text “Nhấn vào đây”
    - Nút nhấn 2 có tên là: button_thoát và hiển thị text “Thoát”
    - Label có tên là Label và hiển thị text “Hiển thị ND”
  - Bước 2: Biên dịch sang file *.py: pyuic5 <file *.ui> -o <file *.py>
  - Bước 3: Mở file *.py và thêm các đoạn code sau:

Định nghĩa 2 hàm mới:

```
# Định nghĩa hàm (slot) hiển thị chuỗi lời chào ra label
def HienThi(self):
    if  self.label.text():
        self.label.setText("")
    else:
        self.label.setText("Hello World!")

def Thoat(self):
    MainWindow.close()
```

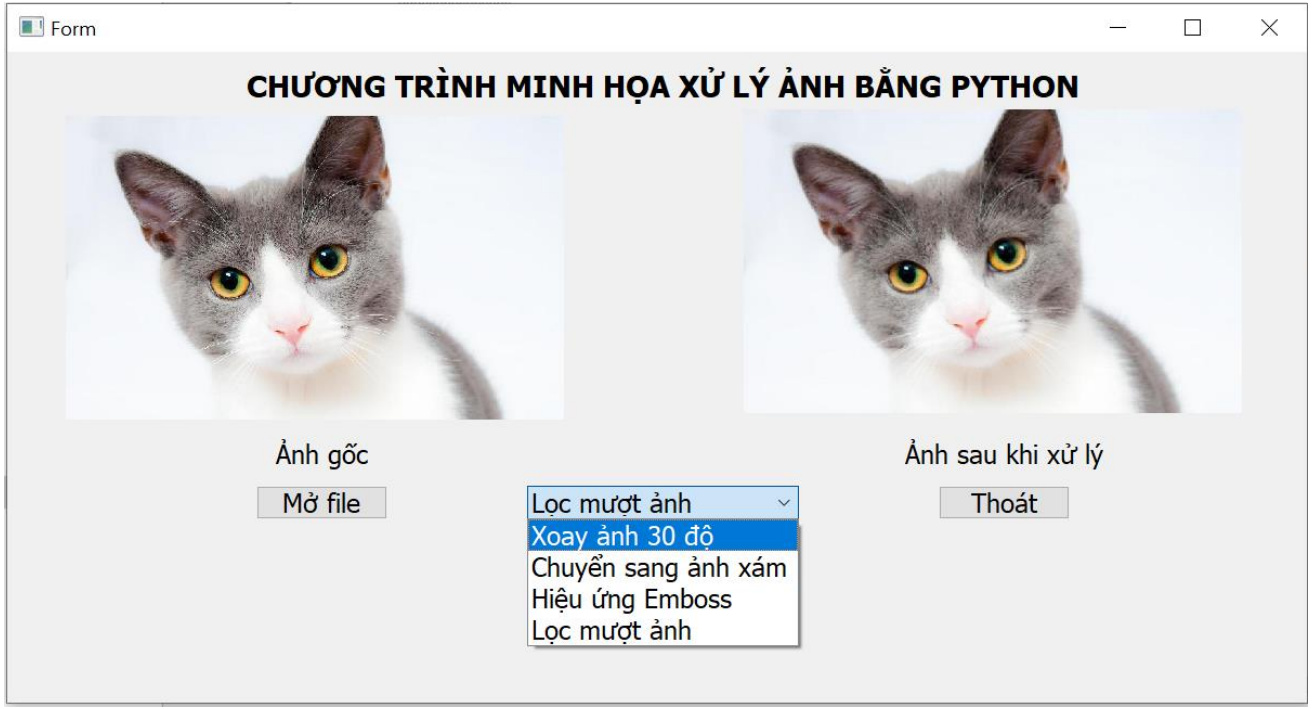
Trong định nghĩa hàm retranslateUi() thêm 2 dòng lệnh để kết nối Signal và Slot

```
# Kết nối Signal và Slot
self.button1.clicked.connect(self.HienThi)
self.button_Thoat.clicked.connect(self.Thoat)
```

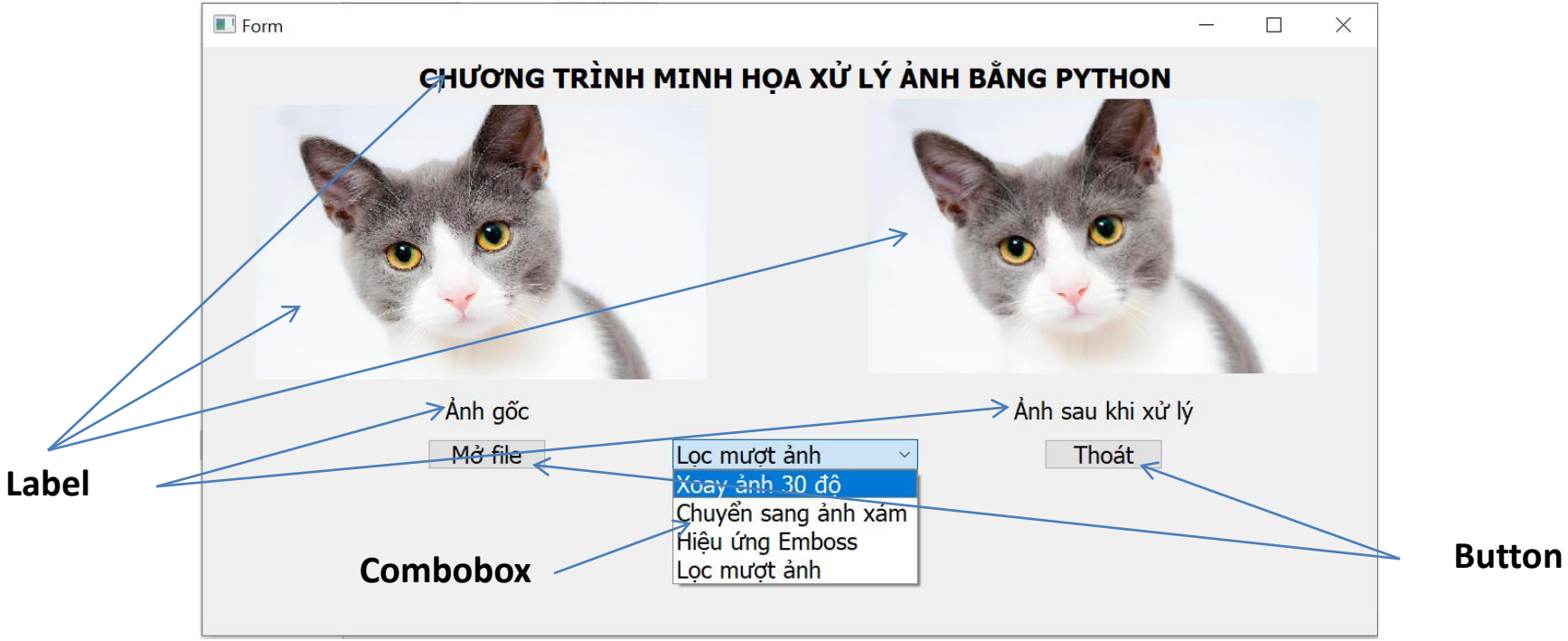
Chuyển đến cuối chương trình, thêm:

```
if __name__ == "__main__":
    app = QtWidgets.QApplication(sys.argv)
    MainWindow = QtWidgets.QMainWindow()
    ui = Ui_MainWindow()
    ui.setupUi(MainWindow)
    MainWindow.show()
    sys.exit(app.exec_())
```

- **Bài 2.** Xây dựng ứng dụng xử lý ảnh



- **Bước 1.** Khởi động QT Designer và thiết kế giao diện. Nhấn Ctrl-R để xem kết quả



- **Bước 2:** Lưu file *.ui và chuyển file *.ui sang file *.py bằng câu lệnh

pyuic5 <file *.ui> -o <file *.py>

- **Bước 3:** Mở file *.py để thêm các đoạn code sau
  - Tạo hàm cho nút Mở file và hiển thị trong Label:

```
def Open(self):
    self.filename = QFileDialog.getOpenFileName(filter="*.jpg;;*.png;;*.jpeg;;*.bmp")[0]
    pixmap = QtGui.QPixmap(self.filename)
    if pixmap.width() > self.label_4.width():
        self.pixmap = pixmap.scaled(460, 280)
        self.label_4.setPixmap(self.pixmap)
        self.label_4.resize(460, 280)
    else:
        self.label_4.setPixmap(pixmap)
```

  - Tạo hàm cho nút Thoát:

```
def Exit(self):
    Form.close()
```

- **Bước 3:** Mở file *.py để thêm các đoạn code sau
  - Tạo hàm cho các Combobox:
    - Xoay ảnh

```
img = Image.open(self.filename)
if text=="Xoay ảnh 30 độ":
    imgtam = img.rotate(30)  # Xoay ảnh 30 độ
    imgtam.save("xoay.jpg")
    pixmap = QtGui.QPixmap("xoay.jpg")
    if pixmap.width() > self.label_5.width():
        self.pixmapnew = pixmap.scaled(460, 280)
        self.label_5.setPixmap(self.pixmapnew)
        self.label_5.resize(460, 280)
    else:
        self.label_5.setPixmap(pixmap)
```

- **Bước 3:** Mở file *.py để thêm các đoạn code sau
  - Tạo hàm cho các Combobox:
    - Chuyển sang ảnh xám

```
elif text=="Chuyển sang ảnh xám":
    imgtam = img.convert("L")  # Chuyển sang ảnh xám (ảnh đen trắng)
    imgtam.save("xam.jpg")
    pixmap = QtGui.QPixmap("xam.jpg")
    if pixmap.width() > self.label_5.width():
        self.pixmapnew = pixmap.scaled(460, 280)
        self.label_5.setPixmap(self.pixmapnew)
        self.label_5.resize(460, 280)
    else:
        self.label_5.setPixmap(pixmap)
```

- **Bước 3:** Mở file *.py để thêm các đoạn code sau
  - Tạo hàm cho các Combobox:
    - Hiệu ứng Emboss

```
elif text=="Hiệu ứng Emboss":
    # Tạo hiệu ứng khắc nổi
    imgtam = img.filter(ImageFilter.EMBOSS)
    imgtam.save("khac_noi.jpg")
    pixmap = QtGui.QPixmap("khac_noi.jpg")
    if pixmap.width() > self.label_5.width():
        self.pixmapnew = pixmap.scaled(460, 280)
        self.label_5.setPixmap(self.pixmapnew)
        self.label_5.resize(460, 280)
    else:
        self.label_5.setPixmap(pixmap)
```

- **Bước 3:** Mở file *.py để thêm các đoạn code sau
  - Tạo hàm cho các Combobox:
    - Lọc mượt ảnh

```
        else:
            imgtam = img.filter(ImageFilter.SMOOTH)
            imgtam.save("loc_muot.jpg")
            pixmap = QtGui.QPixmap("loc_muot.jpg")
            if pixmap.width() > self.label_5.width():
                self.pixmapnew = pixmap.scaled(460, 280)
                self.label_5.setPixmap(self.pixmapnew)
                self.label_5.resize(460, 280)
            else:
                self.label_5.setPixmap(pixmap)
```

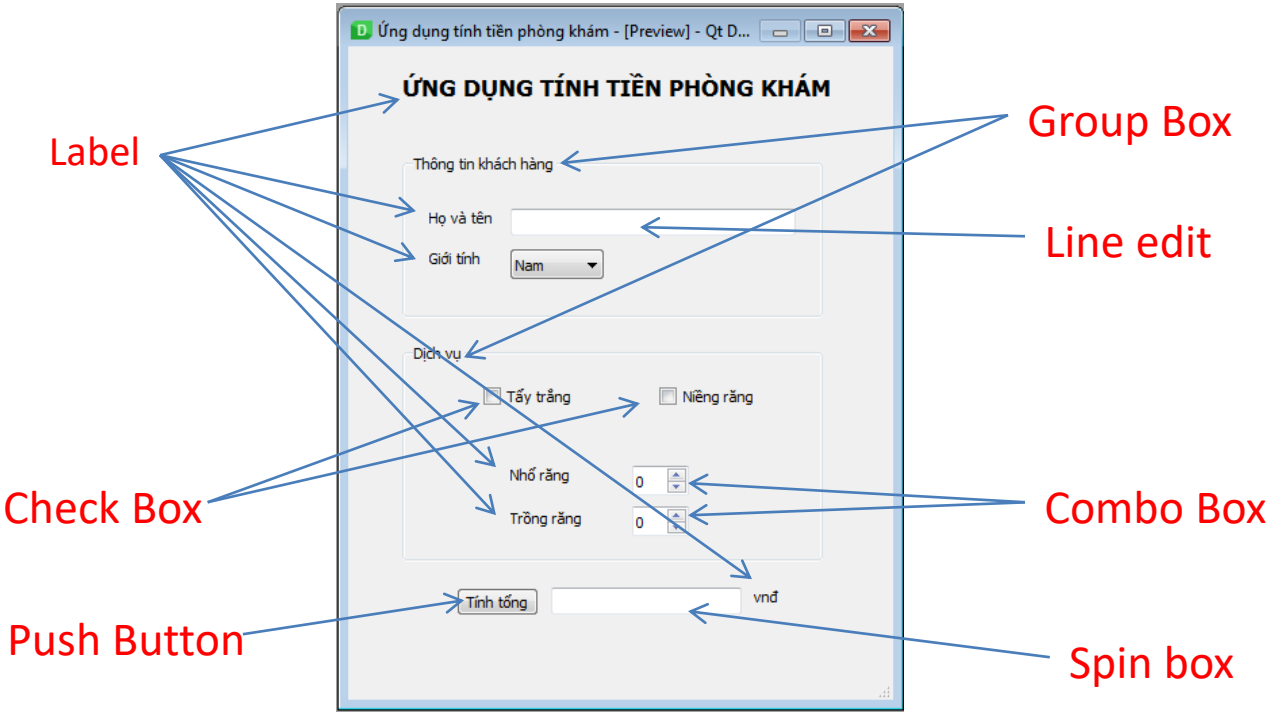
- **Bước 4:** Kết nối Signal và Slot

```
self.pushButton.clicked.connect(self.Exit)
self.pushButton_Open.clicked.connect(self.Open)
self.comboBox.activated[str].connect(self.Chon_xu_ly)
```

- **Bước 5**: Thêm Code cho hàm main

```
if __name__ == "__main__":
    app = QtWidgets.QApplication(sys.argv)
    Form = QtWidgets.QMainWindow()
    ui = Ui_Form()
    ui.setupUi(Form)
    Form.show()
    sys.exit(app.exec_())
```

- **Bài 3. Xây dựng ứng dụng tính tổng tiền tại một phòng khám**



- Label: Dùng để hiển thị nội dụng text đơn thuần.
- Group Box: Dùng để nhóm các Widget có cùng mục đích lại với nhau.
- Line edit: Dùng để nhập dữ liệu dạng 1 dòng.
- Combo Box: Dùng để người dùng chọn các nội dung có sẵn.
- Check Box: Cho phép người dụng chọn hoặc bỏ chọn 1 nội dung.
- Spin box: Tương tự như Line Edit nhưng chỉ được phép nhập số.
- Push Button: Đơn giản là một cái nút nhấn.

- **Bước 1.** Khởi động QT Designer và thiết kế giao diện
  - Lưu ý: Chọn kiểu Main Window, và ID của các Widget để viết Code cho đúng
  - Ấn Ctrl – H để xem trước giao diện kết quả
  - Click vào Window bao ngoài và chỉnh sửa WindowTitle để thay đổi title cửa sổ
- **Bước 2:** Lưu file *.ui và chuyển file *.ui sang file *.py bằng câu lệnh

pyuic5 <file *.ui> -o <file *.py>

- **Bước 3:** Mở file *.py để thêm các đoạn code sau
  - Định nghĩa hàm để tính nút tính tổng

```
def calcTotal(self):
    prices = {
        "tayTrang": 100000,
        "niengRang": 200000,
        "nhoRang": 50000,
        "trongRang": 500000,
    }
    sum = (0 if self.tayTrangCb.checkState() == 0 else prices["tayTrang"])
\
          + (0 if self.niengRangCb.checkState() == 0 else
prices["niengRang"]) \
          + int(self.nhoRangNum.text()) * prices["nhoRang"] \
          + int(self.trongRangNum.text()) * prices["trongRang"]

    self.total.setText(str(sum))
```

- **Bước 3:** Mở file *.py để thêm các đoạn code sau
  - Định nghĩa hàm để tính nút tính tổng (tiếp theo)

```
# Hiển thị hộp hội thoại
msg = QtWidgets.QMessageBox()
msg.setIcon(QtWidgets.QMessageBox.Information)
msg.setText("Chào {} {}!".format("Ông" if self.sex.currentText() == "Nam"
else "Bà", self.fullname.text()))
msg.setInformativeText("Tổng chi phí là {} vnđ".format(str(sum)))
msg.setWindowTitle("Chi phí")
msg.exec_()
```

- **Bước 4:** Kết nối Signal và Slot

```
self.calcBtn.clicked.connect(self.calcTotal)
```

- **Bước 5**: Thêm Code cho hàm main

```
if __name__ == "__main__":
    import sys # Có thể khai báo đầu chương trình
    app = QtWidgets.QApplication(sys.argv)
    MainWindow = QtWidgets.QMainWindow()
    ui = Ui_MainWindow()
    ui.setupUi(MainWindow)
    MainWindow.show()
    sys.exit(app.exec_())
```