

# 사용자 인터페이스

# 자주 발생하는 오류 (1)

## ■ pip?

- pip는 python으로 작성된 패키지 라이브러리를 관리해주는 시스템
- 다양한 패키지 설치방법
  - pip install [package\_name]

※ 코드 실행 중, 만약 다음과 같은 오류 메시지가 나온다면?

<error message>

```
import cv2  
ModuleNotFoundError: No module named 'cv2'
```

→ 'cv2' 패키지를 찾을 수 없다는 것을 의미

### <해결방법1>

1. Pycharm->Settings->project->python interpreter
2. python interpreter: AI\_env 선택 유무 확인

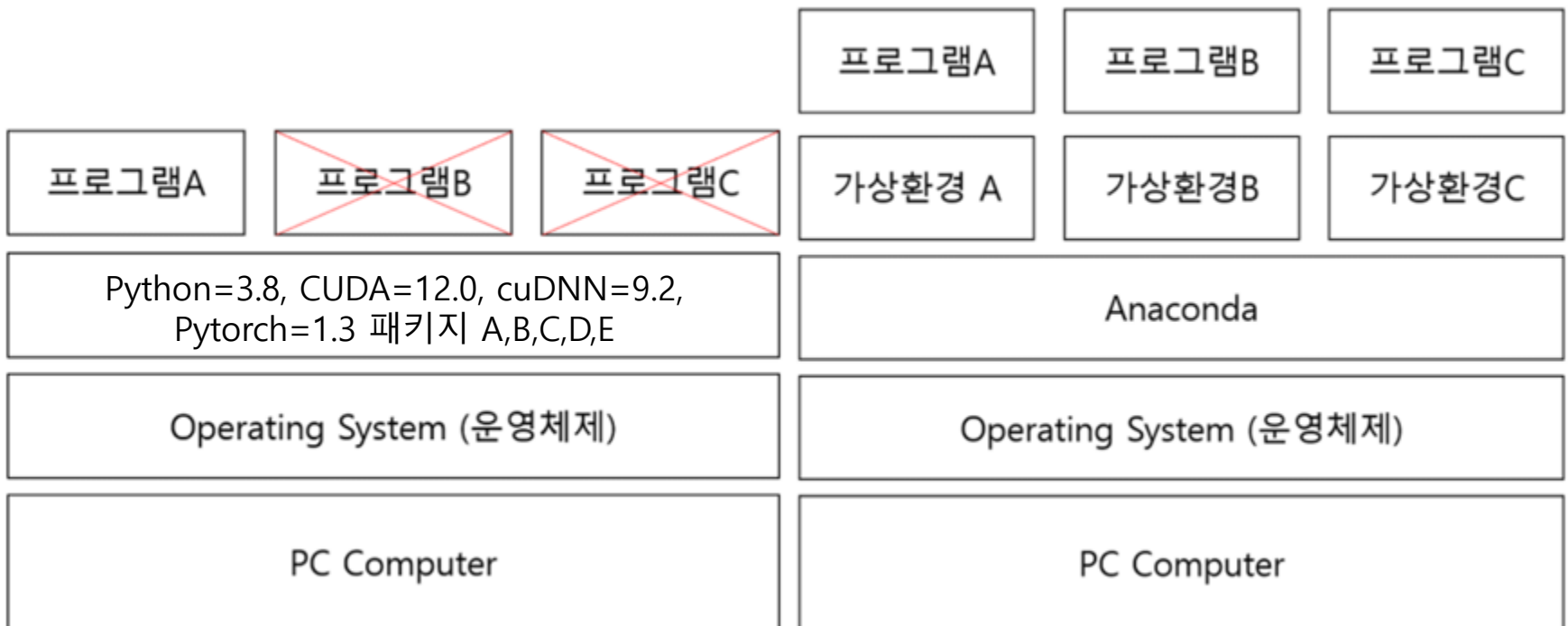
### <해결방법2>

1. Anaconda prompt 실행
2. conda activate [가상환경 이름]
3. pip install opencv-python

# 아나콘다와 가상환경

## ■ conda?

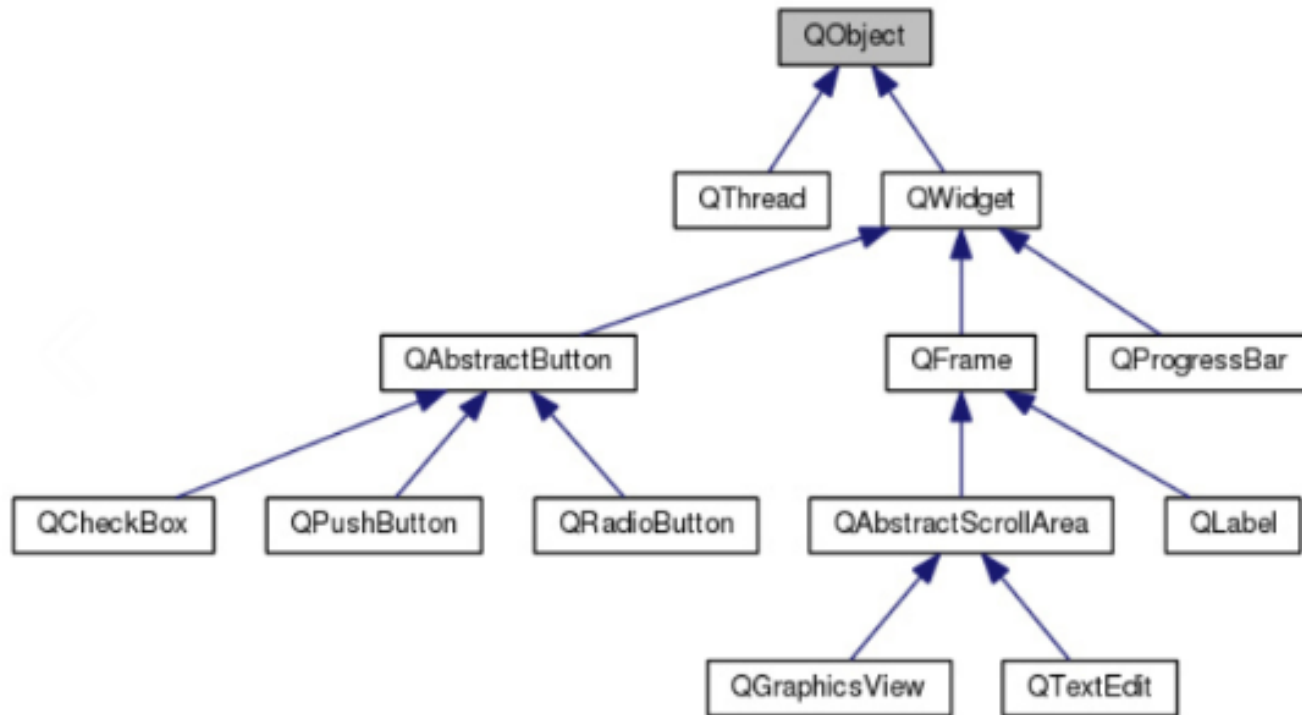
- Anaconda에서 제공하는 패키지 관리자
- 패키지 설치 시 자동으로 기존 설치되어 있는 패키지들의 호환성을 체크해주는 장점
- Python, R, Ruby, Lua, Scala, Java, Javascript, C/C++, FORTRAN 등 많은 언어들의 의존성, 패키지, 환경을 관리
- 주로 가상환경 생성과 패키지 관리에 사용



# PySide6

## ■ PySide6?

- Python의 User Interface 개발을 위한 패키지 중 하나로 Qt를 기반으로 한 오픈소스  
※ Qt? – GUI 제작 크로스 플랫폼 프레임워크



[QWidget의 계층구조]

# PySide6

## ■ PySide6 설치

- pip install pyside6

```
Anaconda Prompt
(AI_env) C:\Users\USER>pip install pyside6
Collecting pyside6
  Downloading PySide6-6.6.3.1-cp38-abi3-win_amd64.whl.metadata (5.5 kB)
Collecting shiboken6==6.6.3.1 (from pyside6)
  Downloading shiboken6-6.6.3.1-cp38-abi3-win_amd64.whl.metadata (2.6 kB)
Collecting PySide6-Essentials==6.6.3.1 (from pyside6)
  Downloading PySide6_Essentials-6.6.3.1-cp38-abi3-win_amd64.whl.metadata (3.8 kB)
Collecting PySide6-Addons==6.6.3.1 (from pyside6)
  Downloading PySide6_Addons-6.6.3.1-cp38-abi3-win_amd64.whl.metadata (4.2 kB)
  Downloading PySide6-6.6.3.1-cp38-abi3-win_amd64.whl (520 kB)
    ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 520.6/520.6 kB 5.4 MB/s eta 0:00:00
  Downloading PySide6_Addons-6.6.3.1-cp38-abi3-win_amd64.whl (111.7 MB)
    ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 111.7/111.7 MB 8.0 MB/s eta 0:00:00
  Downloading PySide6_Essentials-6.6.3.1-cp38-abi3-win_amd64.whl (77.3 MB)
    ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 77.3/77.3 MB 14.2 MB/s eta 0:00:00
  Downloading shiboken6-6.6.3.1-cp38-abi3-win_amd64.whl (1.1 MB)
    ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.1/1.1 MB 70.5 MB/s eta 0:00:00
Installing collected packages: shiboken6, PySide6-Essentials, PySide6-Addons, pyside6
Successfully installed PySide6-Addons-6.6.3.1 PySide6-Essentials-6.6.3.1 pyside6-6.6.3.1 shiboken6-6.6.3.1
(AI_env) C:\Users\USER>
```

# 1. PySide6: Hello World

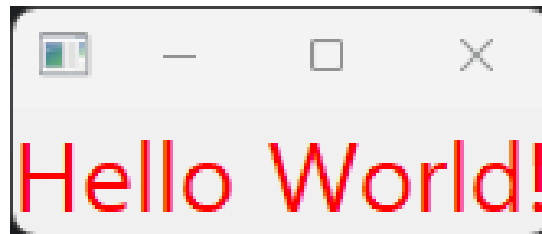
(ex2-1)

## ■ QtWidgets 애플리케이션 만들기

- QApplication – 위젯의 구동을 위한 초기화를 수행
  - Qt는 command line에서 인수(argument)를 받을 수 있으므로, 명령QApplication 객체에 인수를 전달할 수 있음 QApplication(**sys.argv**)
- QLabel – 텍스트나 이미지를 표시하는데 사용

```
import sys
from PySide6.QtWidgets import QApplication, QLabel

app = QApplication(sys.argv)
label = QLabel("Hello World!")
# label = QLabel("<font color=red size=40>Hello World!</font>")
label.show()
app.exec()
```



# sys 사용 용도

- 프로그램 실행시 인자 값을 받고 싶은 경우 sys.argv 사용
  - ex) python ex\_pyside6\_test World 2024

```
import sys
print(f"sys: {sys.argv}")
print(f"Hello {sys.argv[1]} & {sys.argv[2]}")
```

## 2. 버튼 만들기

### ■ 기본버튼 만들기

- 윈도우에 기본적인 버튼

### ■ QPushButton([label])

- 클릭으로 동작하는 버튼 생성함수
- 인자: 버튼의 레이블을 나타내는 문자열

```
import sys
from PySide6.QtWidgets import QApplication, QPushButton

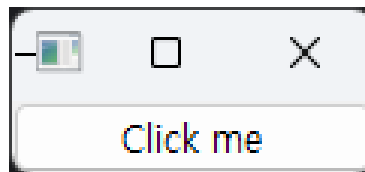
# Create the Qt Application
app = QApplication(sys.argv)

# Create a button
button = QPushButton("Click me")

# Show the button
button.show()

# Run the main Qt loop
app.exec()
```

버튼의 레이블





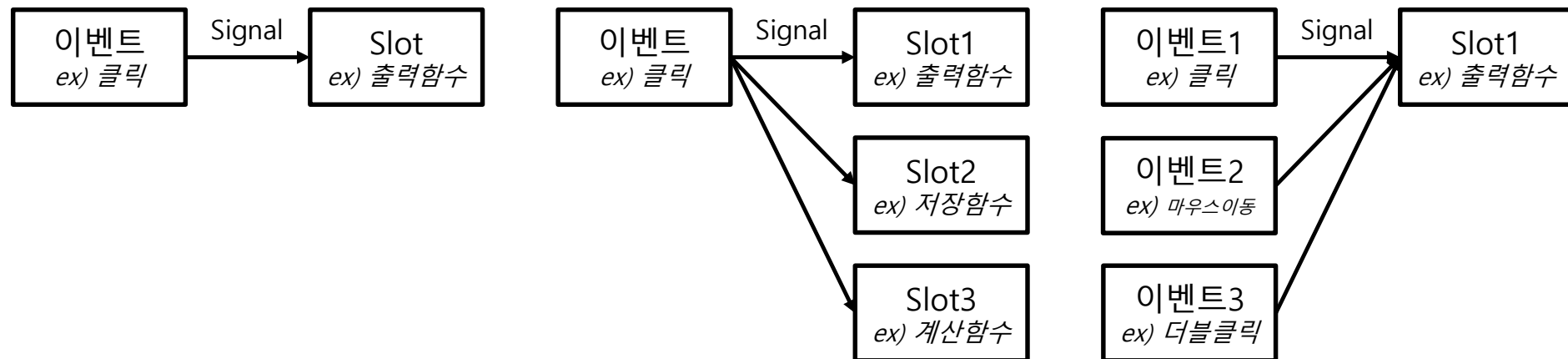
## 2. 버튼 만들기 (기능추가)

### ■ 신호(Signal)

- 신호는 프로그램 동작 과정에서 어떠한 일이 일어 났을 때 widget으로부터 발생하는 알림  
(※ 어떠한 일이란? 버튼 클릭 / 마우스 클릭 / 마우스 움직임 등과 같은 event)

### ■ 슬롯(Slot)

- 슬롯은 이벤트에 의해 발생한 시그널에 대한 응답으로 실행되는 함수(function)
- 슬롯은 Qt에서 사용하는 신호 수신기로서, 신호 발생 시 동작되는 함수를 의미
  - 원하는 만큼 많은 신호를 단일 슬롯에 연결할 수 있으며, 신호는 필요한 만큼 많은 슬롯에 연결할 수 있음

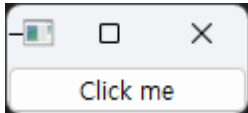


## 2. 버튼 만들기 (기능추가)

(ex2-2)

### ■ 기능이 추가된 버튼 만들기

- 신호(signal/event)와 슬롯을 처리하는 방법으로, 버튼을 클릭할 때마다 "Button clicked, Hello" 메시지를 파이썬 콘솔에 기록하는 기능을 구현



```
Button clicked, Hello!  
Button clicked, Hello!  
Button clicked, Hello!  
Button clicked, Hello!
```

```
import sys  
from PySide6.QtWidgets import QApplication, QPushButton  
from PySide6.QtCore import Slot  
  
def say_hello():  
    print("Button clicked, Hello!")  
  
# Create the Qt Application  
app = QApplication(sys.argv)  
  
# Create a button  
button = QPushButton("Click me")  
  
# Connect the button to the function  
button.clicked.connect(say_hello)  
  
# Show the button  
button.show()  
# Run the main Qt loop  
app.exec()
```

버튼 이벤트의 함수 연결

### 3. MainWindow 어플리케이션 생성

(ex2-3)

#### ■ 메인화면 생성

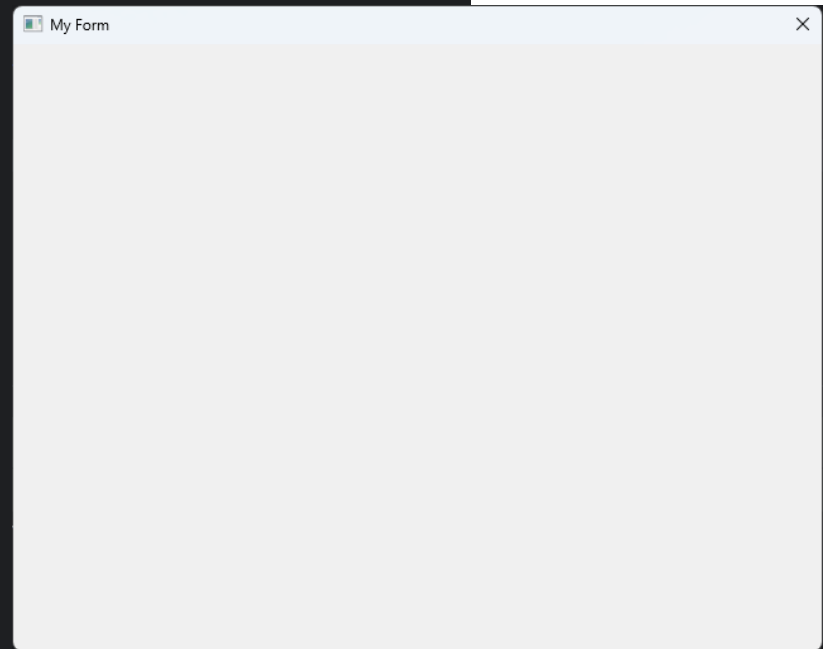
- 기본 위젯으로 간단한 대화 상자를 만드는 코드

```
import sys
from PySide6.QtWidgets import QApplication, QMainWindow

class Form(QMainWindow):

    def __init__(self, parent=None):
        super(Form, self).__init__(parent)
        self.setWindowTitle("My Form")
        self.setGeometry(0, 0, 800, 500)

if __name__ == '__main__':
    # Create the Qt Application
    app = QApplication(sys.argv)
    # Create and show the form
    form = Form()
    form.show()
    # Run the main Qt loop
    sys.exit(app.exec())
```



## 4. 레이아웃 정렬

### ■ QVBoxLayout()

- 위젯을 구성하는데 도움을 주는 레이아웃 함수
- 생성되는 요소(widget)들을 **수직으로** 배치

```
layout = QVBoxLayout()  
layout.addWidget(self.button1)  
layout.addWidget(self.button2)
```

### ■ QHBoxLayout()

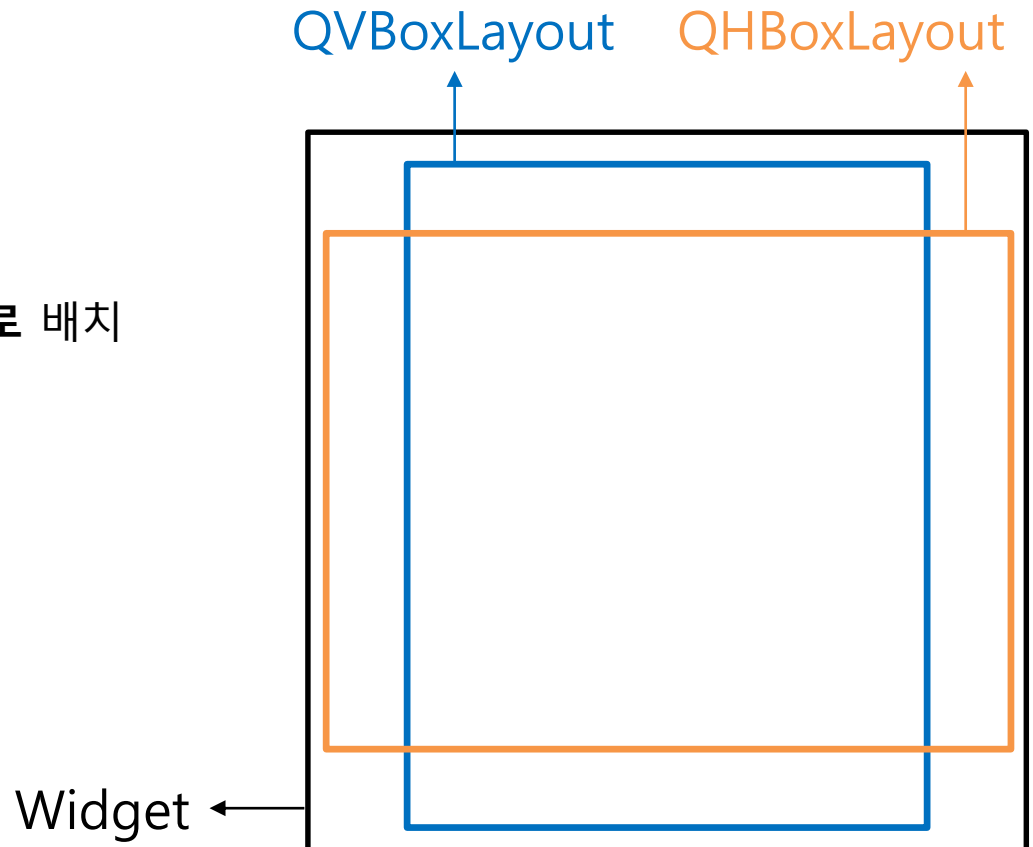
- 생성되는 요소(widget)들을 **수평으로** 배치

```
layout = QHBoxLayout()  
layout.addWidget(self.button1)  
layout.addWidget(self.button2)
```

### ■ QWidget()

- 빈 창 위젯을 생성

```
widget = QWidget(self)  
widget.setLayout(layout)
```



## 4. 레이아웃 정렬

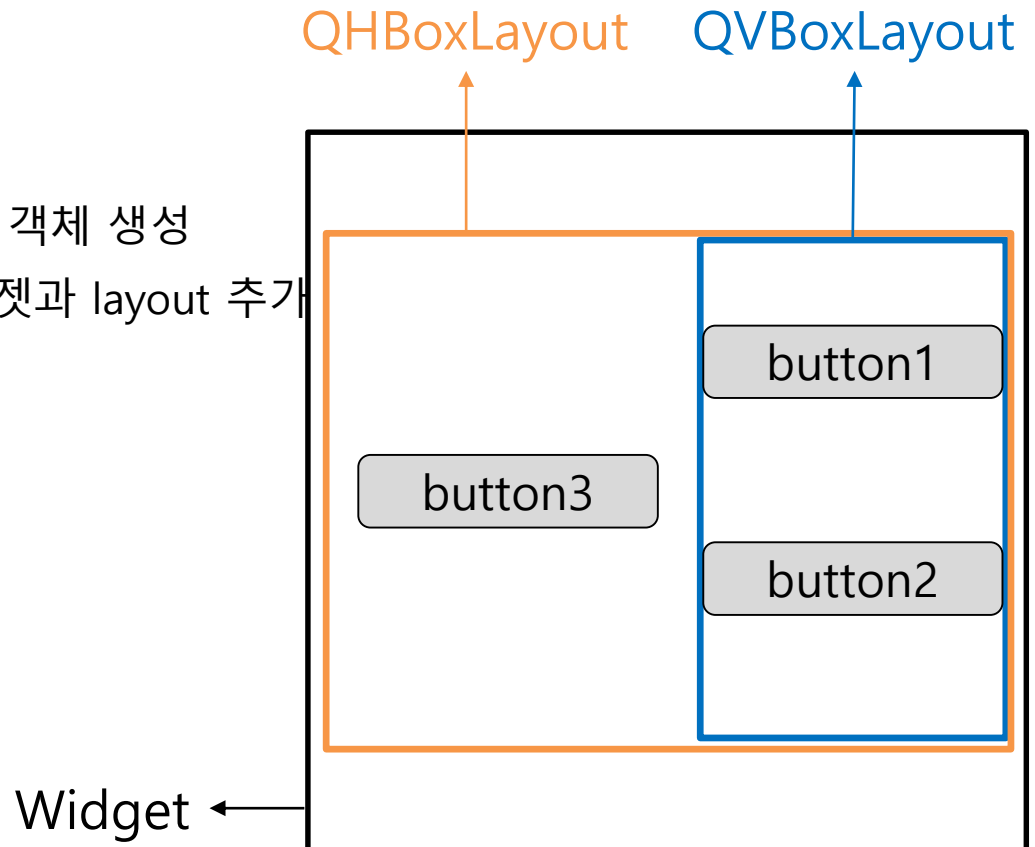
### ■ 수평 정렬에 수직 정렬 넣기

- 1) 수직 정렬을 돕는 Layout 클래스 객체 생성
- 2) 수직 정렬 Layout에 들어가는 위젯 추가

```
layout_vert = QVBoxLayout()  
layout_vert.addWidget(self.button1)  
layout_vert.addWidget(self.button2)
```

- 3) 수평 정렬을 돕는 Layout 클래스 객체 생성
- 4) 수평 정렬 Layout에 들어가는 위젯과 layout 추가

```
layout_hori = QHBoxLayout()  
layout_hori.addWidget(self.button3)  
layout_hori.addLayout(layout_vert)
```

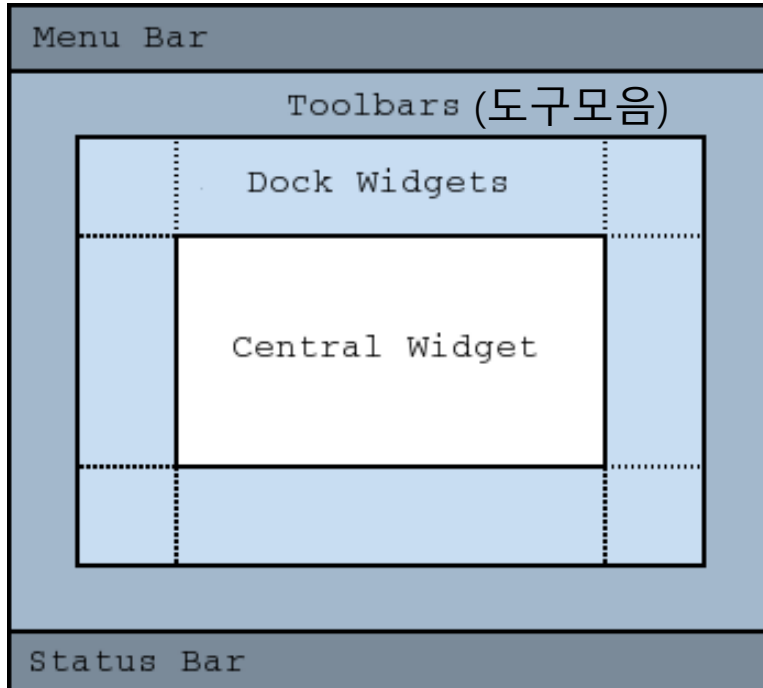


## 4. 레이아웃 정렬

### ■ QMainWindow의 레이아웃

- Central Widget은 메인 윈도우의 중앙부를 나타냄
- `setCentralWidget([위젯명])` 을 통하여 위젯을 화면 중앙에 배치

```
widget = QWidget(self)
widget.setLayout(layout)
self.setCentralWidget(widget)
```



## 4. 레이아웃 정렬

(ex2-4)

### ■ 레이아웃 예제

```
import sys
from PySide6.QtWidgets import (QPushButton, QApplication,
                                QWidget, QVBoxLayout, QMainWindow)

class Form(QMainWindow):
    def __init__(self, parent=None):
        super(Form, self).__init__(parent)

        # 3개의 버튼 객체를 생성

        # 수직으로 버튼을 배치

        # 위젯을 중앙에 배치

        # 3개의 버튼이 각각 버튼번호를 출력하는 Slot 만들기
        self.button1.clicked.connect(self.button1_slot)

        # Greets the user
        def button1_slot(self):
            print(f"Button 1")
```

```
if __name__ == '__main__':
    # Create the Qt Application
    app = QApplication(sys.argv)
    # Create and show the form
    form = Form()
    form.show()
    # Run the main Qt loop
    sys.exit(app.exec())
```

## 5. 문자열 입력 받기

### ■ QLineEdit(str)

- 사용자가 문자열을 입력할 수 있는 편집기를 생성
- 인자: 생성 초기에 출력시키는 문자열

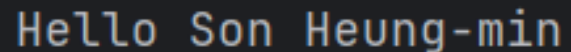
```
self.edit = QLineEdit( " Write my name here " )
```



### ■ 문자열 출력

- QLineEdit의 text()함수를 이용하여 입력 받은 문자열을 출력하는 함수를 정의

```
def greetings(self):  
    print(f"Hello {self.edit.text()}")
```



```
Hello Son Heung-min
```



## 5. 문자열 입력 받기

(ex2-5)

### ■ 대화상자(Dialog) 생성

- 입력 문자열을 터미널에 출력하는 대화상자 코드

```
import sys
from PySide6.QtWidgets import (QLineEdit, QPushButton,
    QApplication, QWidget, QVBoxLayout, QMainWindow)
```

```
class Form(QMainWindow):
```

```
    def __init__(self, parent=None):
        super(Form, self).__init__(parent)
```

```
        # QLineEdit 객체 생성
```

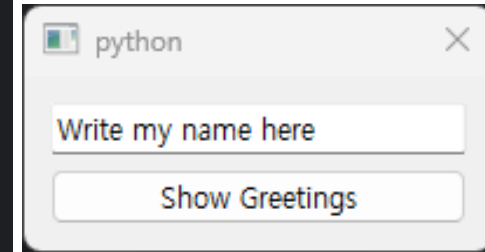
```
        # 버튼 객체 생성
```

```
        # 수직으로 QLineEdit과 버튼 배치
```

```
        # 위젯 생성 및 전체 레이아웃을 메인 윈도우 중앙에 배치
```

```
        # 버튼 클릭 시 greetings 함수를 실행하도록 신호와 함수 연결
```

```
    def greetings(self):
        print(f"Hello {self.edit.text()}")
```



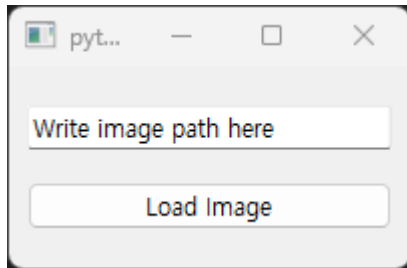
```
if __name__ == '__main__':
    # Create the Qt Application
    app = QApplication(sys.argv)
    # Create and show the form
    form = Form()
    form.show()
    # Run the main Qt loop
    sys.exit(app.exec())
```

## 6. 영상 불러오기

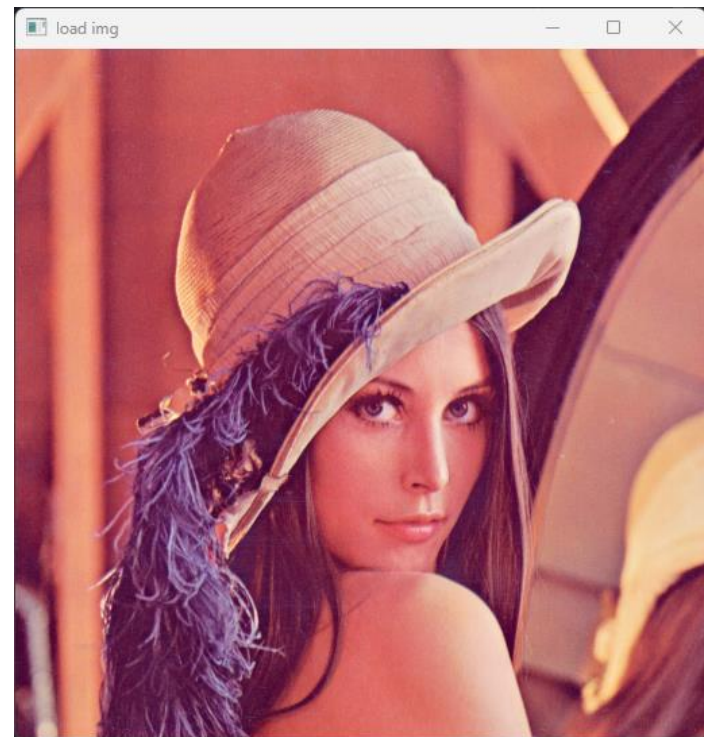
(ex2-6)

### ■ LineEdit과 OpenCV를 사용하여 영상 불러오기

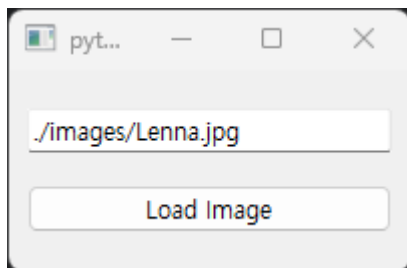
#### 1. 초기 실행 화면



#### 3. Load Image 버튼 실행 후 출력 화면



#### 2. 이미지 경로 입력 후 Load Image 버튼 실행



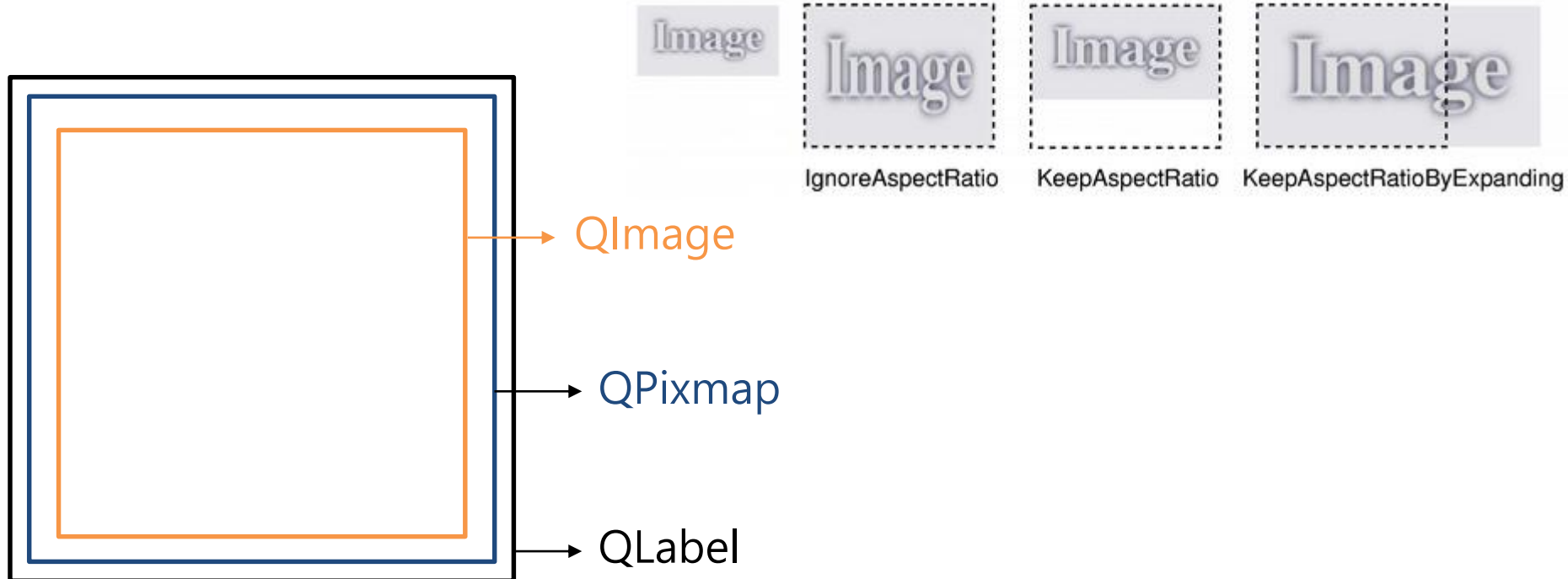
## 7. 메인 윈도우에 영상 출력하기

### ■ QPixmap

- 이미지를 화면에 표시하도록 설계 및 최적화된 클래스

### ■ QImage

- 픽셀 데이터에 직접 접근하여 이미지의 변환을 가능하게 하는 클래스
- QImage(data, width, height, bytesPerLine, image\_format)
- QImage.scaled(가로길이, 세로길이, 가로세로 비율모드)



## 7. 메인 윈도우에 영상 출력하기

(ex2-7)

```
import sys
import cv2
from PySide6.QtCore import Qt
from PySide6.QtGui import QImage, QPixmap
from PySide6.QtWidgets import QApplication, QLabel, QMainWindow
```

```
class Window(QMainWindow):
```

```
    def __init__(self):
        super().__init__()
        self.label = QLabel("Image here")
        self.label.setFixedSize(640, 480)
        self.setCentralWidget(self.label)
```

```
    img = cv2.imread(f"./images/Lenna.jpg", cv2.IMREAD_COLOR)
    img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
```

```
    h, w, ch = img.shape
```

```
    img = QImage(img.data, w, h, ch * w, QImage.Format_RGB888)
    scaled_img = img.scaled(640, 480, Qt.KeepAspectRatio)
```

```
    self.label.setPixmap(QPixmap.fromImage(scaled_img))
    print("update image")
```

```
if __name__ == "__main__":
    app = QApplication()
    w = Window()
    w.show()
    sys.exit(app.exec())
```

## 7. 메인 윈도우에 영상 출력하기

