

삼성 청년 SW 아카데미

임베디드 GUI

<알림>

본 강의는 삼성 청년 SW아카데미의 컨텐츠로
보안서약서에 의거하여
강의 내용을 어떠한 사유로도 임의로 복사,
촬영, 녹음, 복제, 보관, 전송하거나
허가 받지 않은 저장매체를
이용한 보관, 제3자에게 누설, 공개,
또는 사용하는 등의 행위를 금합니다.

Day3-1. QTimer와 QThread

챕터의 포인트

- GUI App 개발과 멀티 태스킹
- QTimer
- QThread

GUI App 개발과 멀티 태스킹

GUI App은 멀티 태스킹이 가능해야 한다.

- **다운로드 버튼을 눌렀더니, 프로그램이 멈춘다?!**
→ User는 이 일을 잊지 않을 것입니다.
- **로그를 출력하는 동안, 프로그램이 멈춘다?!**
→ User는 이 일을 잊지 않을 것입니다.

GUI App은 한 가지 동작만 수행해서는 안된다.

Thread와 Timer 를 이용해, 멀티 태스킹이 가능하도록 할 수 있다.

- Qt는 다양한 플랫폼을 지원하기 때문에, Thread 와 Timer 를 모두 구현 가능하다.

우리는 Qt의 Thread 와 Timer 기법을 익혀야만 한다!

App은 두 가지 작업이 동시에 동작해야 한다.

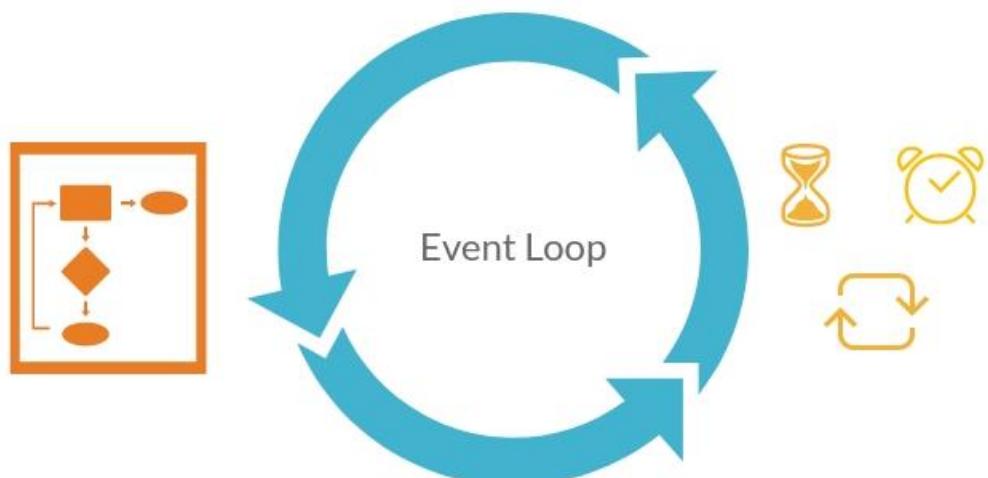
- 1. GUI 화면을 띄우는 작업 (UI)
- 2. 프로그램 내 이벤트 처리 작업 (Background)

Thread 와 Timer 모두 마찬가지

- User 가 App을 동작 시키면 이벤트가 발생한다.
- 발생된 이벤트는 Background 에서 처리한다.
- UI 작업은 계속 구동 중이며, User는 다른 이벤트를 발생 시킬 수 있다.
- Background 에서 처리한 결과를 다시 UI에 표시한다.

Qt App 내부에는 이벤트 감시자인 이벤트 loop 가 존재한다.

- 1) App 동작, GUI 화면 출력
- 2) App 내부에는 이벤트를 감시하기 위한 polling이 동작 중
- 3) User / OS 에 의해 App에 이벤트 발생
- 4) 이벤트 큐로 이벤트가 Push 됨
- 5) 이벤트 Loop 가 이벤트를 감지하고 적절한 Signal 발생



멀티 태스킹이 되지 않는다면,
이벤트 Loop가 멈추게 되어,
App이 멈춘 것처럼 보인다.

QTimer

특정 시간 주기로 Signal을 보내는 객체

- QBasicTimer 보다 정밀 <https://doc.qt.io/qtforpython-6/PySide6/QtCore/QTimer.html>
- 정밀도 : 1ms
 - OS와 H/W에 따라 다른 결과를 나타낼 수 있다.

Accuracy and Timer Resolution

The accuracy of timers depends on the underlying operating system and hardware. Most platforms support a resolution of 1 millisecond, though the accuracy of the timer will not equal this resolution in many real-world situations.

- 멀티 Thread App에서 활용
 - 여러 쓰레드에서 타이머를 독립적으로 생성, 관리해야 한다.

In multithreaded applications, you can use `QTimer` in any thread that has an event loop. To start an event loop from a non-GUI thread, use `exec()`. Qt uses the timer's `thread affinity` to determine which thread will emit the `timeout()` signal. Because of this, you must start and stop the timer in its thread; it is not possible to start a timer from another thread.

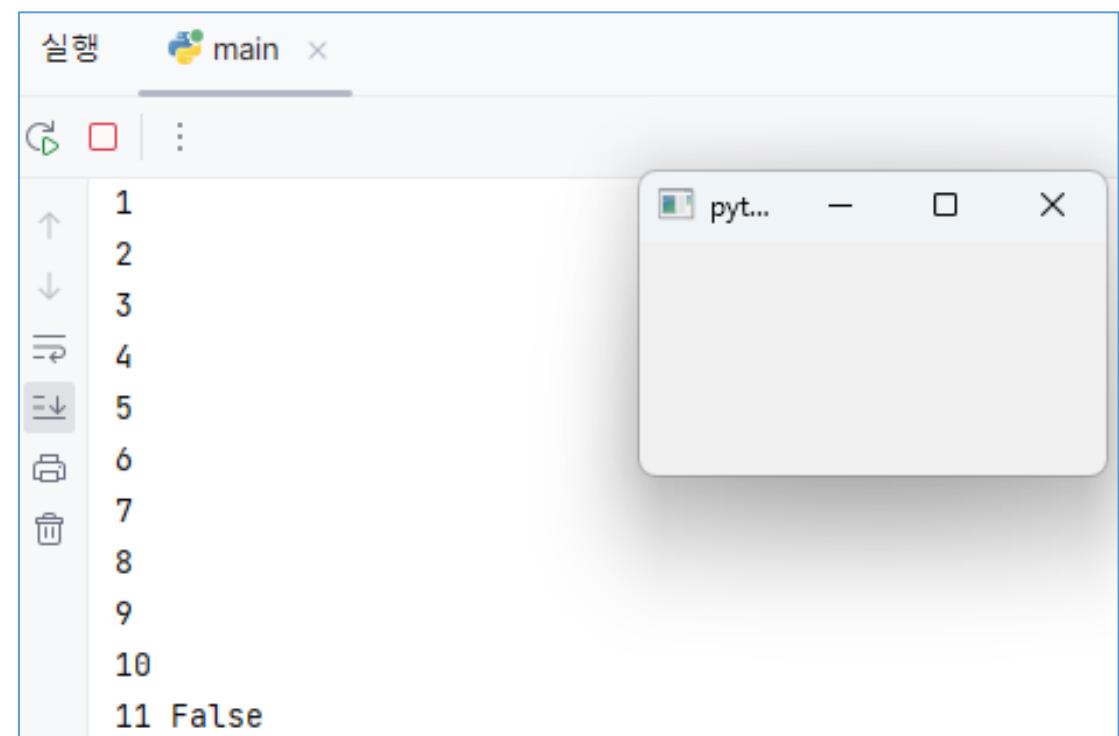
Timer API

- `start()` : 타이머 시작
- `stop()` : 타이머 중지
- `setInterval()` : ms 단위로 시간 단위 설정
- `interval()` : 현재 세팅된 interval 값 리턴
- `isActive()` : Timer가 동작중인지 True / False

```
8  def main(self):  
9      self.tm = QTimer()  
10     self.tm.setInterval(500)  
11     self.tm.timeout.connect(self.run)  
12     self.tm.start()  
13     print(self.tm.interval(), self.tm.isActive())  
14  
15     num = 0  
16     1개의 사용 위치  
17     def run(self):  
18         self.num += 1  
19         if self.num > 10:  
20             self.tm.stop()  
21             print(self.num, self.tm.isActive())  
22             return  
23             print(self.num)
```

<https://gist.github.com/hoconoco/6c06f8cdfcf8d8b1520bd24c099c944b>

0.5초에 한번씩 print를 한 뒤, 10이 넘으면 종료된다.



QThread

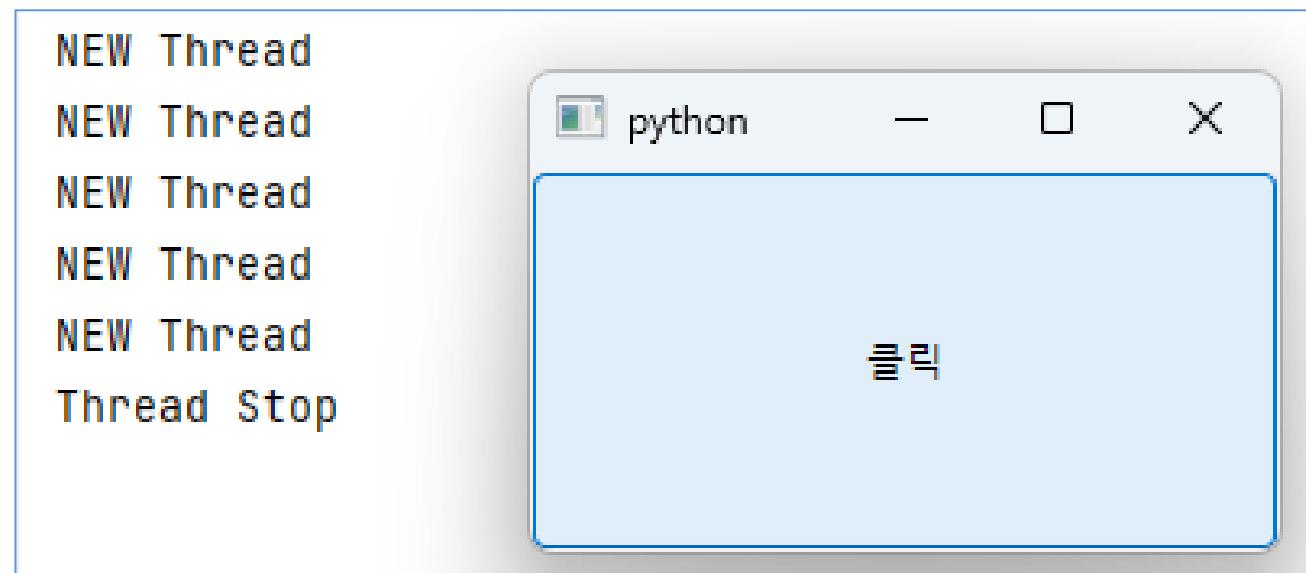
멀티 스레딩을 지원하기 위한 클래스이다.

- 스레드 생성 / 관리
- 스레드 간 통신
- 스레드 종료와 자원 정리
- 이벤트 루프 실행

<https://doc.qt.io/qtforpython-6/PySide6/QtCore/QThread.html>

Thread class 생성 후 관련 함수를 재정의해서 사용한다.

- `.start()` : 스레드 시작, Thread Class 내 `run()` 실행
- `.start()` : 스레드 정지, Thread Class 내 `stop()` 실행
- `run()` : 스레드 동작 시 실행되는 함수
- `stop()` : 스레드 정지 시 실행되는 함수



<https://gist.github.com/hoconoco/3a19b538607d6bd8491deb92175695a3>

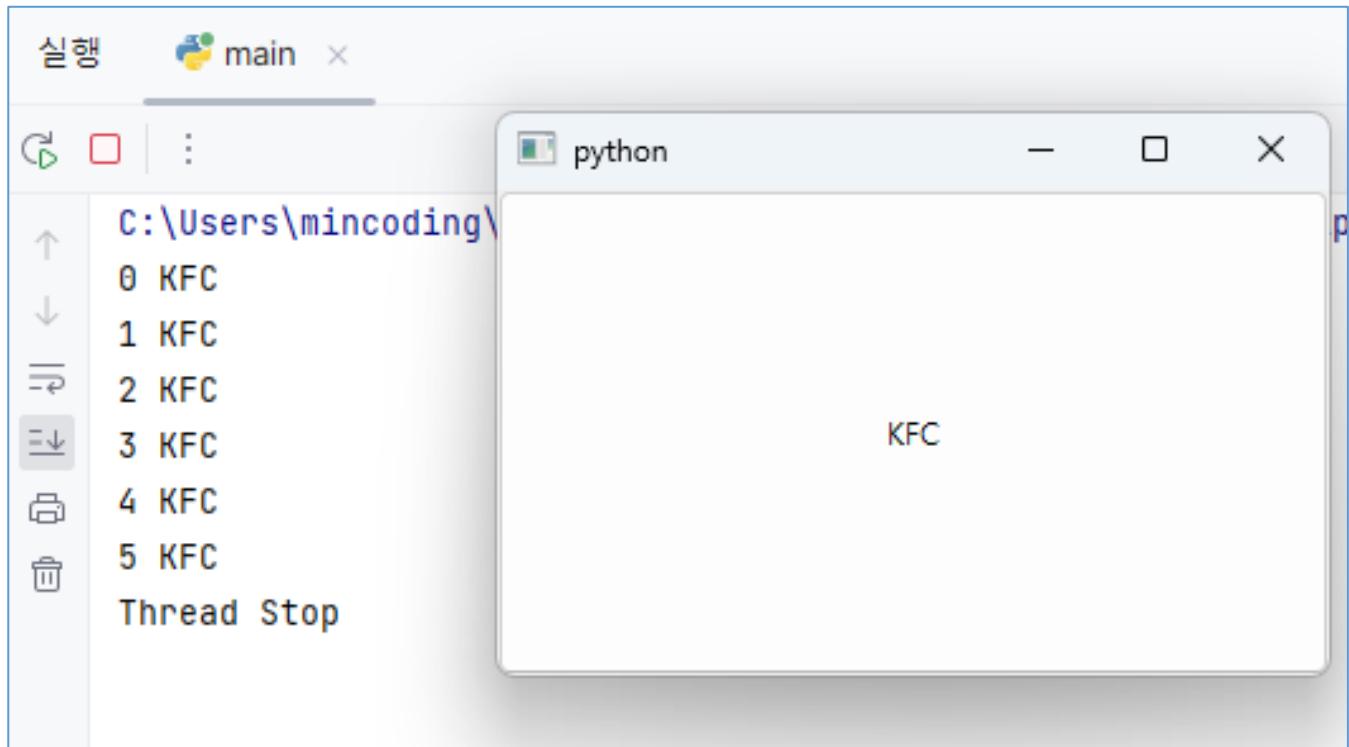
- **th.start() → MyThread 의 run() 호출**
 - NEW Thread 를 1초에 한번씩 출력한다.
- **th.stop() → MyThread 의 stop() 호출**
 - Thread 동작을 멈추고, Thread Stop 출력한다.

```
5  class MyThread(QThread):  
6      def __init__(self):  
7          super().__init__()  
8  
9          flag = False  
10     @  
11     def run(self):  
12         self.flag = True  
13         while self.flag:  
14             print("NEW Thread")  
15             sleep(1)  
16  
17     def stop(self):  
18         self.flag = False  
         print("Thread Stop")
```

```
25  
26  
27         self.btn = QPushButton("클릭")  
28         self.btn.clicked.connect(self.thread_stop)  
29  
30  
31  
32         self.th = MyThread()  
33         self.th.start()  
34  
35         1개의 사용 위치  
36         def thread_stop(self):  
             self.th.stop()
```

Thread 내 신호 주고 받기

- `MyThread() → MyApp()` 으로 2개의 data를 보낸다.
- 나만의 signal 을 만들어서 slot 함수를 호출한다.



<https://gist.github.com/hoconoco/a2e58e0ef00db851e1d55091501130b0>

- **Signal()**
 - 나만의 signal 객체 생성, signal로 전송할 data 개수와 type도 설정
- **.emit()**
 - signal 전송하는 API

```
9  class MyThread(QThread):  
10     mySignal = Signal(int, str)  
11  
12     def __init__(self):  
13         super().__init__()  
14  
15     flag = False  
16  
17     def run(self):  
18         self.flag = True  
19         while self.flag:  
20             self.mySignal.emit(10, "KFC")  
21             sleep(1)
```

```
33     def main(self):  
34         self.btn = QPushButton("클릭")  
35         self.btn.clicked.connect(self.thread_stop)  
36         self.setCentralWidget(self.btn)  
37  
38         self.th = MyThread()  
39         self.th.start()  
40         self.th.mySignal.connect(self.settingUI)  
41  
42     def settingUI(self, val, msg):  
43         self.btn.setText(msg)  
44         print(val, msg)
```

Day3-2. 다양한 Widget 사용

챕터의 포인트

- **QTableWidget**
- **QProgressBar**
- **QDial**
- **QSlider**
- **QLCDNumber**
- **Widget 의 리스트**

QTableWidget

표 형식으로 된 Widget

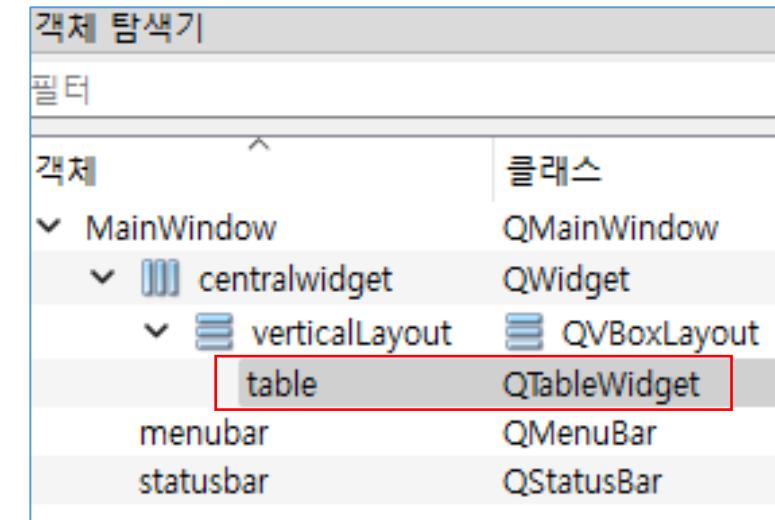
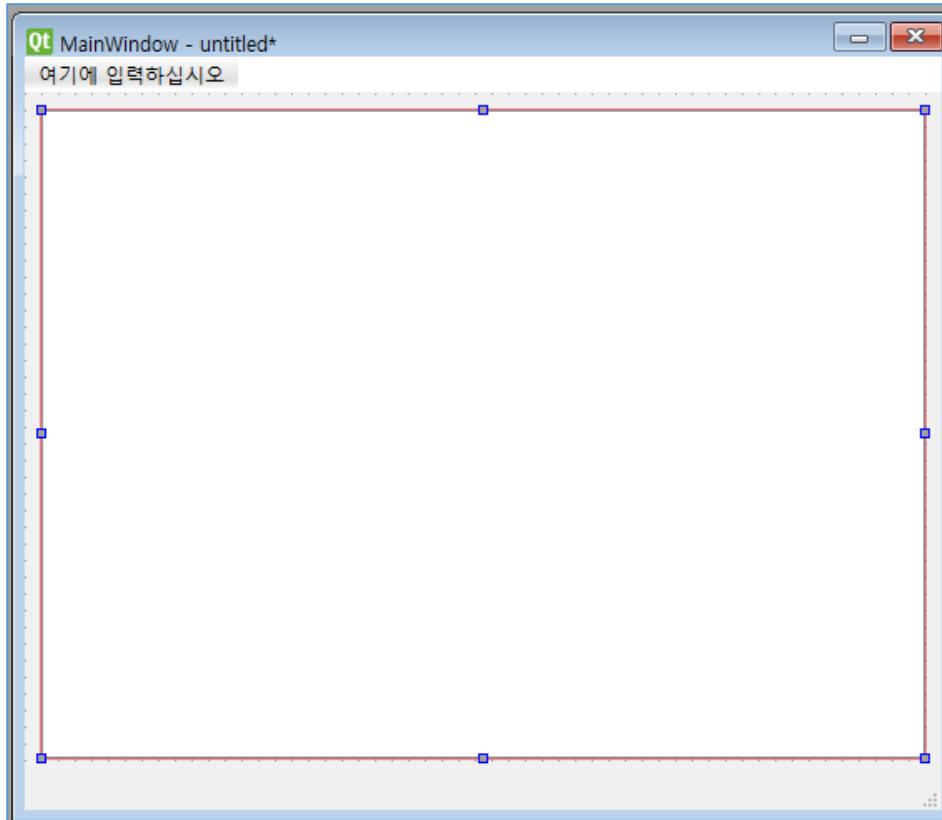
- 2차원 데이터를 표현할 때 사용 (엑셀과 비슷)
- 데이터 관리 업무용 프로그램을 제작할 때 사용

	id	pass
1		
2		
3		
4		
5		

	A	B	C	D	E	F
1						
2						
3						
4						
5						
6						
7						
8						
9						
10						
11						
12						

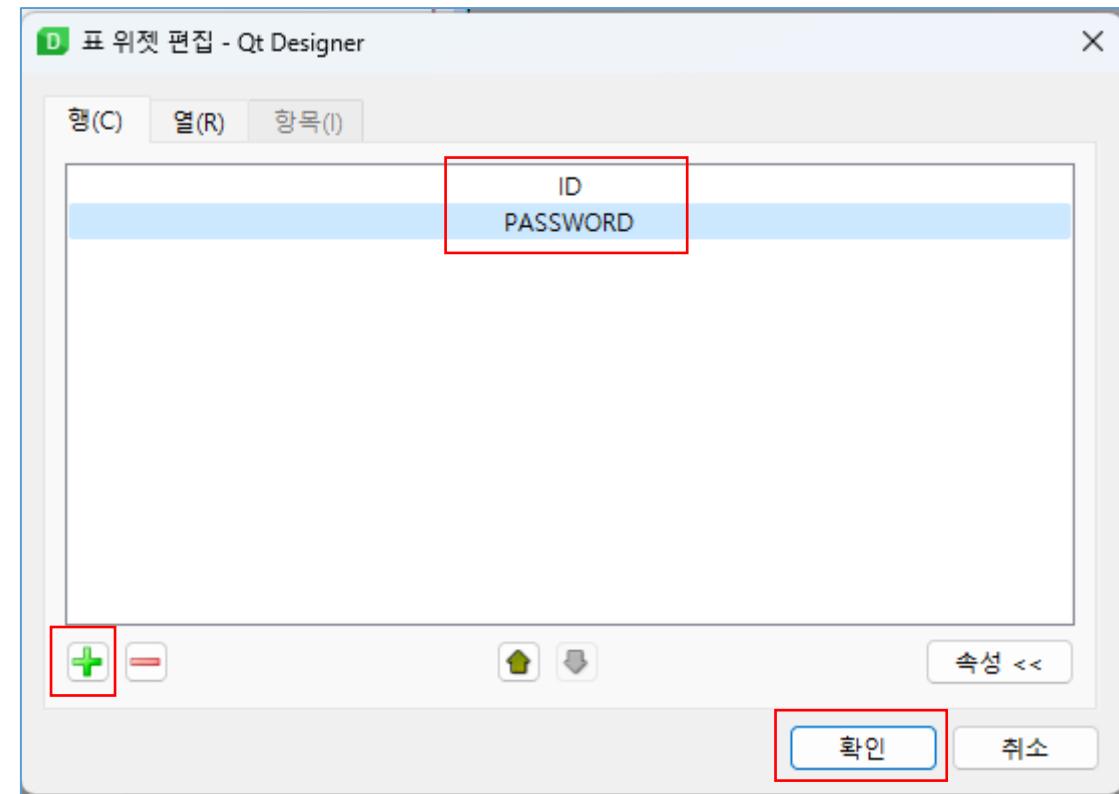
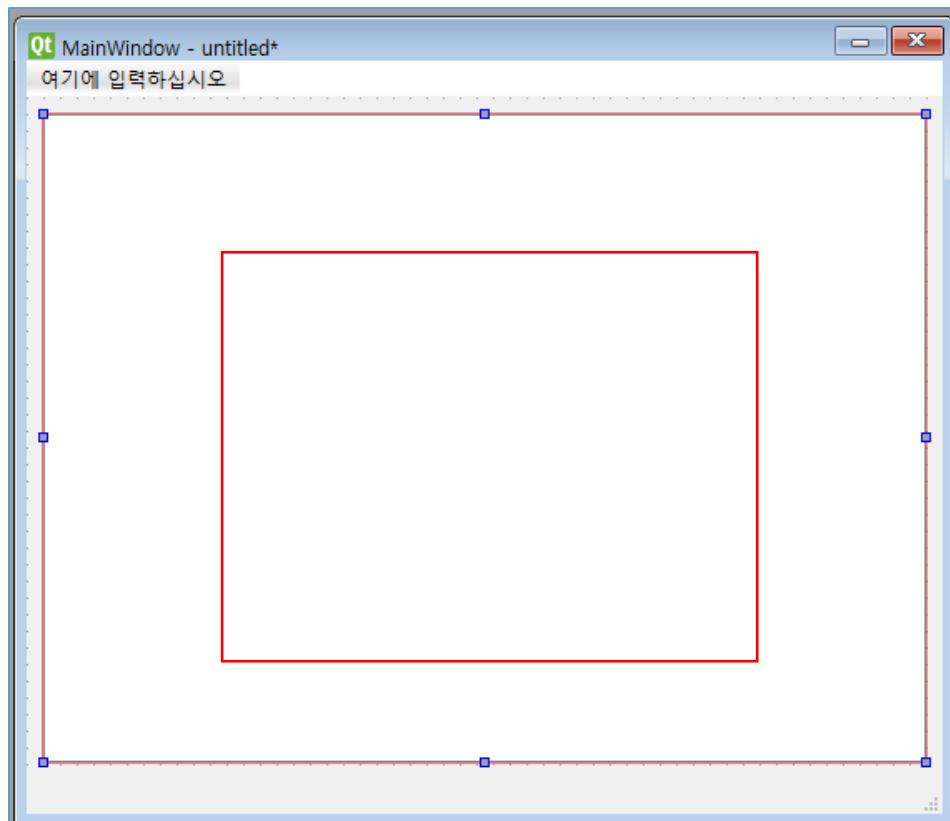
QTableWidget을 배치하자.

- CentralWidget에 수직배치를 넣고, QTableWidget을 배치한다.
- QTableWidget의 이름을 수정한다.



QTableWidget 더블 클릭

- 행(Columns) : ID, PASSWORD 추가



속성 편집기 이용해서 row 개수 정하기

- rowCount : 5
- horizontalHeaderDefaultSectionSize : 200

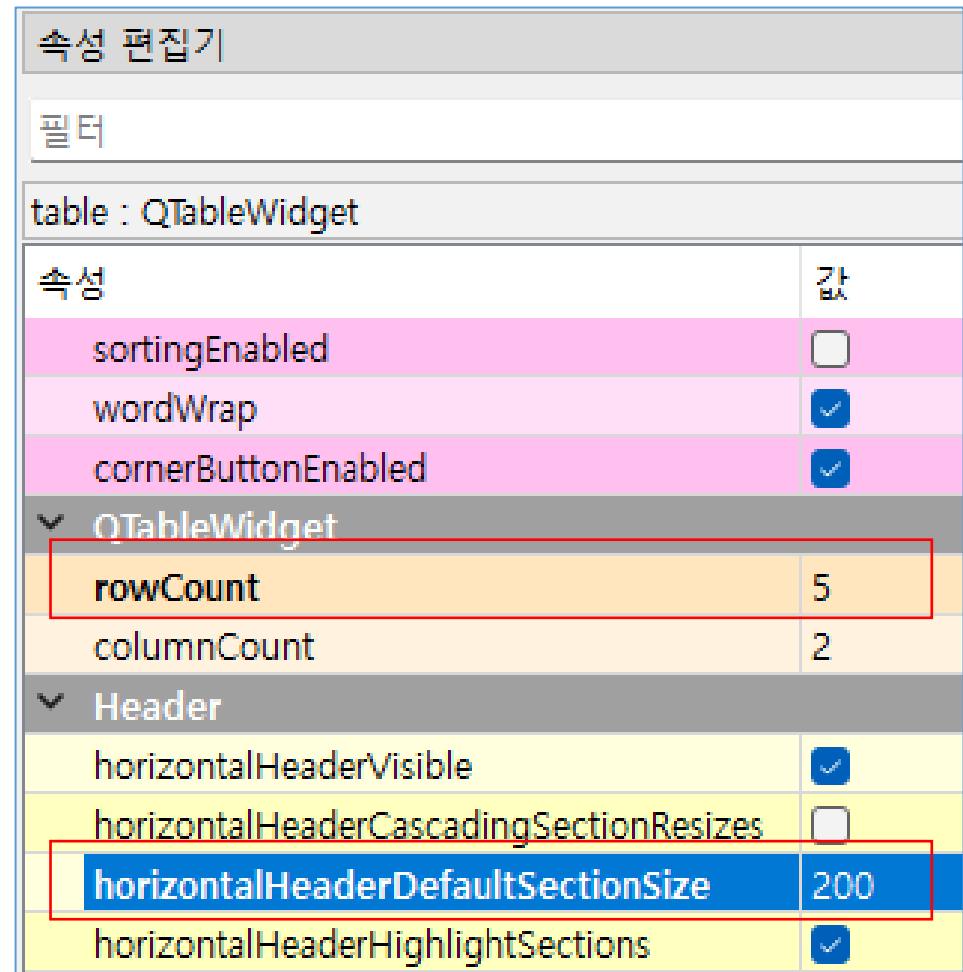
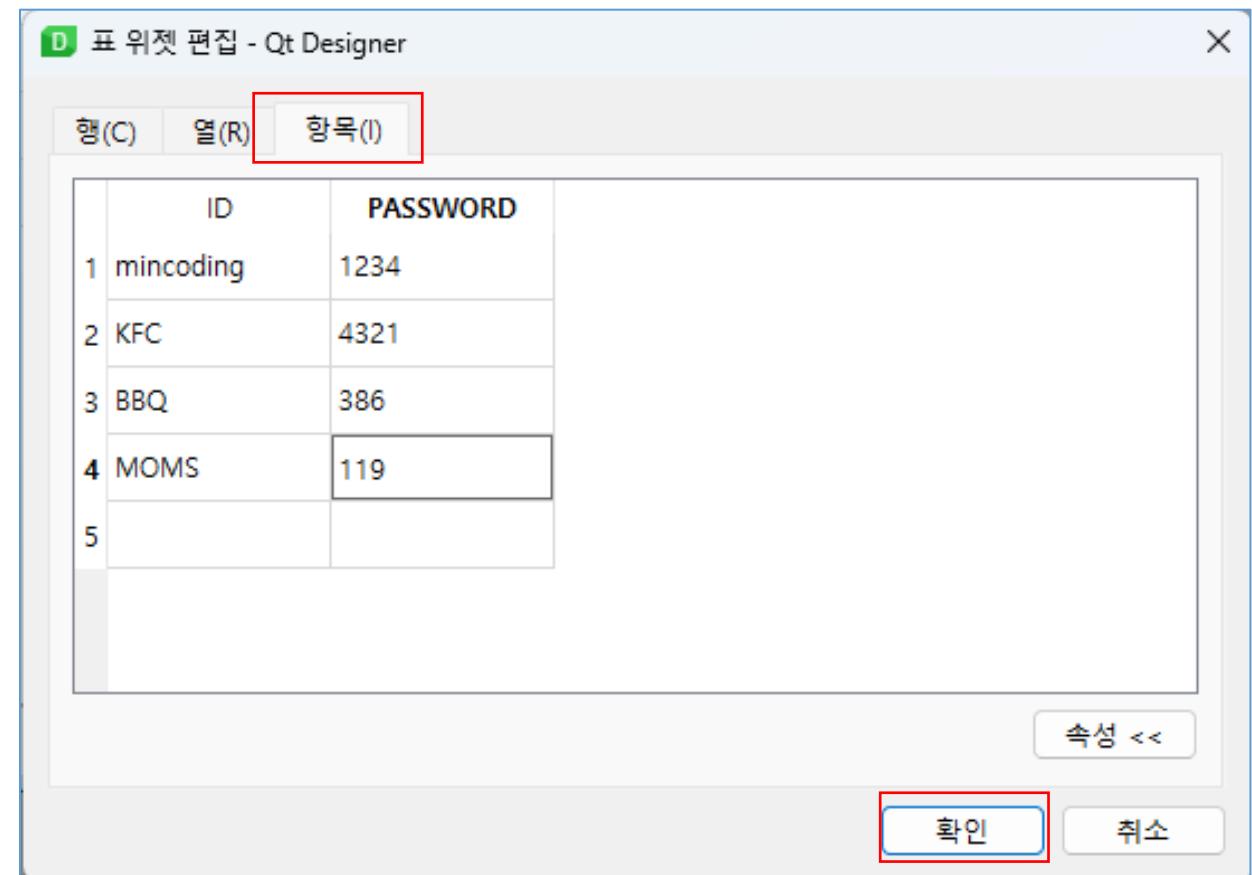


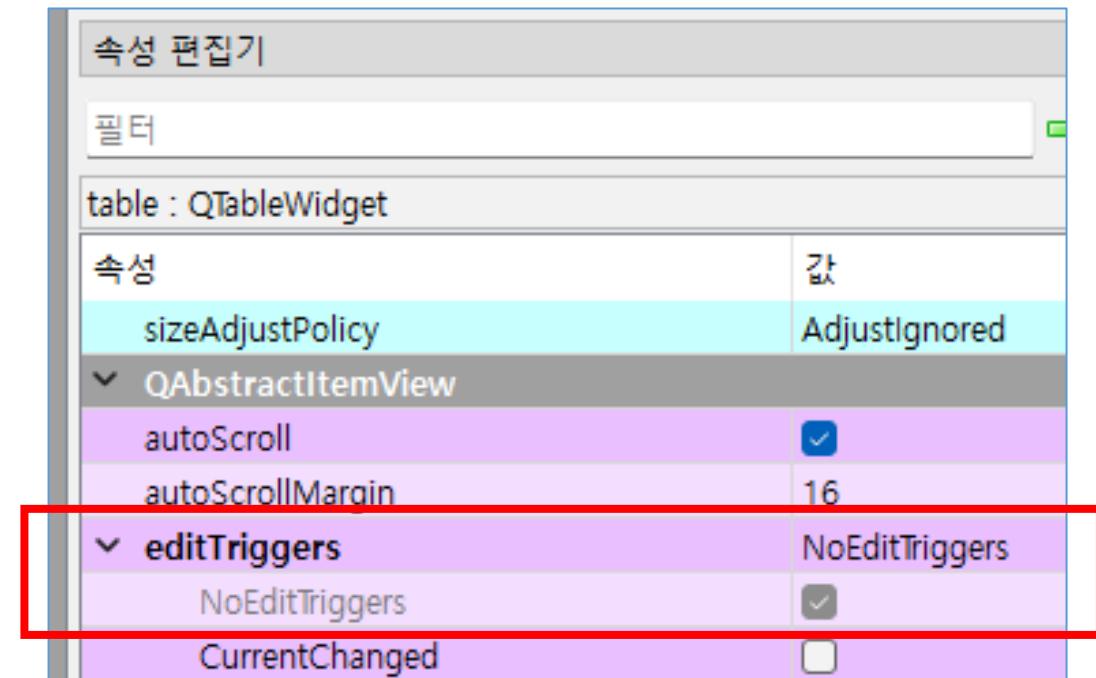
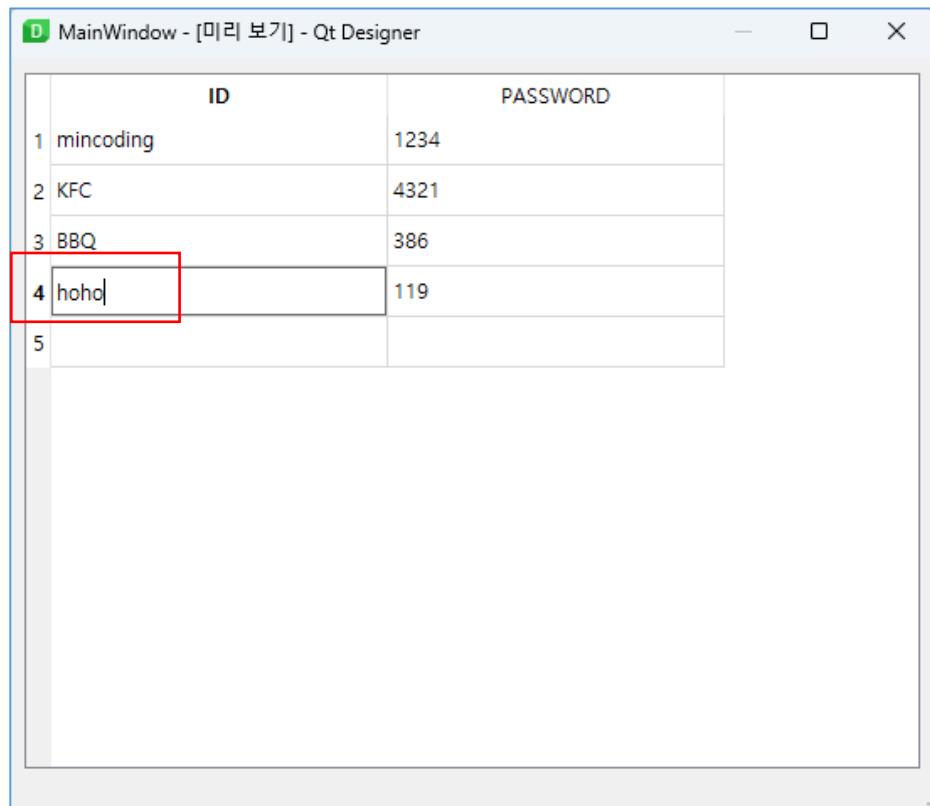
Table 의 data가 들어가는 칸을 Cell 이라 한다.

- 더블 클릭해서 data 직접 입력하기
- 맨 위, 왼쪽 cell 좌표 : (0,0)
- Cell에 들어가는 data 를 item 이라 한다.



미리보기 (Ctrl + R)로 확인한다.

- 입력된 data를 User가 수정할 수 있다.
- editTriggers 옵션 변경
 - NoEditTriggers 체크 (수정 금지 옵션)



Signal / Slot 을 등록하자.

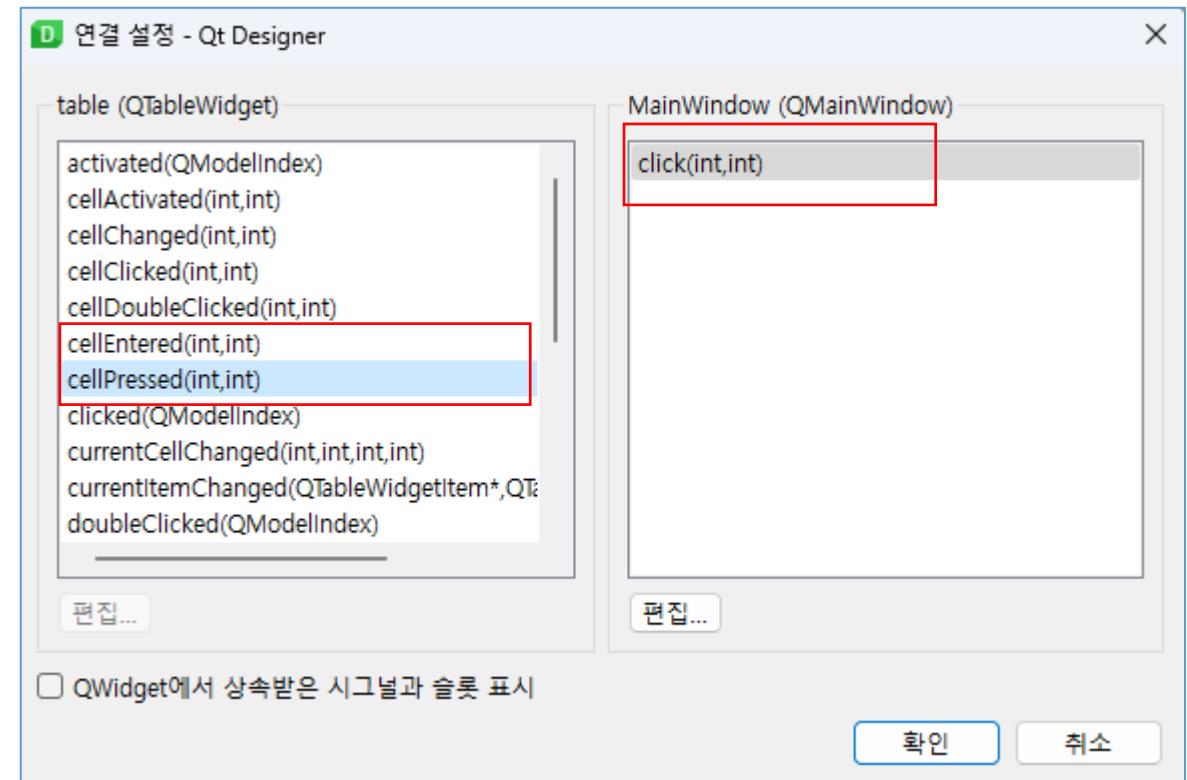
- **signal**

- cellEntered(int,int)
- cellPressed(int,int)

- **slot**

- click(int,int)

시그널/슬롯 편집기			
+	-		
송신자	시그널	수신자	슬롯
table	cellEntered(int,int)	MainWindow	click(int,int)
table	cellPressed(int,int)	MainWindow	click(int,int)



item() 함수로 바로 출력할 수 없다.

- .rowCount() : row 개수
- .columnCount() : column 개수
- .item(y,x) : (x,y) 좌표의 item 의 QTableWidgetItem() 타입으로 return

```
9  def main(self):  
10     for y in range( self.table.rowCount() ):  
11         for x in range( self.table.columnCount() ):  
12             data = self.table.item(y,x)  
13             if data:  
14                 print( data.text(), end = ' ' )  
15     print()
```

<https://gist.github.com/hoconoco/e58051d553f331fb9c38148859694a11>

setItem() 으로 데이터를 수정할 수 있다.

- 공백이 있으면 data를 추가한다.

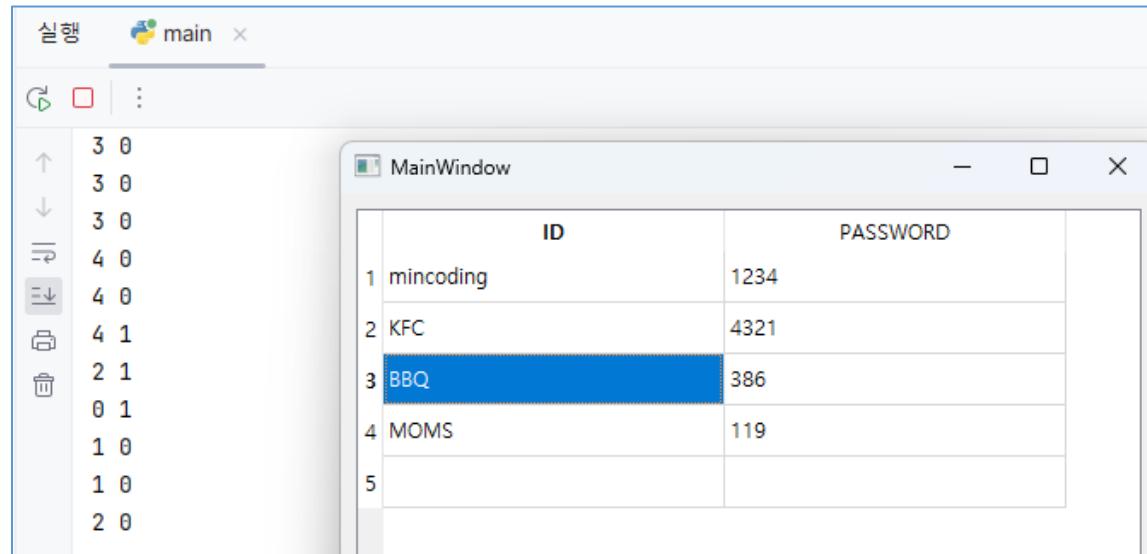
	ID	PASSWORD
1	mincoding	1234
2	KFC	4321
3	BBQ	386
4	MOMS	119
5	HI	HI

```
9     def main(self):
10        for y in range( self.table.rowCount() ):
11            for x in range( self.table.columnCount() ):
12                data = self.table.item(y,x)
13                if data:
14                    print( data.text(), end = ' ' )
15                else:
16                    self.table.setItem(y,x, QTableWidgetItem("HI"))
17                    print()
```

<https://gist.github.com/hoconoco/fd27ba4c64de41644e665a78051060e8>

Table 의 Cell 을 클릭하면, Cell의 x,y 값이 출력된다.

- `cellEntered(int,int)`
- `cellPressed(int,int)`



```
22      def click(self, x, y):  
23          print( x, y )
```

<https://gist.github.com/hoconoco/29ba75ece2b90e2b68dd53ce12a37151>

QProgressBar

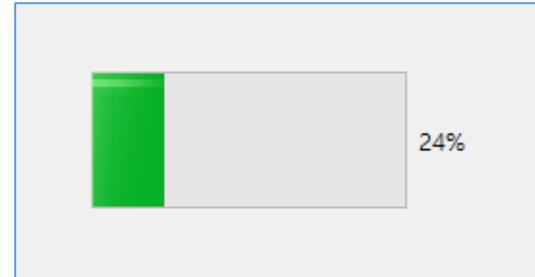
진행 사항을 알려주는 용도로 사용되는 ProgressBar

주요 API

- `value()`
- `setValue()`

주요 옵션

- `minimum`: 최소 값
- `maximum` : 최대값
- `value` : 현재 값
- `textVisibal` : 글씨 보이기 여부
- `orientation` : 수직 / 수평 방향 설정
- `invertedAppearance` : 정방향 / 역방향 설정

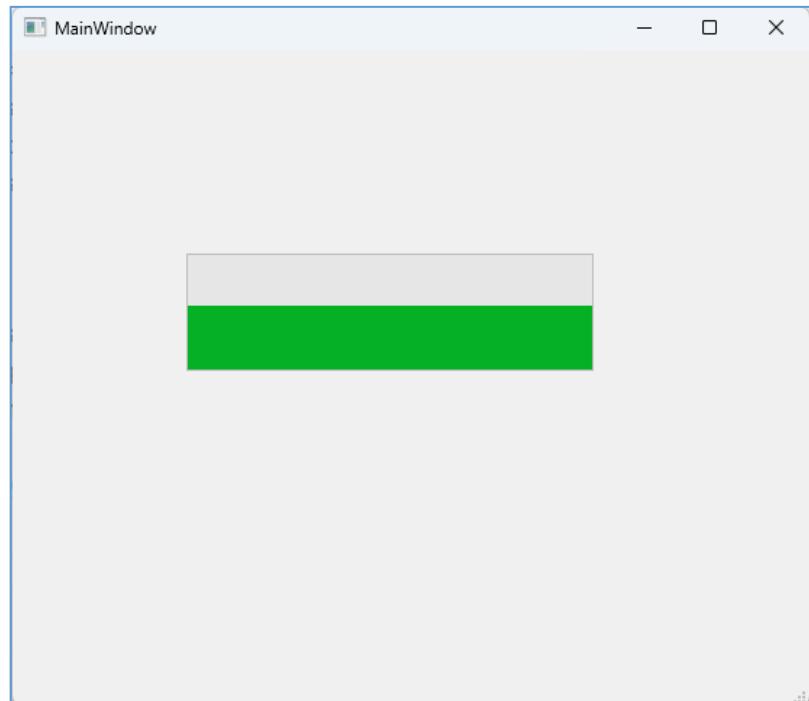


QProgressBar	
minimum	0
maximum	100
value	24
alignment	왼쪽정렬, 수직가운데정렬
textVisible	<input checked="" type="checkbox"/>
orientation	Horizontal
invertedAppearance	<input type="checkbox"/>
textDirection	TopToBottom
format	%p%

<https://doc.qt.io/qtforpython-6/PySide6/QtWidgets/QProgressBar.html>

QTimer 이용해서 ProgressBar 움직이는 코드

- `.setOrientation()` : Qt.Vertical / Qt.Horizontal, 수평 / 수직 방향 변경
- `.setInvertedAppearance()` : True, False, 정방향 / 역방향 변경



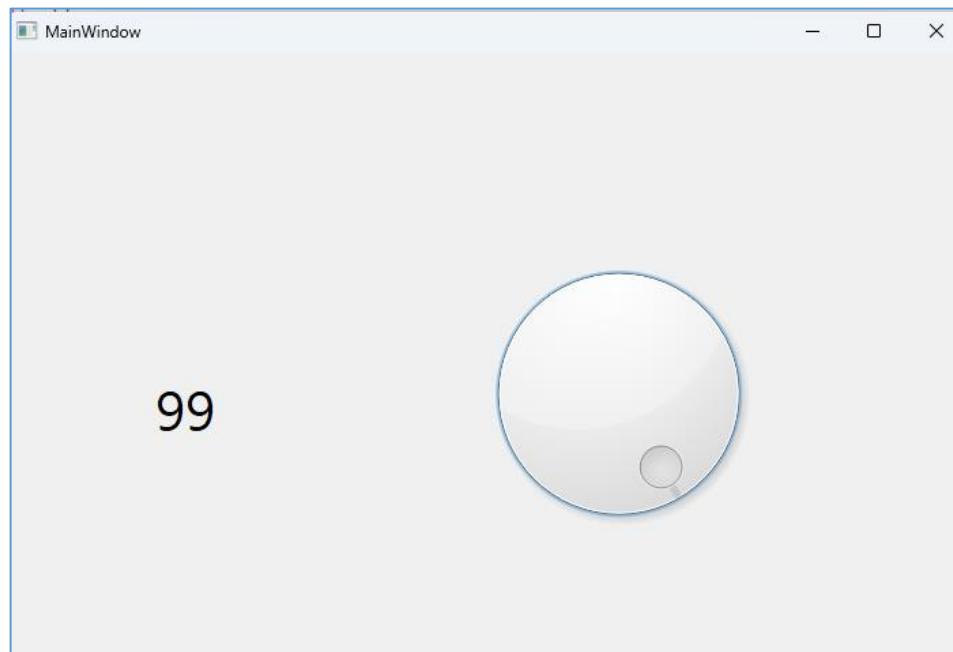
```
11 def main(self):  
12     self.progressBar.setValue(0)  
13     self.progressBar.setOrientation(Qt.Vertical)  
14     self.progressBar.setInvertedAppearance(False)  
15  
16     self.timer = QTimer()  
17     self.timer.setInterval(30)  
18     self.timer.timeout.connect(self.run)  
19     self.timer.start()  
20  
21     1개의 사용 위치  
22     def run(self):  
23         self.progressBar.setValue(self.progressBar.value() + 1)  
24         if self.progressBar.value() == 100:  
25             self.timer.stop()
```

<https://gist.github.com/hoconoco/2e693914c80f5c277bf6bbc642208b5c>

QDial

다이얼을 이용한 사용자 입력 가능

- `setValue()` : 값 정하기
- `value()` : 현재 값 읽기
- `signal` : `valueChanged`



```
10      def main(self):  
11          self.dial.valueChanged.connect(self.run)  
12  
13          1개의 사용 위치  
14      def run(self, val):  
15          self.lbl.setText(str(val))  
          print(val)
```

QSlider

슬라이더를 이용해 값을 변경할 수 있다.

주요 API

- `value()`

주요 옵션

- `minimum`: 최소 값
- `maximum` : 최대값
- `singleStep` : 한번에 증감하는 값
- `pageStep` : page 별 값 증감하는 값
- `value` : 현재 값
- `orientation` : 수직 / 수평 방향 설정
- `invertedAppearance` : 정방향 / 역방향 설정

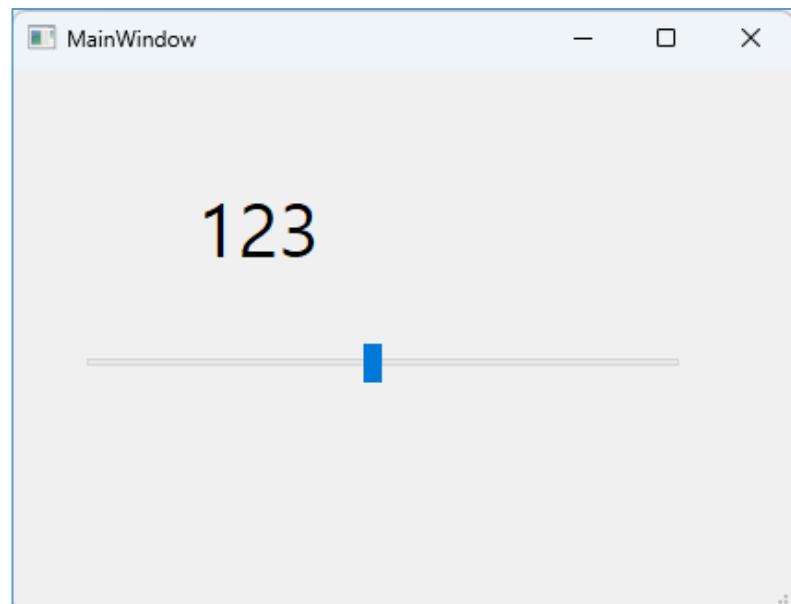


QAbstractSlider	
minimum	0
maximum	99
singleStep	1
pageStep	10
value	50
sliderPosition	50
tracking	<input checked="" type="checkbox"/>
orientation	Horizontal
invertedAppearance	<input type="checkbox"/>
invertedControls	<input type="checkbox"/>

<https://doc.qt.io/qtforpython-6/PySide6/QtWidgets/QSlider.html>

QSlider

- Slider 를 움직여 값을 조절할 수 있다.
- `.setPageStep()` : 슬라이더의 값 증감을 30씩 정하는 API
- `.setMaximum()` : 슬라이더의 최대 값 변경하는 API
- signal : `valueChanged`



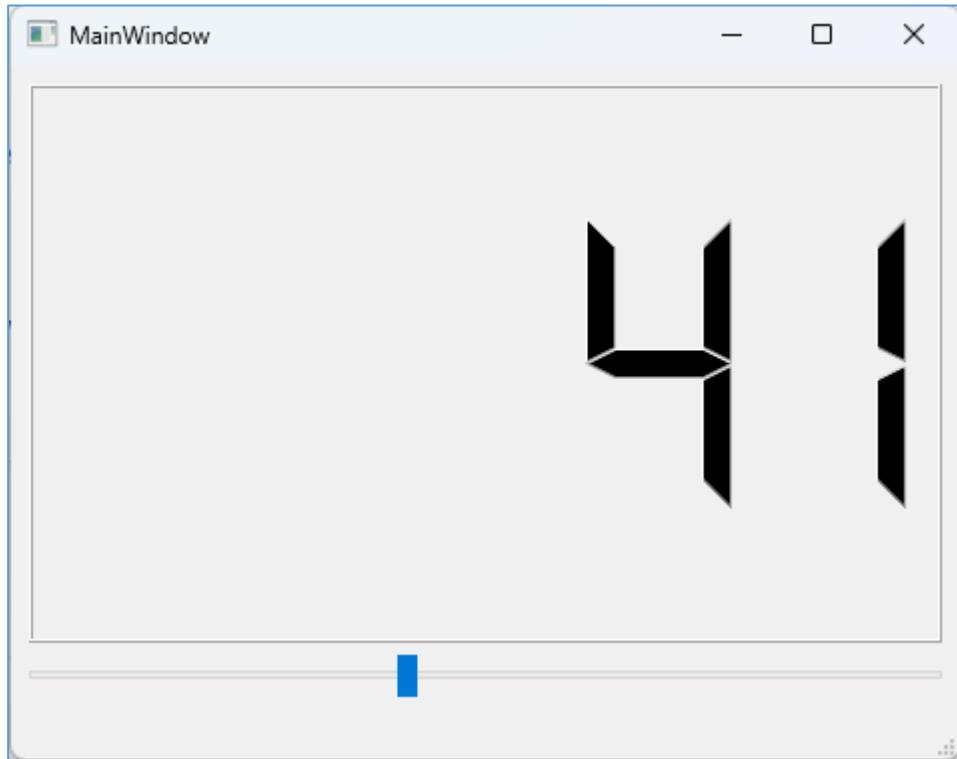
```
10 def main(self):  
11     self.hslider.valueChanged.connect(self.run)  
12     self.hslider.setPageStep(30)  
13     self.hslider.setMaximum(255)  
14  
15     def run(self, val):  
16         self.lbl.setText(str(val))  
17         print(val)
```

<https://gist.github.com/hoconoco/d9875d6c056aea4539ec8a8d206cc29d>

QLCDNumber

숫자 값을 보여주는 display

- `.display()` : 값을 보여준다.



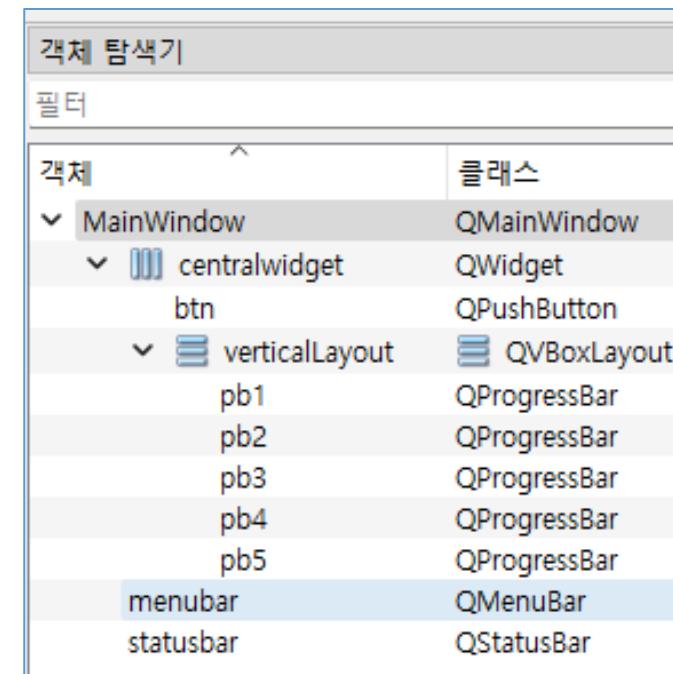
```
10     def main(self):  
11         self.hslider.valueChanged.connect(self.run)  
12  
13     def run(self, val):  
14         self.lcd.display(val)
```

<https://gist.github.com/hoconoco/8958d0173a4b2dc794800c513682dbab>

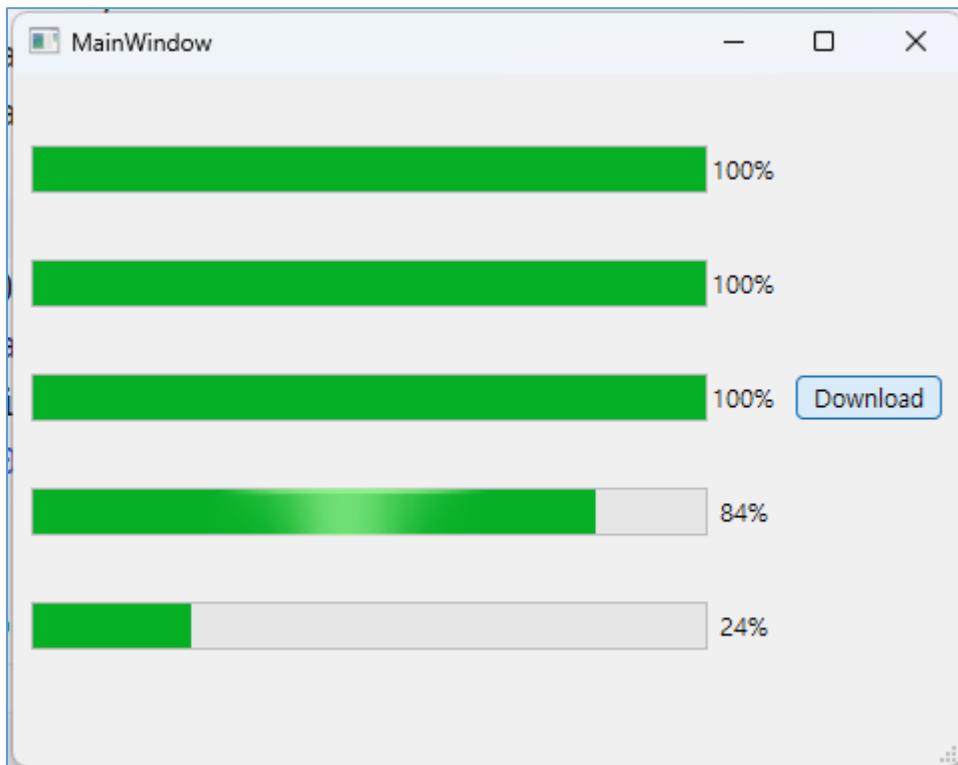
Widget의 List

Python의 list를 이용해 widget에 접근해야 한다.

- 수직배치 레이아웃에 있는 ProgressBar에 접근해보자.



- `.count()` : 레이아웃에 담긴 항목 수를 반환
- `.itemAt(index)` : 레이아웃에 담긴 항목을 반환, index로 접근 가능
- `.widget()` : 해당 widget을 반환

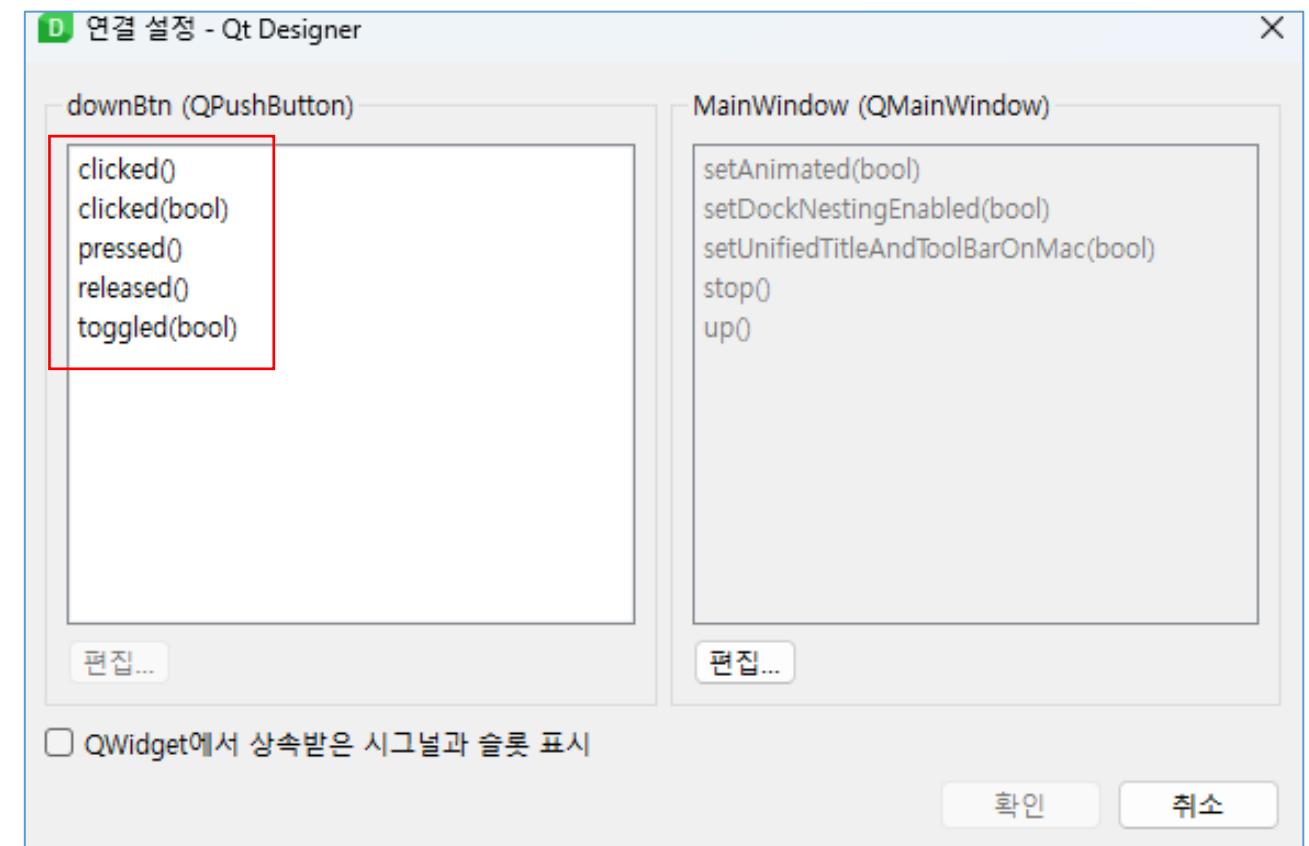


```
11      pplist = []
12
13      def main(self):
14          lay = self.verticalLayout
15          for i in range(lay.count()):
16              self.pplist.append(lay.itemAt(i).widget())
17
18      def go(self):
19          for n in range(5):
20              for val in range(101):
21                  self.pplist[n].setValue(val)
22                  sleep(0.01)
```

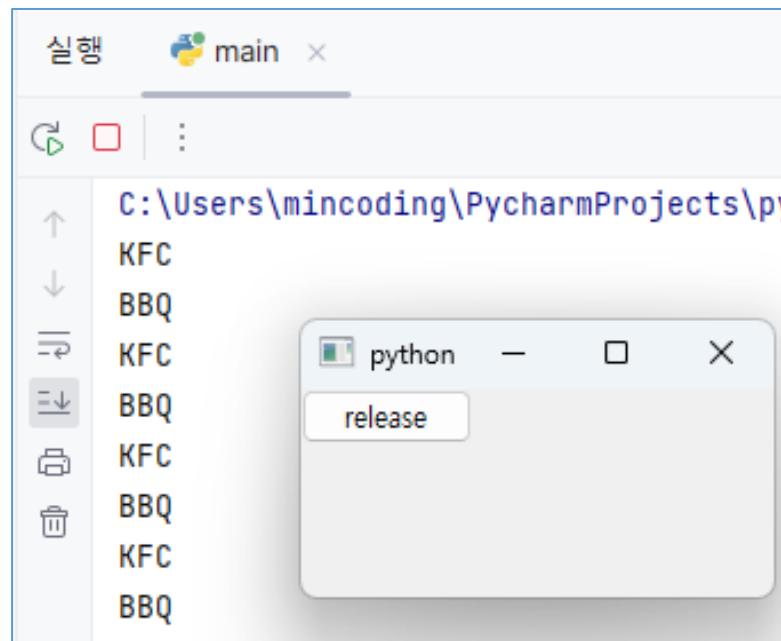
다양한 Signal

버튼의 Signal에 따라 다양한 동작을 구현할 수 있다.

- `clicked()` : 버튼이 클릭 될 때
- `clicked(bool)` : (체크형 버튼) 버튼이 클릭될 때
- `pressed()` : 버튼을 누를 때
- `released()` : 버튼에서 뗄 때
- `toggled(bool)` : (체크형 버튼) 토글 될 때



버튼을 누르고 있으면, “KFC”
떼면, “BBQ” 출력하는 샘플 코드



```
8     def main(self):
9         self.btn = QPushButton("클릭?!", self)
10        self.btn.pressed.connect(self.press)
11        self.btn.released.connect(self.release)
12
13    def press(self):
14        print("KFC")
15        self.btn.setText("press")
16
17    def release(self):
18        print("BBQ")
19        self.btn.setText("release")
```

Day3-3. RaspberryPi with Qt

챕터의 포인트

- 라즈베리파이와 Pyside2
- 개발환경 구축
- Cross Compile
- Qt + H/W

라즈베리파이와 Pyside2

Qt의 모토

One framework, One Codebase, Any Platform

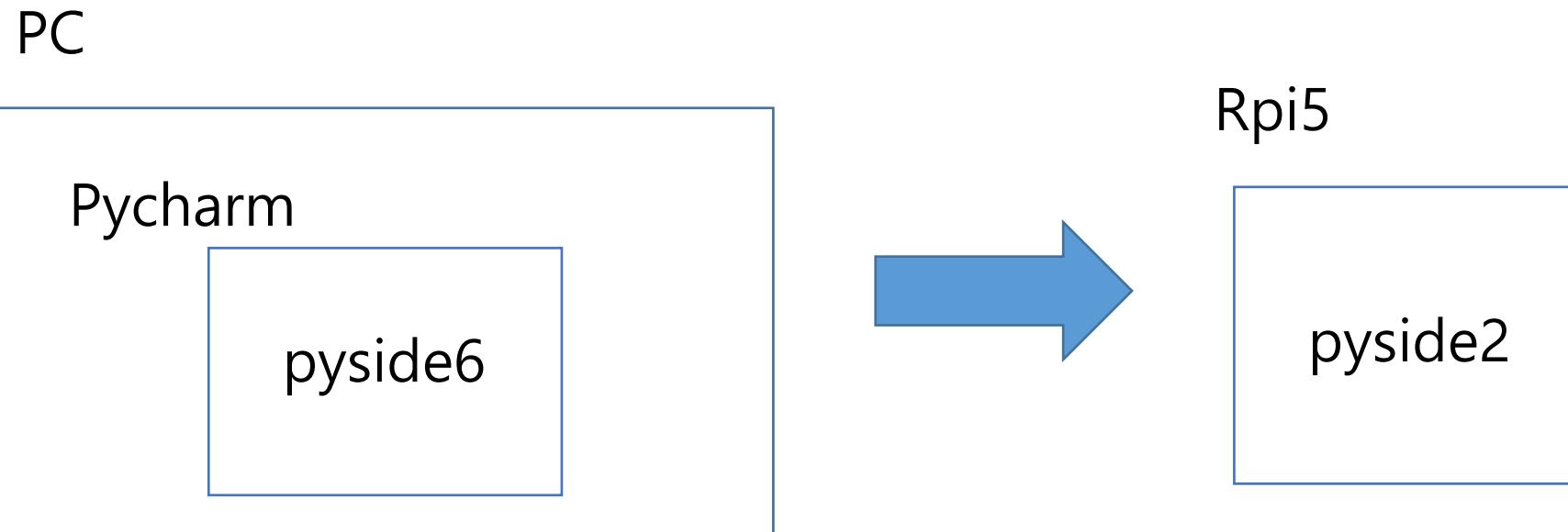
우리는 라즈베리파이에 Qt를 이용한
GUI App 개발을 할 것이다.



라즈베리파이는 아직 PySide6 를 지원하지 않는다.

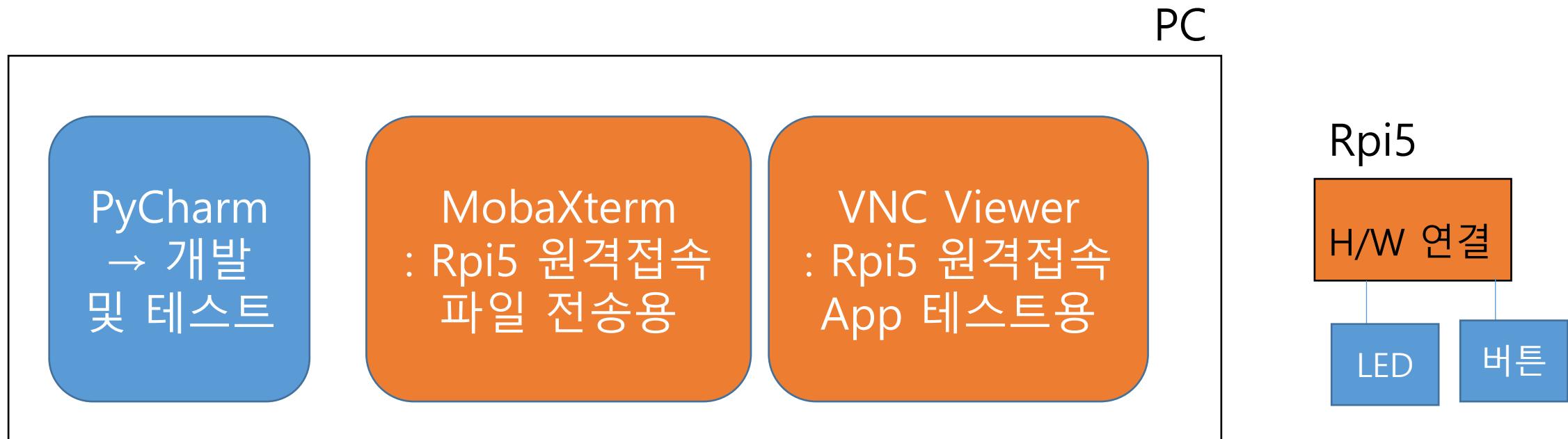
→ 그래서 Pyside2 를 설치 한 뒤, 코드를 수정해서 빌드한다.

- 사용법이 동일하며, API 이름만 조금 다르다.



MobaXterm 과 VNC Viewer 를 활용한다.

- MobaXterm : PC에서 개발한 코드를 쉽게 옮길 수 있다.
- VNC Viewer : 빌드한 결과를 쉽게 확인할 수 있다.
 - 라즈베리파이의 LCD를 연결해도 좋다.



개발환경 구축

Pyside2 설치하기

- sudo apt install pyside2-tools -y
- sudo apt install python3-pyside2.* -y

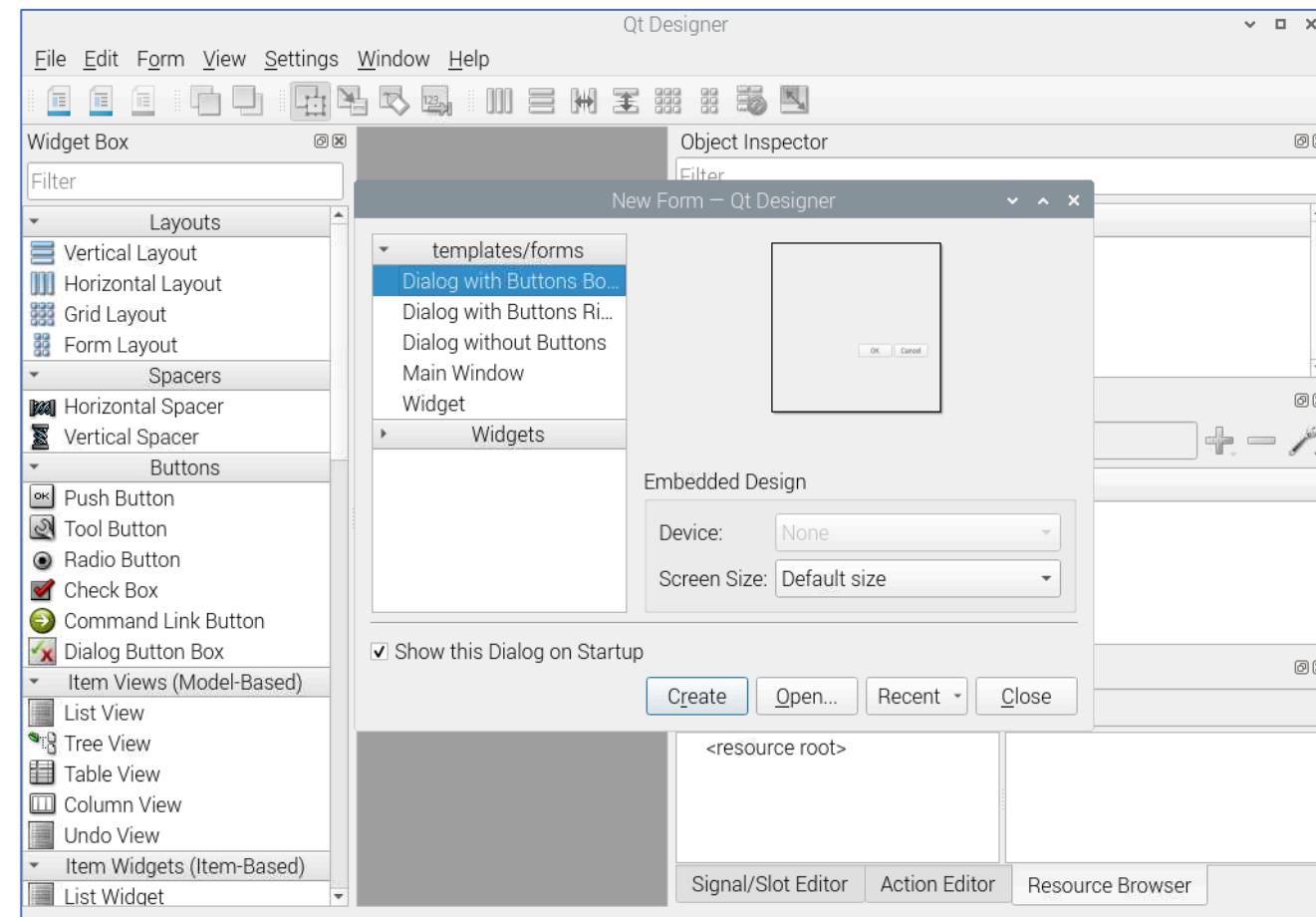
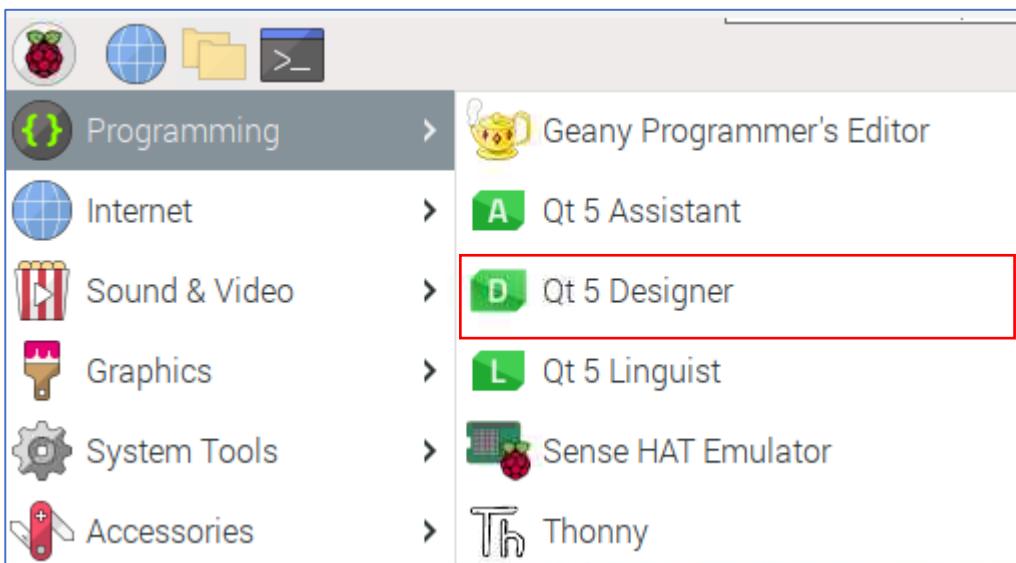
[참고] Qt Designer 설치

- sudo apt-get install qttools5-dev-tools -y

```
pi@raspberrypi:~ $ sudo apt install pyside2-tools -y
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
pyside2-tools is already the newest version (5.15.2-1).
The following package was automatically installed and is no longer needed:
  libfuse2
Use 'sudo apt autoremove' to remove it.
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
pi@raspberrypi:~ $ sudo apt install python3-pyside2.* -y
Reading package lists... Done
Building dependency tree... Done
```

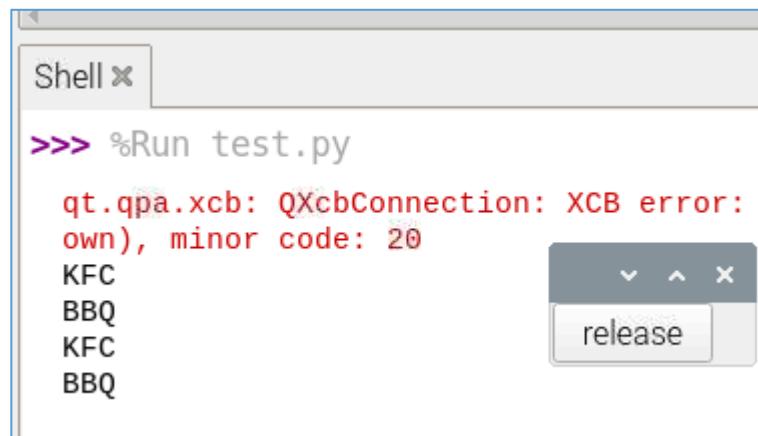
Qt Designer 도 설치 가능하다.

- 하지만, 라즈베리파이에서 GUI를 직접 제작하는 것은 힘들기 때문에, PC에서 개발한다.



코드를 작성한 뒤 빌드한다.

- 빌드를 한 뒤, 버튼을 눌러 동작을 테스트한다.
- xcb error는 무시해도 된다.



<https://gist.github.com/hoconoco/d86279037faed796cf382133c2fd9913>

```
from PySide2.QtWidgets import *

class MyApp(QWidget):
    def __init__(self):
        super().__init__()
        self.main()

    def main(self):
        self.btn = QPushButton("click!", self)
        self.btn.pressed.connect(self.press)
        self.btn.released.connect(self.release)

    def press(self):
        print("KFC")
        self.btn.setText("press")

    def release(self):
        print("BBQ")
        self.btn.setText("release")

if __name__ == "__main__":
    app = QApplication(sys.argv)
    ex = MyApp()
    ex.show()
    sys.exit(app.exec_())
```

두 패키지는 큰 차이가 없다.

1) 패키지 이름 차이

```
from PySide2.QtWidgets import *
```

2) .exec() API 이름 차이

- python 의 exec() API와 겹쳐서 PySide2 에서는 exec_() 사용

```
app.exec_()
```

Cross Compile

개발 PC와 실제 동작하는 기기가 다른 경우,

개발 PC에서 소스 코드를 작성 한 뒤,

실제 동작하는 기기로 코드를 옮겨서 Compile 을 하여 실행하는 것을
Cross Compile 이라 한다.

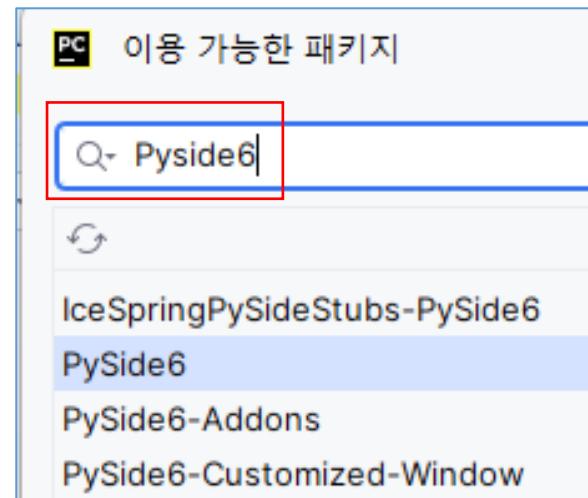
- 실제 동작하는 기기에서 코드 작업 및 빌드 테스트가 쉽지 않은 경우
- 실제 동작하는 기기보다 대부분 PC의 성능이 훨씬 좋다.

임베디드 개발에서는 피해갈 수 없는 방법이다.

- 피할 수 없으면 즐겨라?!

새 프로젝트를 만들고,
PySide6 패키지를 설치한다.

- 파일 → 새 프로젝트.. 클릭
- 파일 → 설정 → 프로젝트 → Python 인터프리터
→ PySide6 설치



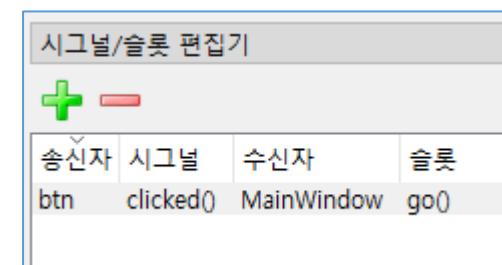
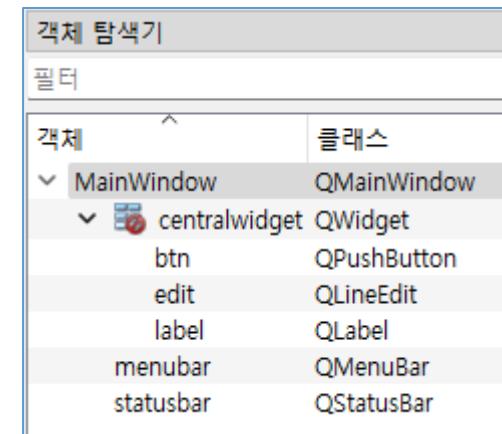
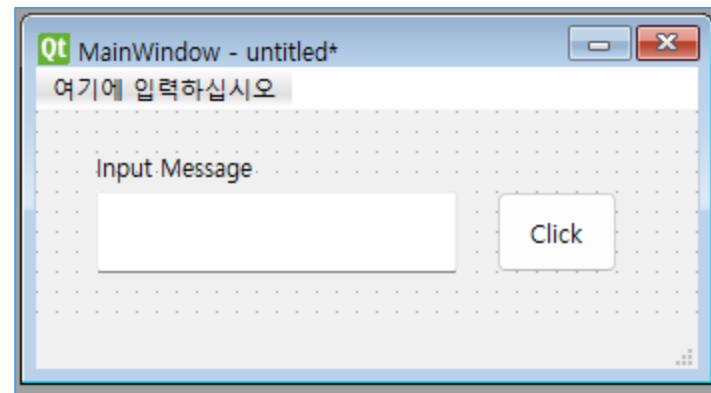
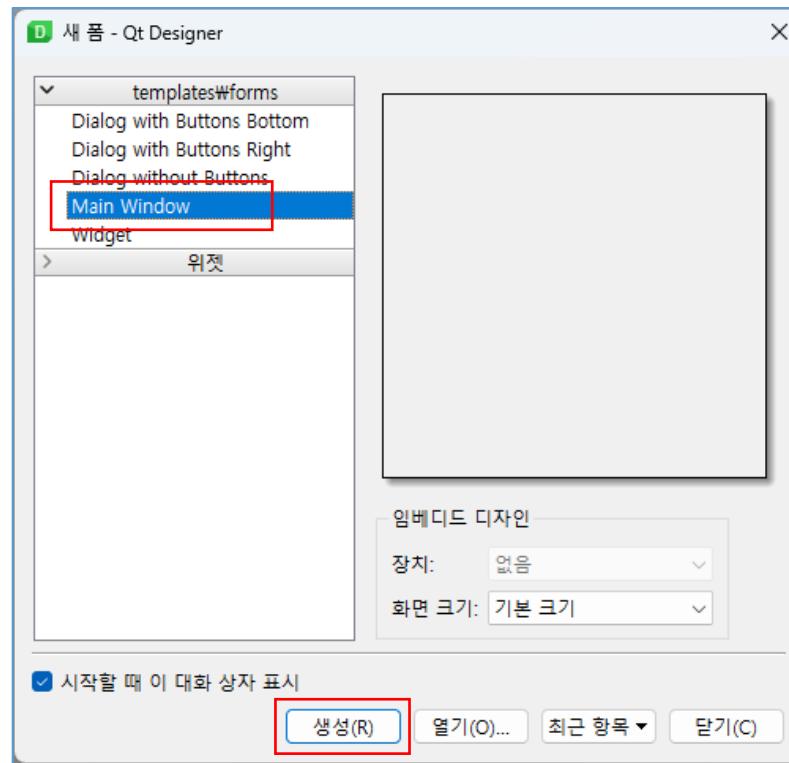
Qt Designer 실행

- 프로젝트 폴더 → venv → Scripts → pyside6-designer.exe 실행



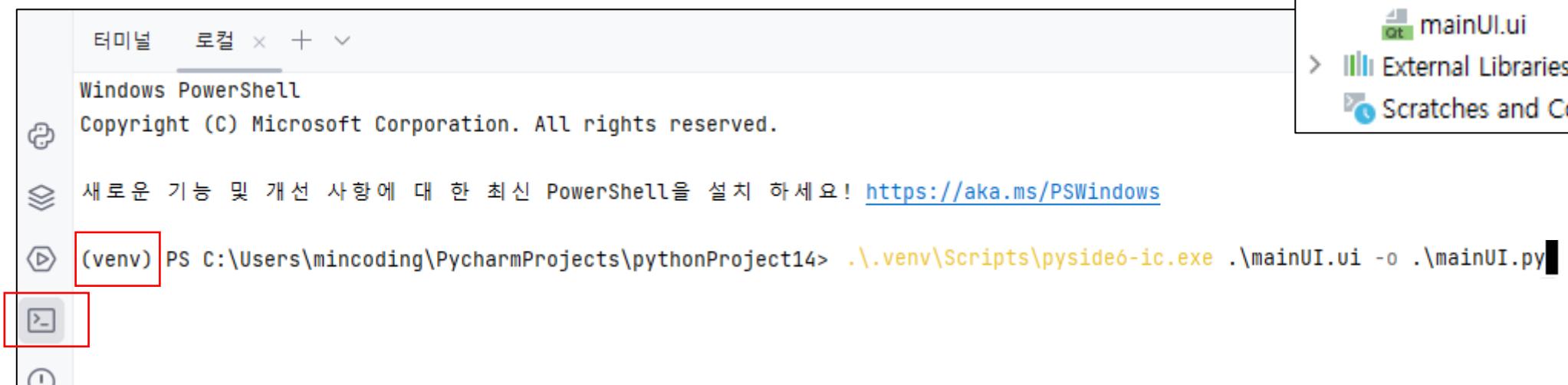
Main Window로 GUI App 제작 후 저장

- 라즈베리파이는 Main Window 만 원활하게 동작한다.
- QLineEdit에 작성한 메시지를 버튼을 눌러서 SenseHat으로 출력할 예정

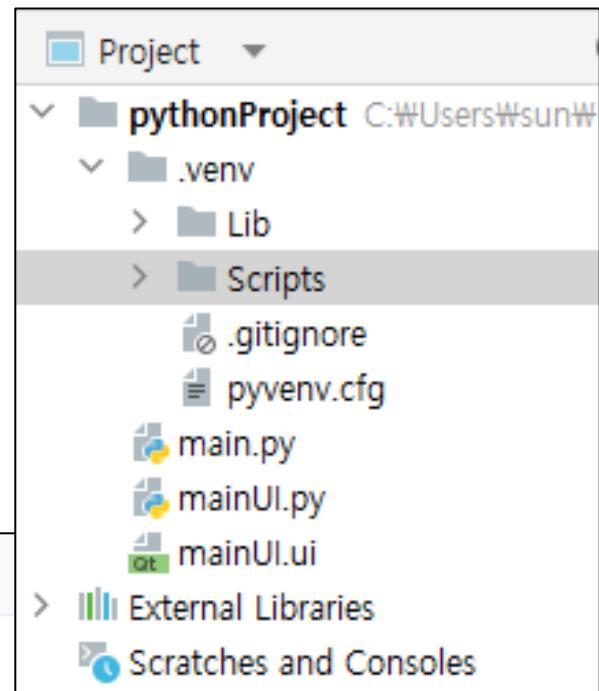


uic 유틸리티로 .py 로 변환 후 동작 테스트를 한다.

- .\venv\Scripts\pyside6-uic.exe .\mainUI.ui -o .\mainUI.py

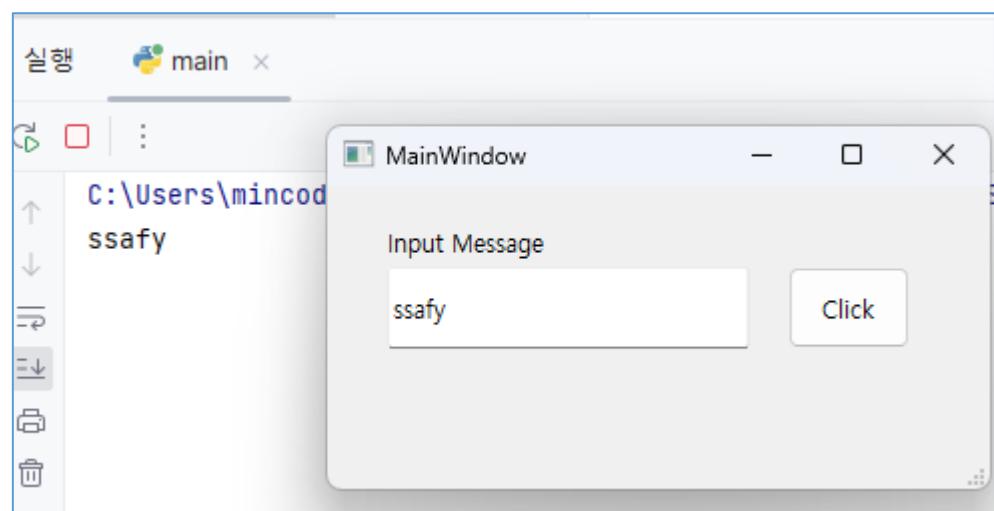


The screenshot shows the PyCharm IDE interface. On the left is a terminal window titled "터미널" (Terminal) running "Windows PowerShell". The terminal output shows the command: ".\venv\Scripts\pyside6-uic.exe .\mainUI.ui -o .\mainUI.py". A red box highlights the terminal icon in the bottom-left corner of the terminal window. On the right is the "Project" tool window, which displays the file structure of the "pythonProject" folder. The "Scripts" directory is selected, showing files like .gitignore, pyvenv.cfg, main.py, mainUI.py, and mainUI.ui. A red box highlights the "Scripts" folder in the project tree.



코드를 작성 한 뒤 빌드한다.

- 이 코드를 이제 Rpi5로 옮겨서 H/W와 연동한다.

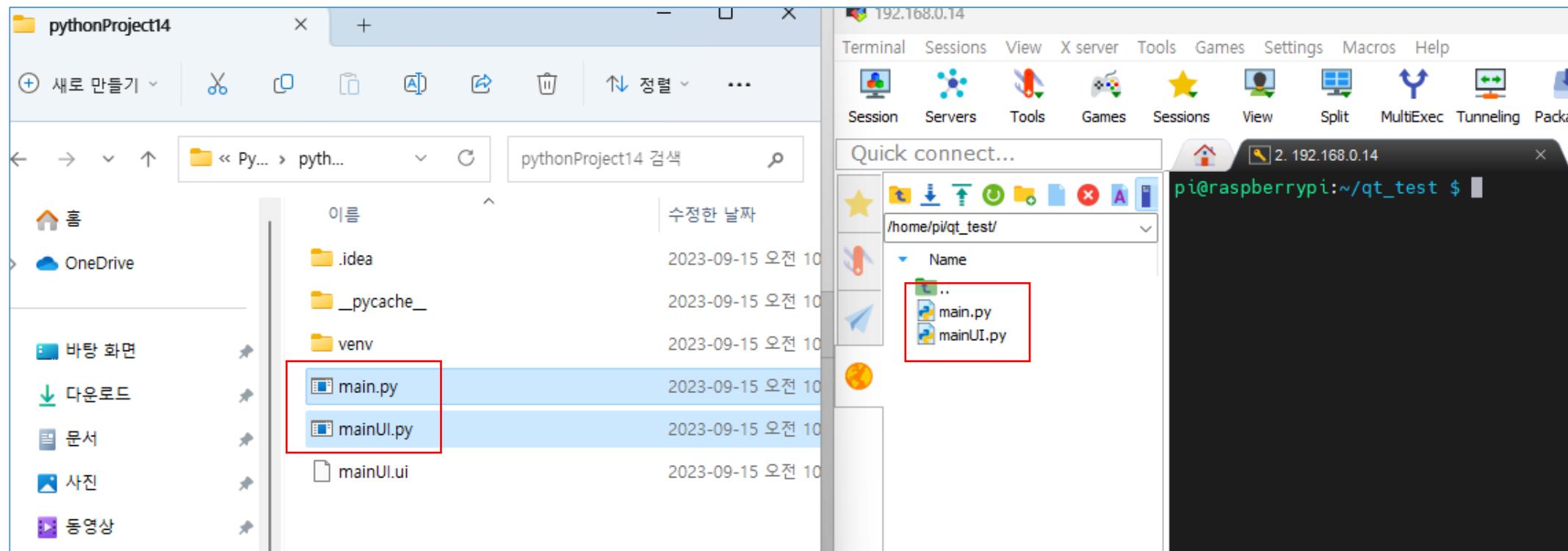


```
main.py x mainUI.py
1  from PySide6.QtWidgets import *
2  from mainUI import Ui_MainWindow
3
4  class MyApp(QMainWindow, Ui_MainWindow):
5      def __init__(self):
6          super().__init__()
7          self.setupUi(self)
8
9
10     def go(self):
11         print(self.edit.text())
12
13     if __name__ == '__main__':
14         app = QApplication()
15         win = MyApp()
16         win.show()
17         app.exec()
```

<https://gist.github.com/hoconoco/36ed83bb0bde719d099efeb1ae9d4dcd>

Pycharm에서 작성한 코드를 MobaXterm을 이용해 Rpi5로 전송한다.

- 프로젝트폴더 마우스 우클릭 → 다음에서 열기 → 탐색기 클릭
- 프로젝트 폴더 내 main.py , mainUI.py 를 옮긴다.



코드를 모두 수정한다.

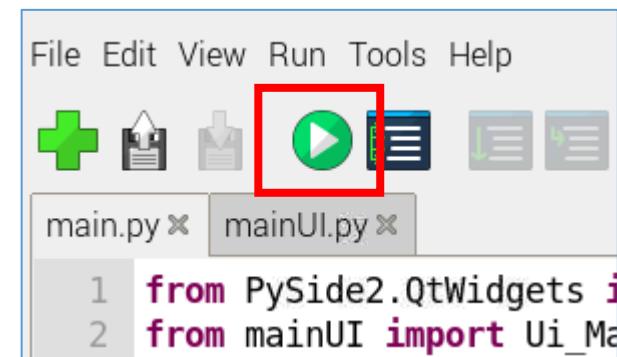
- **main.py**
 - PySide6 → PySide2
 - app.exec() → app.exec_()
- **mainUI.py**
 - PySide6 → PySide2 (3군데)

```
main.py * x mainUI.py * x
9 #####
10
11 from PySide2.QtCore import (QCoreApplication,
12     QMetaObject, QObject, QPoint, QRect,
13     QSize, QTime, QUrl, Qt)
14 from PySide2.QtGui import (QBrush, QColor,
15     QFont, QFontDatabase, QGradient, QIcon,
16     QImage, QKeySequence, QLinearGradient,
17     QPalette, QPixmap, QRadialGradient, QT
18 from PySide2.QtWidgets import (QApplication,
19     QMenuBar, QPushButton, QSizePolicy, QS
20     QWidget)
21
```

```
main.py x mainUI.py x
1 from PySide2.QtWidgets import *
2 from mainUI import Ui_MainWindow
3
4 class MyApp(QMainWindow, Ui_MainWindow):
5     def __init__(self):
6         super().__init__()
7         self.setupUi(self)
8
9     def go(self):
10        print(self.edit.text())
11
12 if __name__ == '__main__':
13     app = QApplication()
14     win = MyApp()
15     win.show()
16     app.exec_()
```

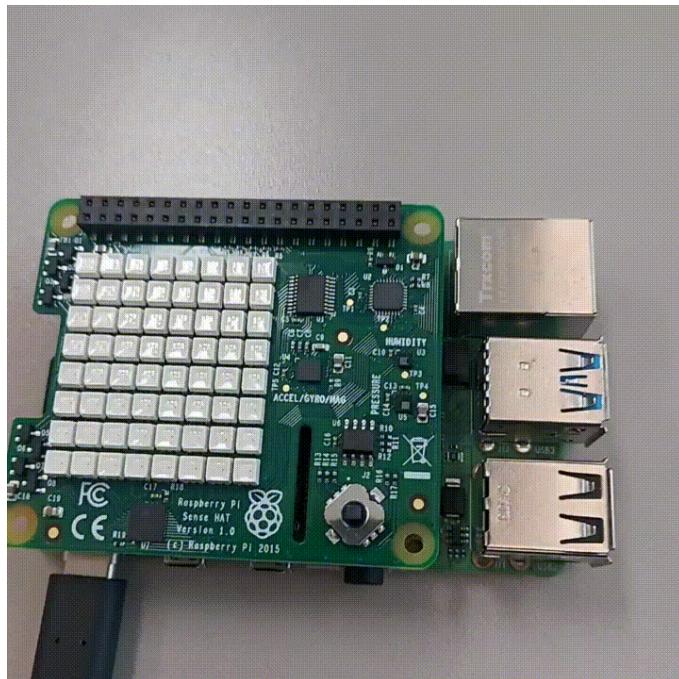
빌드 버튼을 눌러 동작 테스트한다.

- 마찬가지로 잘 된다.
- 이제 senseHat을 이용해 출력한다.



빌드 버튼을 눌러 동작 테스트한다.

- senseHat을 연결해서 테스트 완료
- GUI App을 개발하여, H/W 제어 성공!

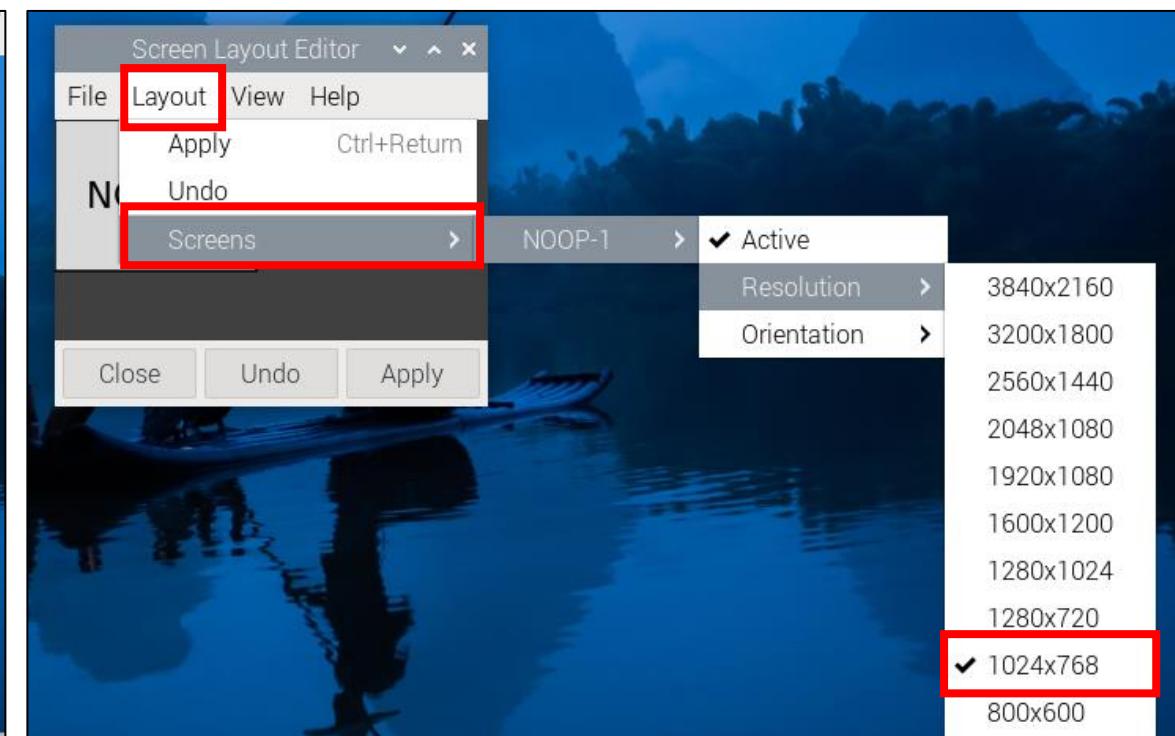
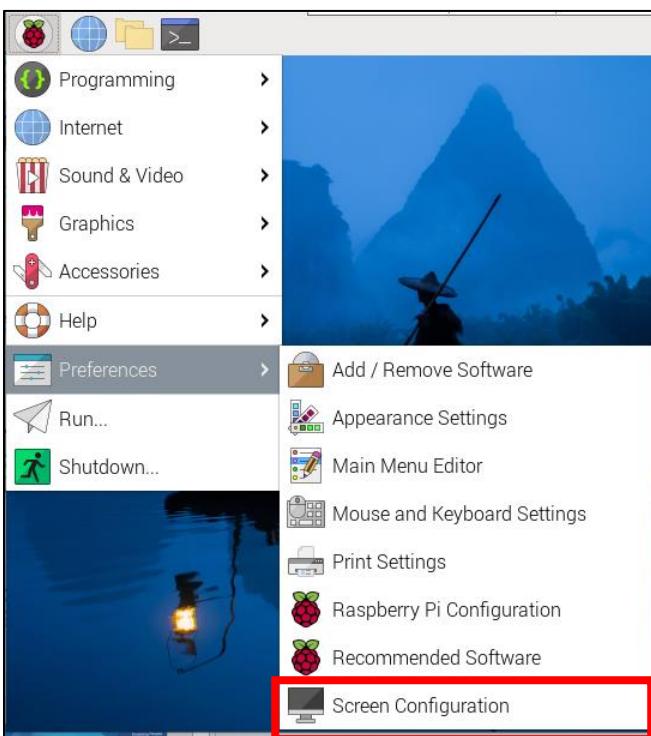


```
main.py x
1 from PySide2.QtWidgets import *
2 from mainUI import Ui_MainWindow
3 from sense_hat import SenseHat
4
5 class MyApp(QMainWindow, Ui_MainWindow):
6     def __init__(self):
7         super().__init__()
8         self.setupUi(self)
9         self.sense = SenseHat()
10
11     def go(self):
12         self.sense.show_message(self.edit.text())
13         print(self.edit.text())
```

<https://gist.github.com/hoconoco/9482758c6268a528343d5ab462f5925e>

라즈베리파이의 VNC 해상도 변경

- 메뉴 > preferences > Screen Configuration



개발하는 모니터의 크기에 맞춰서 앱을 개발하고 싶다.

- 임베디드 업계에서 모니터의 크기는 제각각이다.
- Qt를 이용해서 해상도를 알아내자.
- 라즈베리파이 7인치 LCD모니터 : 800x480

```
10     def main(self):  
11         self.screen = QGuiApplication.primaryScreen()  
12         self.screen_geometry = self.screen.geometry()  
13  
14         w = self.screen_geometry.width()  
15         h = self.screen_geometry.height()  
16         print("w : ", w, ", h : ", h)
```

<https://gist.github.com/hoconoco/878e70e5410d73f81b2799cd71663eae>

라즈베리파이 모니터에 MainWindow 크기를 맞춘 코드

- 7인치 모니터와 waveshare 모니터의 해상도는 다르다.
- width, height 변수에 앞에서 측정한 해상도 값을 넣으면, 원하는 해상도의 모니터로 맞출 수 있다.

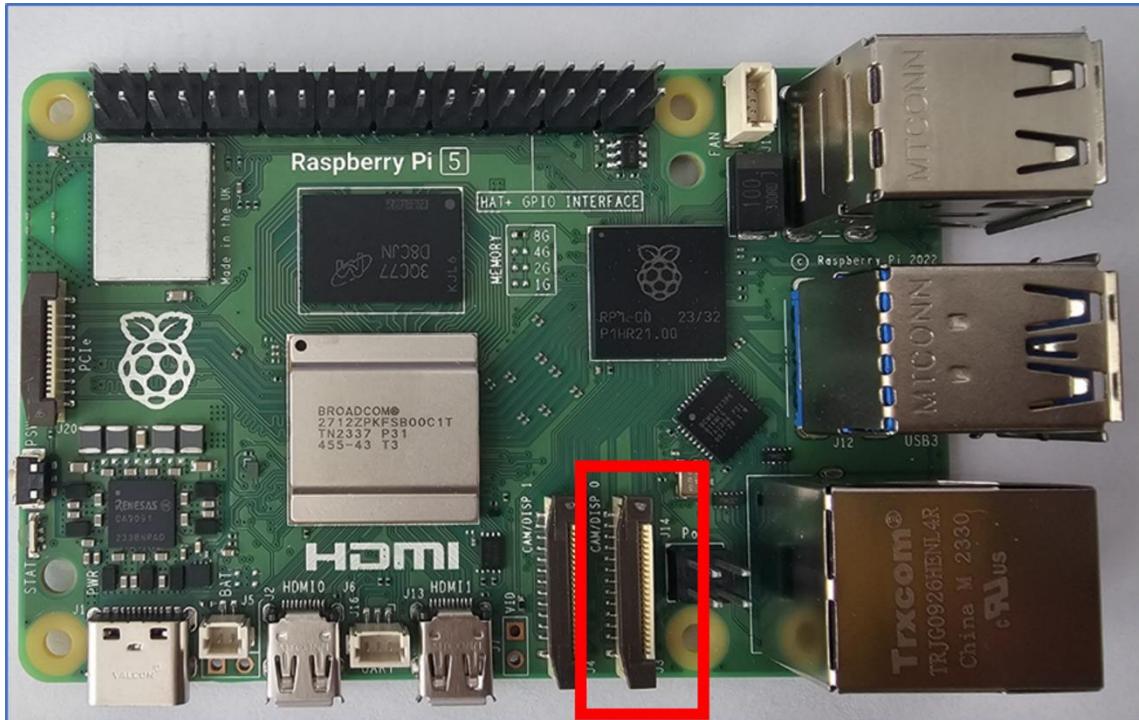
```
5     width = 1024
6     height = 600
7
8     class MyApp(QMainWindow):
9         def __init__(self):
10             super().__init__()
11             self.main()
12
13         def main(self):
14             # -60은 화면 상단 작업표시줄 높이
15             self.setGeometry(0,60, width, height-60)
16             self.bar = self.statusBar()
17             self.bar.showMessage(str(self.width())+","+ str(self.height()))
```

<https://gist.github.com/hoconoco/6d08667a06e581a214e9261b61d3a577>

카메라 테스트

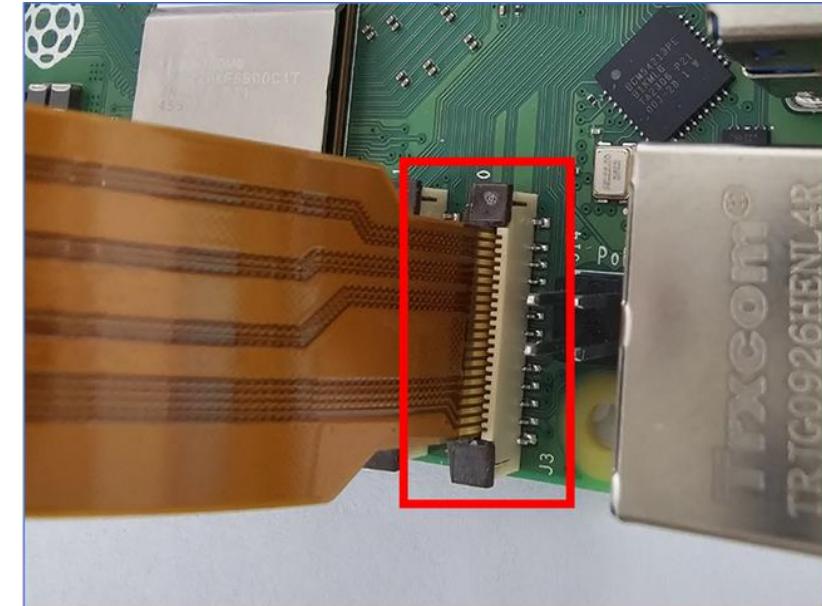
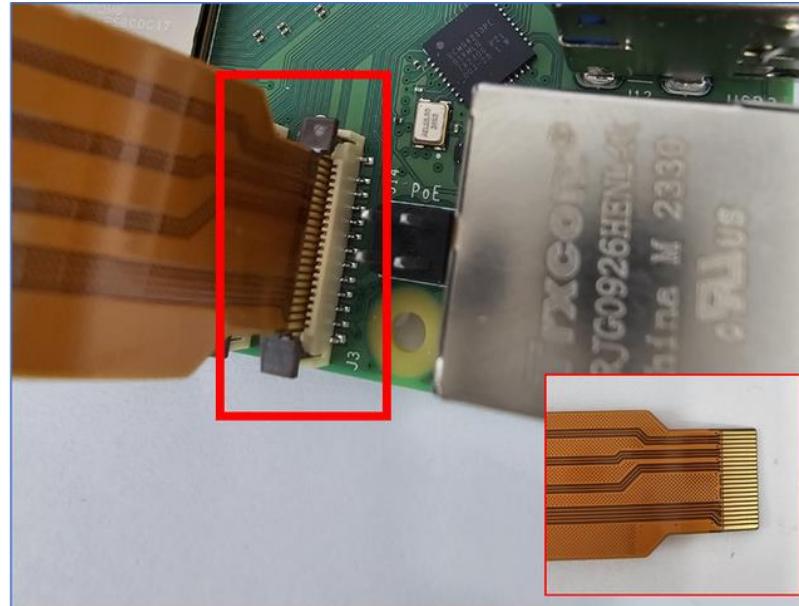
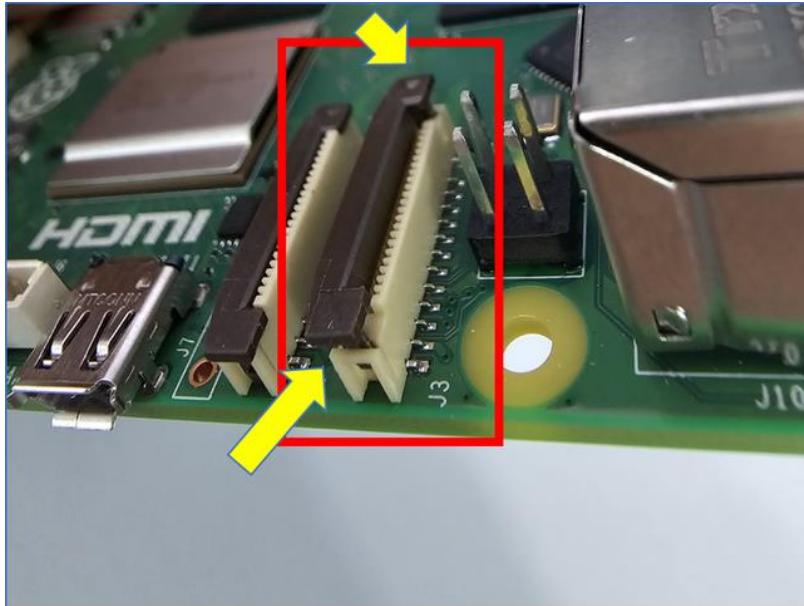
파이 카메라를 연결한다.

- 반드시, 라즈베리파이를 종료한 뒤, 전원 케이블까지 뽑은 다음 카메라를 연결한다.



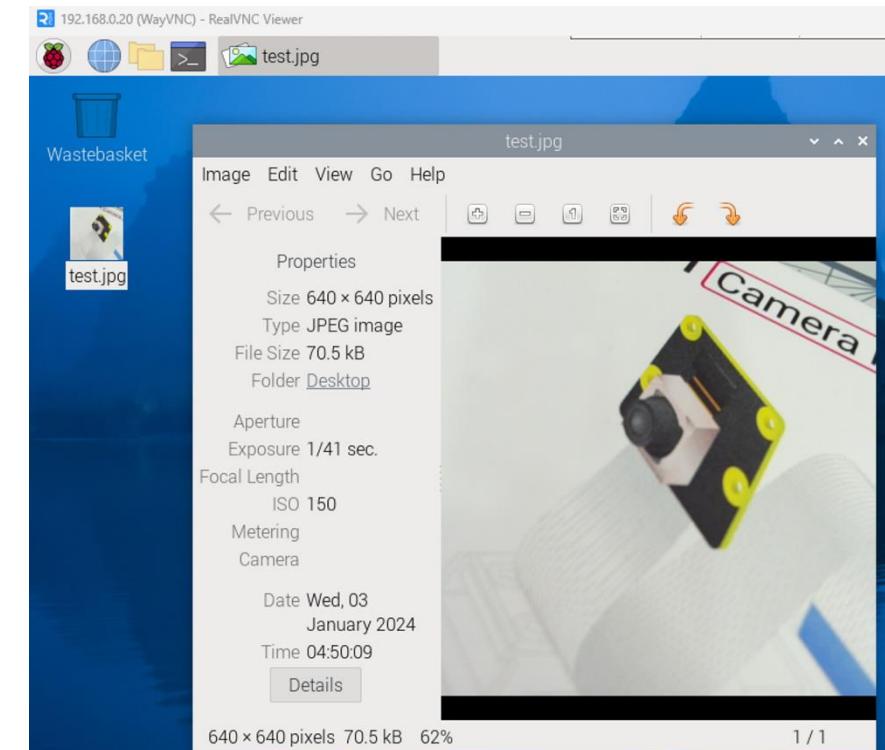
파이 카메라를 연결한다.

- 반드시, 라즈베리파이를 종료한 뒤, 전원 케이블까지 뽑은 다음 카메라를 연결한다.
 - mobaXterm 이 아니라 VNC Viewer or 모니터를 연결한 상태에서 입력해야 한다.
mobaXterm 의 Xwindow 기능이 속도가 빨라 카메라 촬영이 되지 않는다.



카메라 촬영 시, 금속 판을 꼭! 손으로 가린 뒤 촬영한다.

- sudo apt-get update
- sudo apt-get install fswebcam -y
- 촬영 : rpicam-still --width 640 --height 640 -o ~/Desktop/test.jpg



OpenCV 설치

다음 내용을 복사해서 라즈베리파이 쉘에 붙여 넣어 설치한다.

- 소요시간 (최대 90분)

```
mkdir opencv4.9
cd ./opencv4.9
wget https://github.com/Qengineering/Install-OpenCV-Raspberry-Pi-64-bits/raw/main/OpenCV-4-9-0.sh
sudo chmod 755 ./OpenCV-4-9-0.sh

./OpenCV-4-9-0.sh
sudo reboot

sudo rm -r ./opencv
sudo rm -r ./opencv_contrib
```

OpenCV 설치가 완료되면, Python 을 실행하여 테스트한다.

- `python3`
- `import cv2`
- `cv2.__version__`
- 출력 결과 : '4.9.0' 나오면 성공

```
pi@raspberrypi:~ $ python3
Python 3.11.2 (main, Mar 13 2023, 12:18:29) [GCC 12.2.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import cv2
>>> cv2.__version__
'4.9.0-dev'
>>>
```

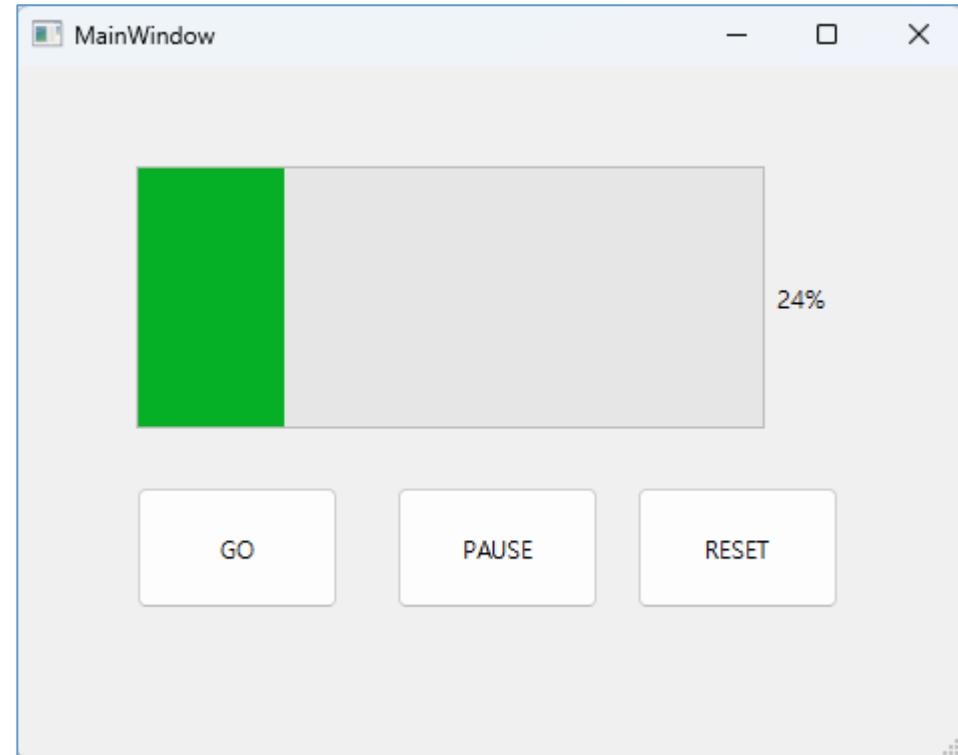
Day3-4 도전

사용한 위젯 (QTimer 사용하기)

- Progress bar, Push Button

구현할 기능

- GO버튼 : 다운로드가 진행된다. 100%에서 정지
- PAUSE버튼 : 다운로드가 멈춘다.
- RESET버튼 : 다운로드를 0부터 다시 진행한다.

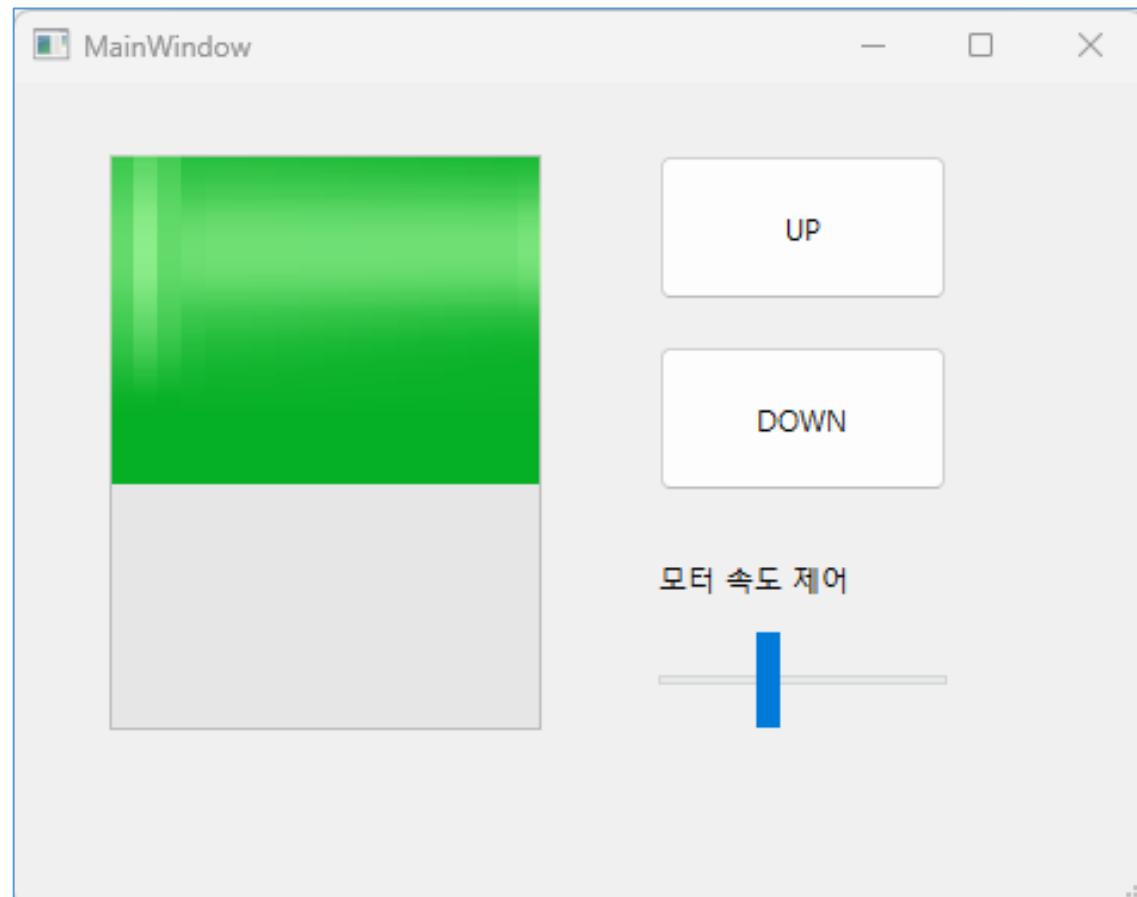


사용한 위젯 (QTimer 사용하기)

- Progress bar, Push Button, Slider, Label

구현할 기능

- UP버튼
 - 누르고 있으면 커튼이 올라간다
 - 떼면, 멈춘다.
- DOWN버튼
 - 누르고 있으면 커튼이 내려온다
 - 떼면, 멈춘다.
- 슬라이더 조절
 - 커튼의 속도가 조절된다.



사용한 위젯 (QThread 사용하기)

- Progress bar, Push Button

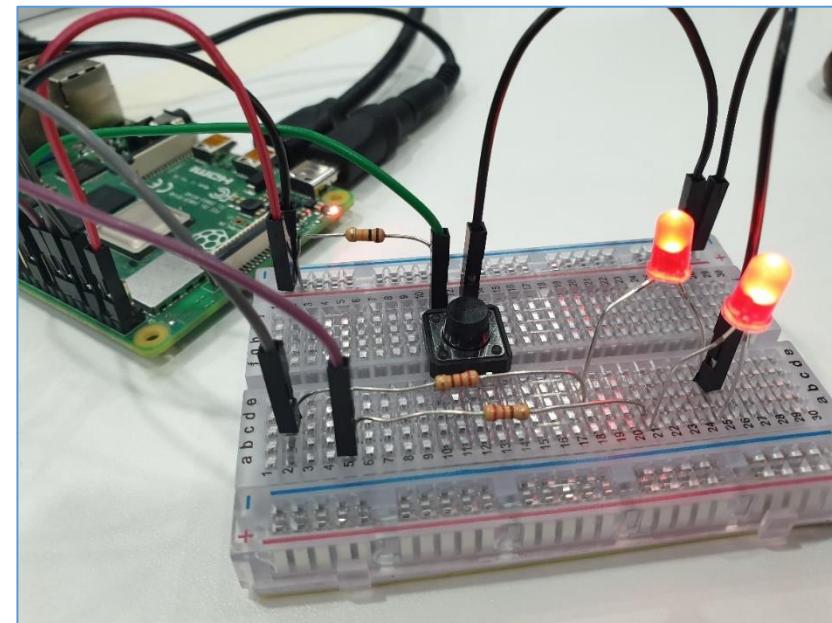
구현할 기능

- Download 버튼 : 다운로드 진행
- 다운로드가 되는 동안 APP 조작 가능



아래 기능을 만족하는 LED 조절기를 제작한다.

- 버튼 ON/OFF 로 LED1 ON / OFF
- 다이얼 조작으로 LED2 밝기 제어
- LCD 에 LED2 밝기 표시



사용한 H/W

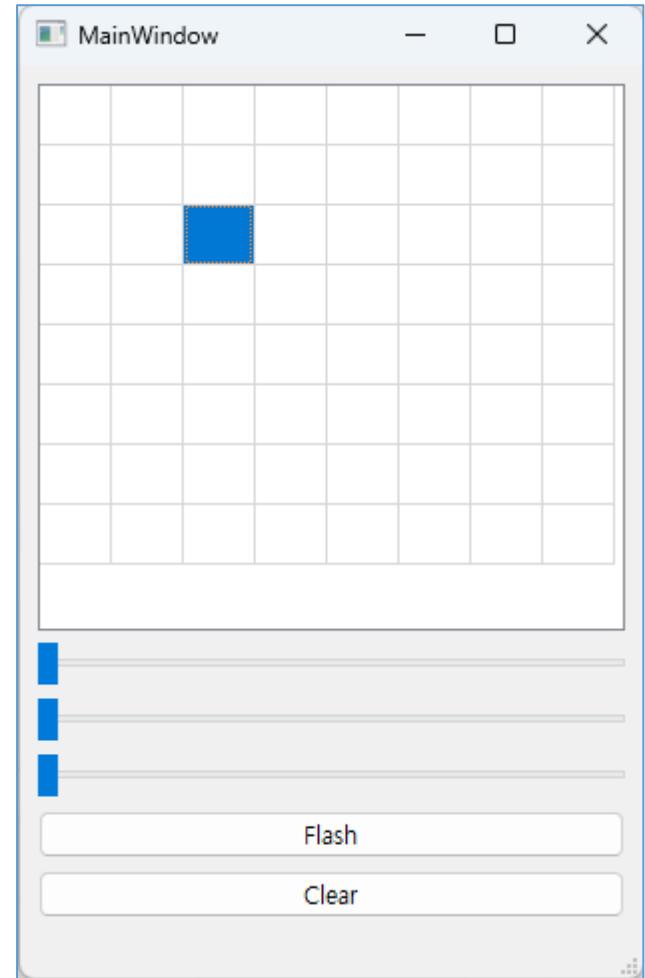
- senseHat

사용한 위젯

- Push Button, TableWidget, Slider

구현할 기능

- Flash 버튼 : 모든 LED가 켜진다.
- Clear 버튼 : 모든 LED가 꺼진다.
- Slider 움직이기 : 켜지는 LED의 R,G,B 값을 조절할 수 있다.
- table 클릭 : 선택 된 table 좌표의 senseHat LED가 켜진다.



객체 탐색기, 시그널/슬롯 편집기 (힌트)

객체 탐색기	
필터	
객체	클래스
MainWindow	QMainWindow
centralwidget	QWidget
verticalLayout_2	QVBoxLayout
clearBtn	QPushButton
flashBtn	QPushButton
sliderB	QSlider
sliderG	QSlider
sliderR	QSlider
table	QTableWidget
menubar	QMenuBar
statusbar	QStatusBar

시그널/슬롯 편집기			
+	-		
송신자	시그널	수신자	슬롯
sliderR	valueChanged(int)	MainWindow	rvalue(int)
sliderG	valueChanged(int)	MainWindow	gvalue(int)
flashBtn	clicked()	MainWindow	flash()
table	cellEntered(int,int)	MainWindow	click(int,int)
table	cellPressed(int,int)	MainWindow	click(int,int)
clearBtn	clicked()	MainWindow	clear()
sliderB	valueChanged(int)	MainWindow	bvalue(int)

속성 편집 (힌트)

- **table**

- rowCount / columnCount : 8
- noEditTriggers : True
- selectionMode : SingleSelection
- verticalHeaderVisible : False
- horizontalHeaderVisible : False
- horizontalHeaderDefaultSectionSize = 36

- **slider (R / G / B)**

- maximum : 255

내일
방송에서
만나요!

삼성 청년 SW 아카데미