QTimer

PyQT를 활용한 GUI 프로그래밍

QTimer

- Qt에서 주기적인 작업을 수행할 때 QTimer 클래스 사용
  - timeout 이벤트가 발생할 때 동작할 함수를 연결
  - timer.timeout.connect(함수)

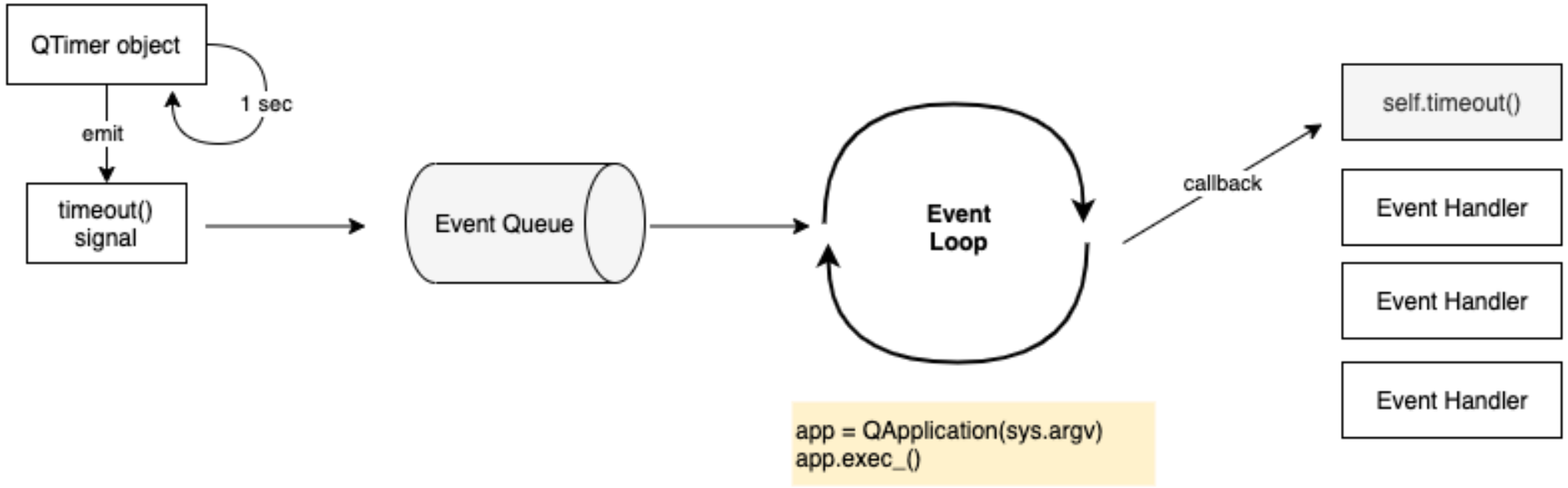
```
class MyWindow(QMainWindow):
    def __init__(self):
        super().__init__()

        timer = QTimer(self)
        timer.start(1000)          # 1 sec
        timer.timeout.connect(self.display_value)

    def display_value(self):
        print("hello")
```

QTimer와 timeout 이벤트 처리

- timeout signal을 emit 하면 이벤트 루프는 지정된 slot을 callback



QTimer 예제

- 다음은 1초에 숫자를 1씩 증가하면서 출력하는 예제

```
class MyWindow(QMainWindow):
    def __init__(self):
        super().__init__()

        self.num = 0

        timer = QTimer(self)
        timer.start(1000)          # 1 sec
        timer.timeout.connect(self. display_value)

    def display_value(self):
        print(self.num)
        self.num += 1
```

연습 문제

- 이전 예제에서 증가하는 숫자를 QLabel에 출력하라.

일회용 타이머

- 타임아웃 시그널이 단 한 번 발생하는 타이머
  - 지정된 시간이 경과했음을 알리는 타이머
  - singleShot 메서드를 사용하면 보다 짧은 코드로 사용 가능

```
class MyWindow(QMainWindow):
    def __init__(self):
        super().__init__()

        QTimer.singleShot(5000, self.timeout)

    def timeout(self):
        msg = "프로그램 실행 후 5초 경과"
        print(msg)
```

타이머와 스레드

- 실행 프로그램(프로세스)는 하나 이상의 스레드를 갖는데 이를 메인 스레드라고 함

```
class MyWindow(QMainWindow):
    def __init__(self):
        super().__init__()

        timer = QTimer(self)
        timer.start(1000)
        timer.timeout.connect(self.timeout)

    def timeout(self):
        cur_thread = threading.currentThread()
        thread_name = cur_thread.getName()
        msg = f"name: {thread_name}"
        self.statusBar().showMessage(msg)
```

타이머 사용 시 주의 사항

- timeout 이벤트는 메인 스레드가 처리
  - timeout 이벤트 처리에 시간이 오래 걸리는 경우 메인 스레드는 다른 이벤트를 처리할 수 없음
  - 이 경우 이벤트 루프가 멈추고 프로그램이 종료될 수 있음

```
class MyWindow(QMainWindow):
    def __init__(self):
        super().__init__()

        timer = QTimer(self)
        timer.start(1000)
        timer.timeout.connect(self.timeout)

    def timeout(self):
        print("before sleep")
        time.sleep(5)
        print("after sleep")
```

타이머 vs. QThread

- 타이머에서 sleep을 사용하면 메인 스레드도 정지 됨
- QThread로 별도의 스레드를 추가해야 함