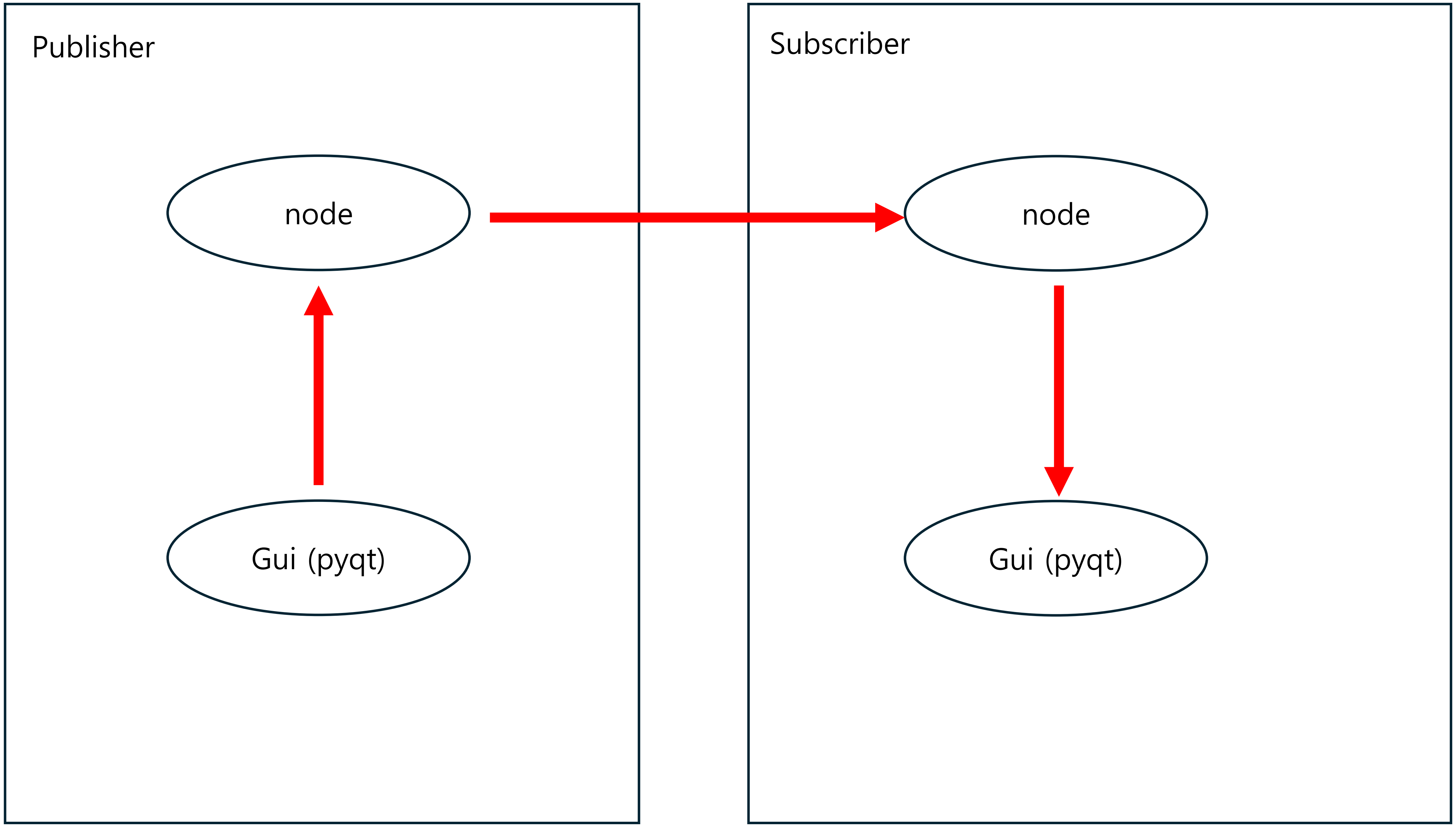
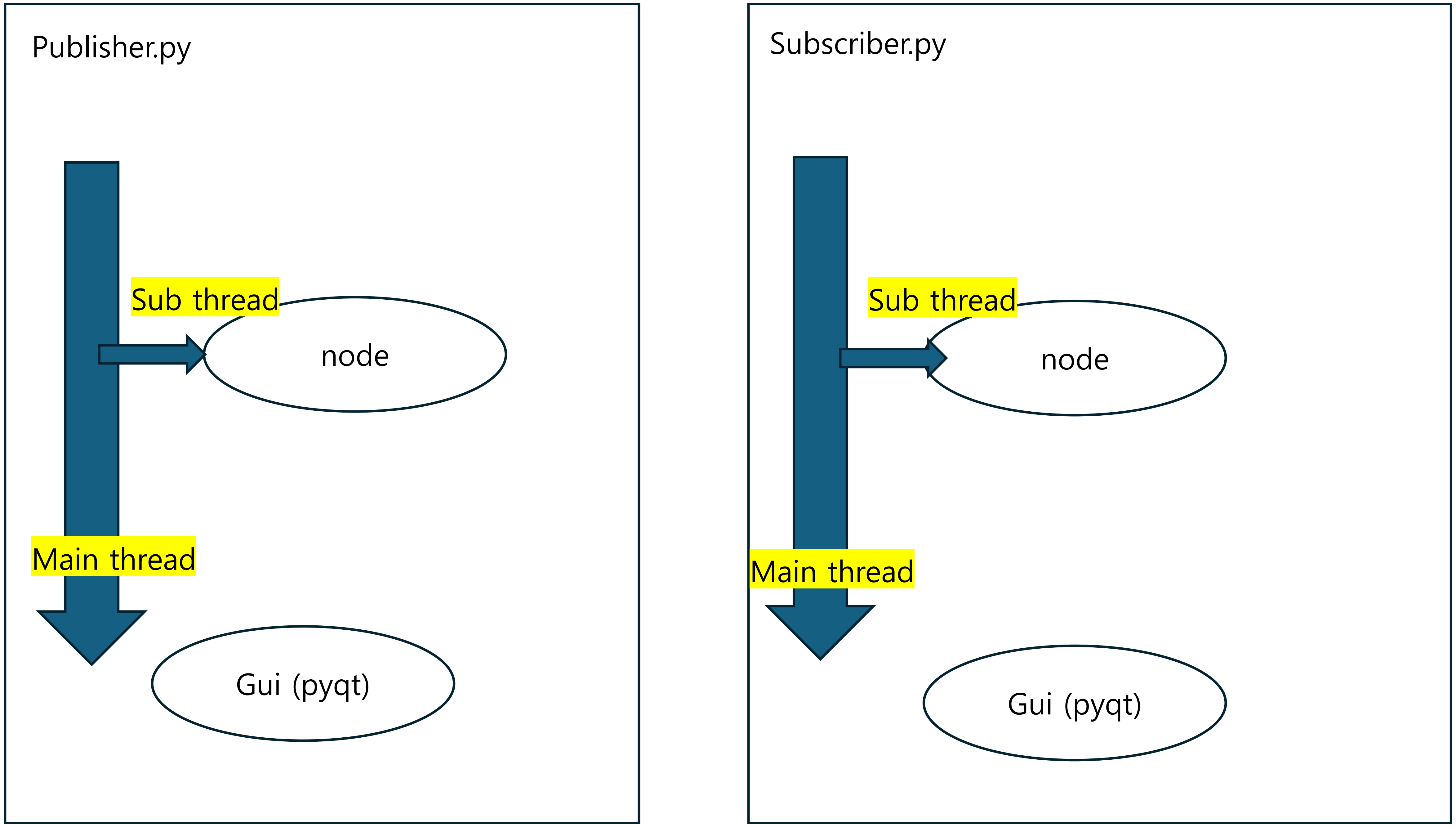
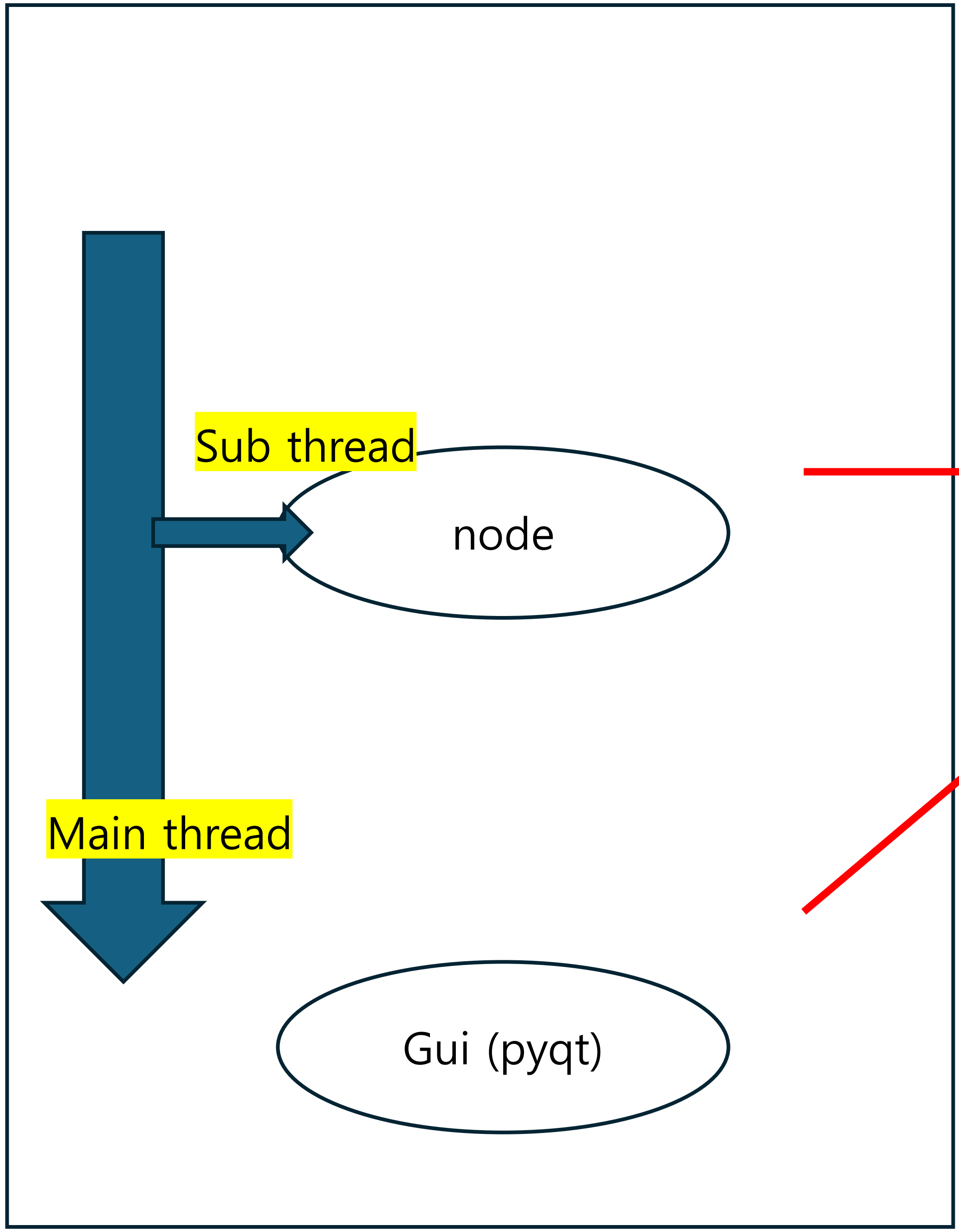
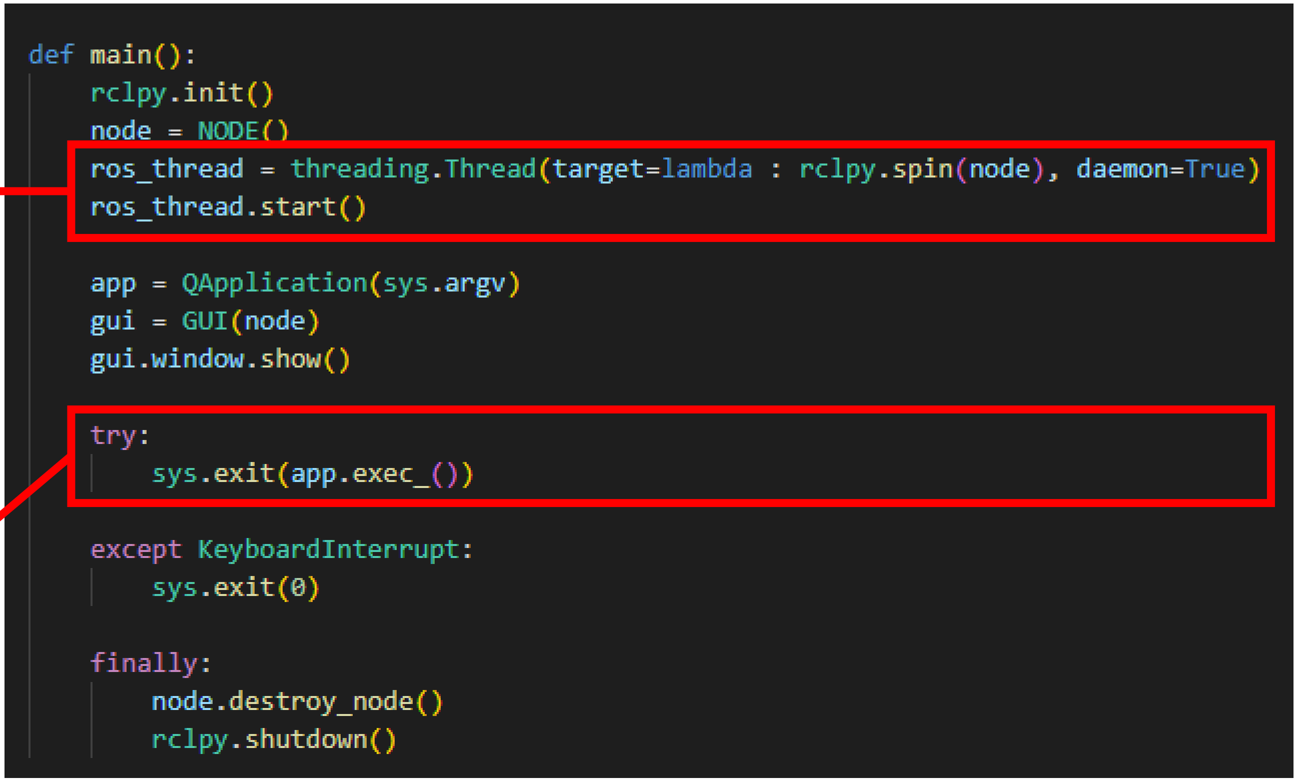
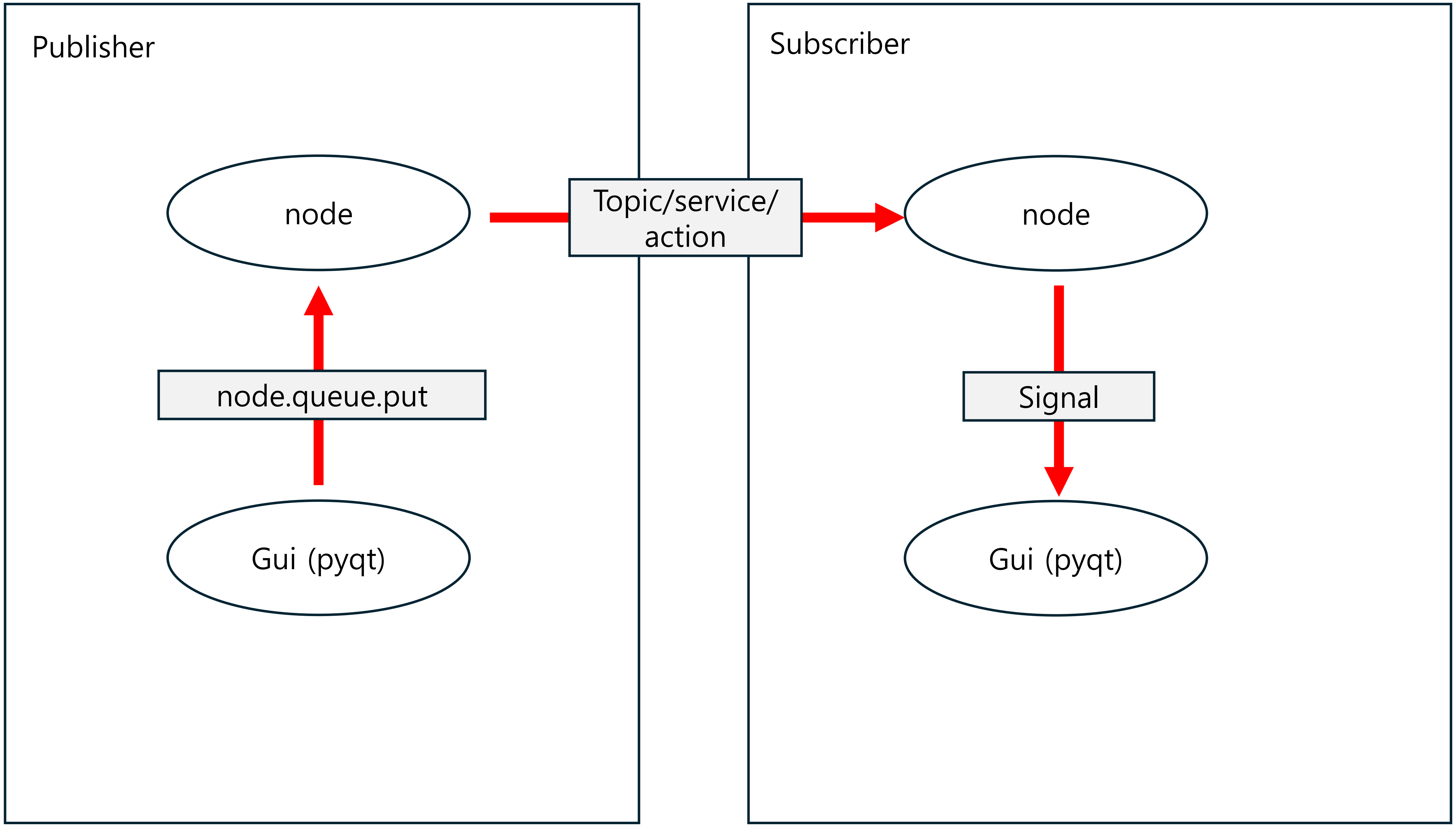

Publisher
node
Gui (pyqt)
Subscriber
node
Gui (pyqt)


Publisher.py
Sub thread
node
Main thread
Gui (pyqt)
Subscriber.py
Sub thread
node
Main thread
Gui (pyqt)

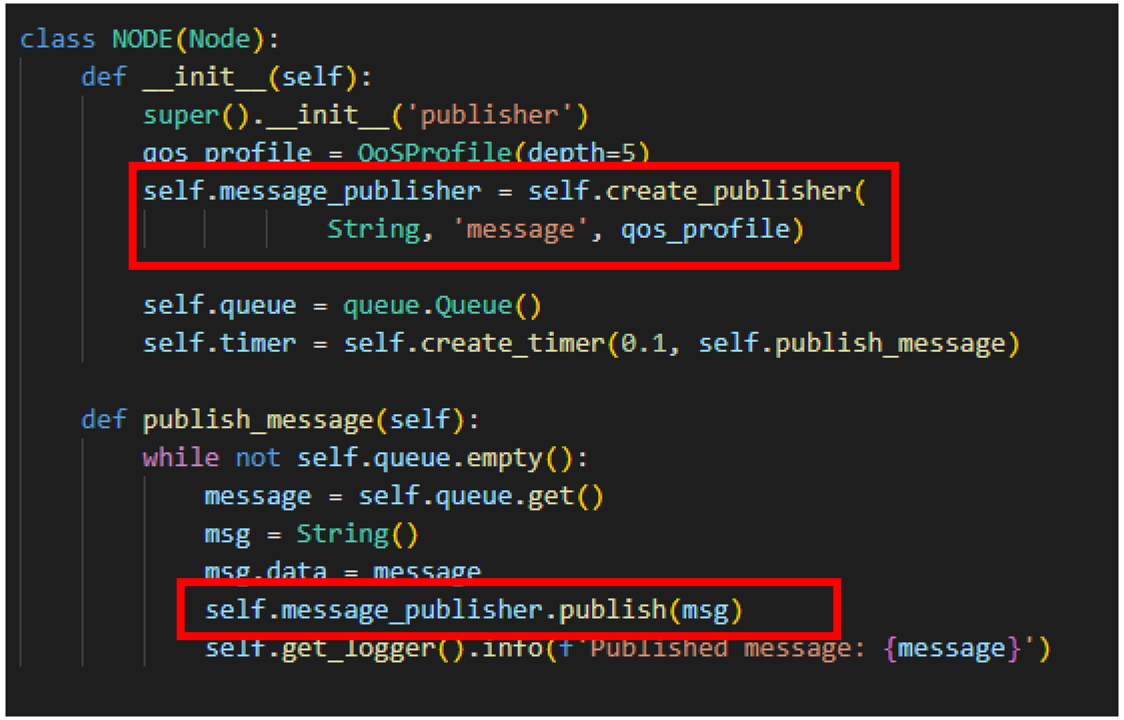




```
def main():
    rclpy.init()
    node = NODE()
    ros_thread = threading.Thread(target=lambda : rclpy.spin(node), daemon=True)
    ros_thread.start()

    app = QApplication(sys.argv)
    gui = GUI(node)
    gui.window.show()

    try:
        sys.exit(app.exec_())

    except KeyboardInterrupt:
        sys.exit(0)

    finally:
        node.destroy_node()
        rclpy.shutdown()
```

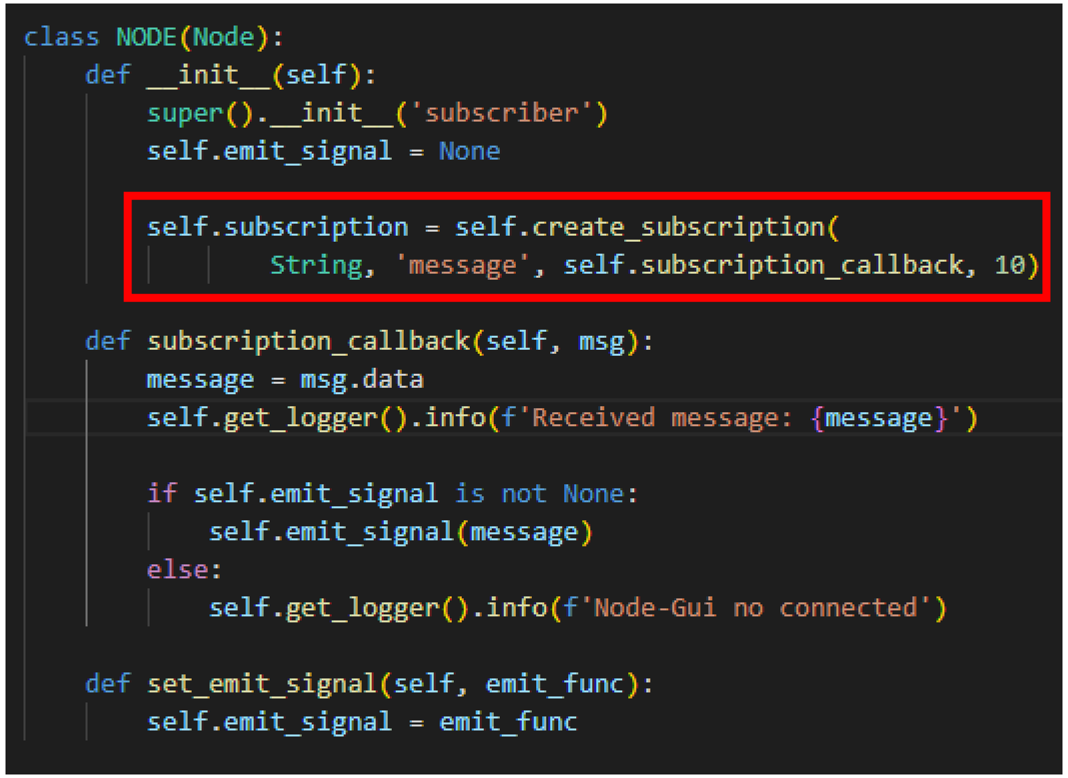

Publisher
node
node.queue.put
Gui (pyqt)
Topic/service/
action
Subscriber
node
Signal
Gui (pyqt)

Publisher



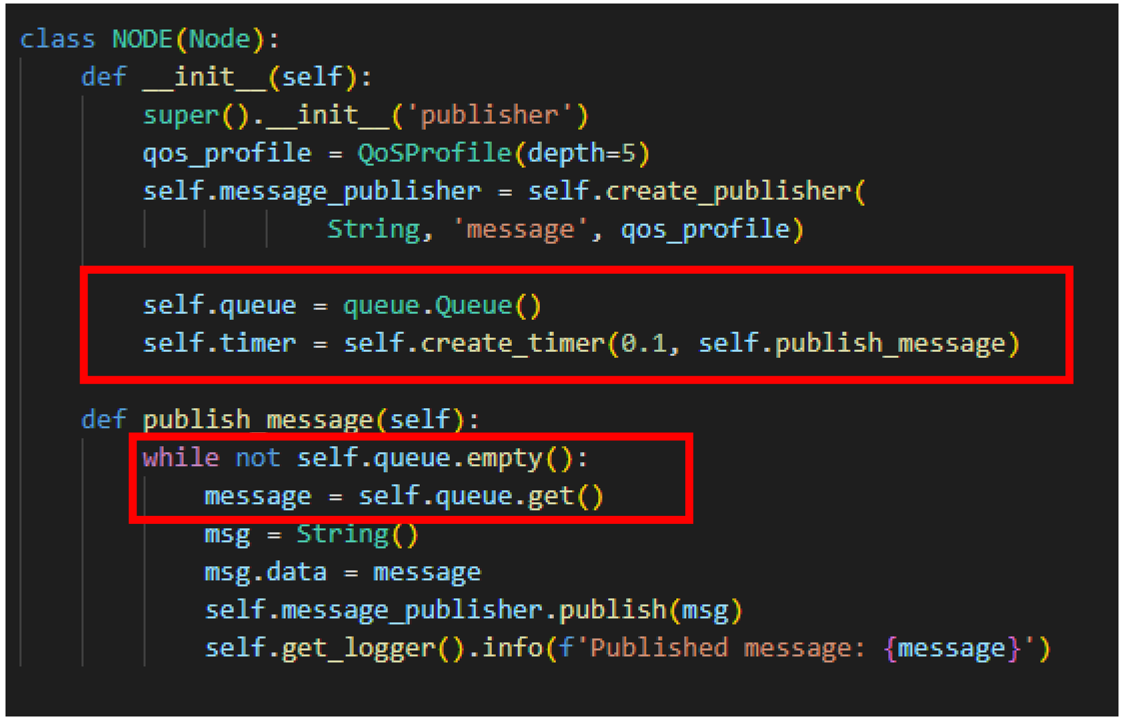
```python
class NODE(Node):
    def __init__(self):
        super().__init__('publisher')
        qos_profile = QoSProfile(depth=5)
        self.message_publisher = self.create_publisher(
                    String, 'message', qos_profile)

        self.queue = queue.Queue()
        self.timer = self.create_timer(0.1, self.publish_message)

    def publish_message(self):
        while not self.queue.empty():
            message = self.queue.get()
            msg = String()
            msg.data = message
            self.message_publisher.publish(msg)
            self.get_logger().info(f'Published message: {message}')
```

Subscriber

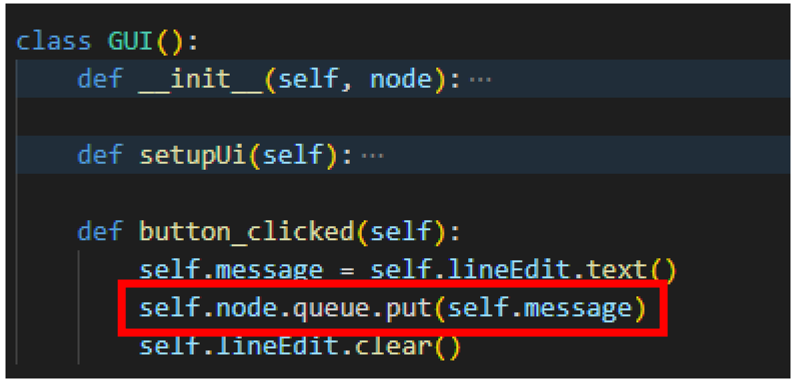


```python
class NODE(Node):
    def __init__(self):
        super().__init__('subscriber')
        self.emit_signal = None

        self.subscription = self.create_subscription(
                String, 'message', self.subscription_callback, 10)

    def subscription_callback(self, msg):
        message = msg.data
        self.get_logger().info(f'Received message: {message}')

        if self.emit_signal is not None:
            self.emit_signal(message)
        else:
            self.get_logger().info(f'Node-Gui no connected')

    def set_emit_signal(self, emit_func):
        self.emit_signal = emit_func
```

Publisher

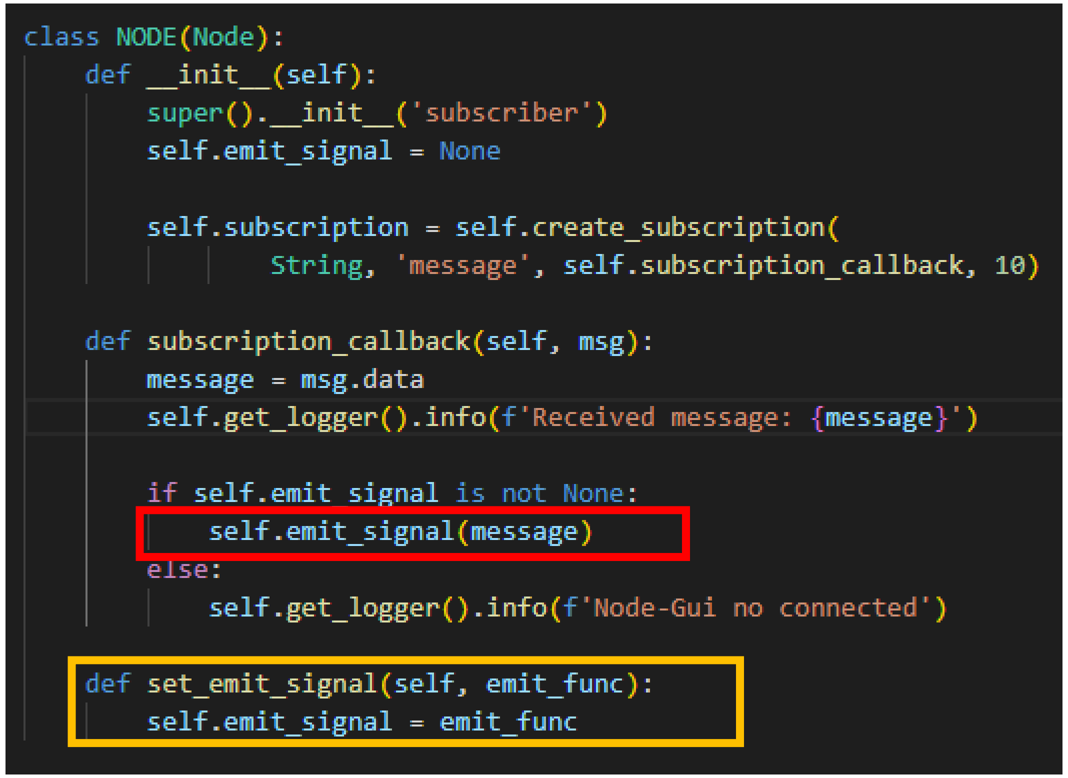


```python
class NODE(Node):
    def __init__(self):
        super().__init__('publisher')
        qos_profile = QoSProfile(depth=5)
        self.message_publisher = self.create_publisher(
                    String, 'message', qos_profile)

        self.queue = queue.Queue()
        self.timer = self.create_timer(0.1, self.publish_message)

    def publish_message(self):
        while not self.queue.empty():
            message = self.queue.get()
            msg = String()
            msg.data = message
            self.message_publisher.publish(msg)
            self.get_logger().info(f'Published message: {message}')
```

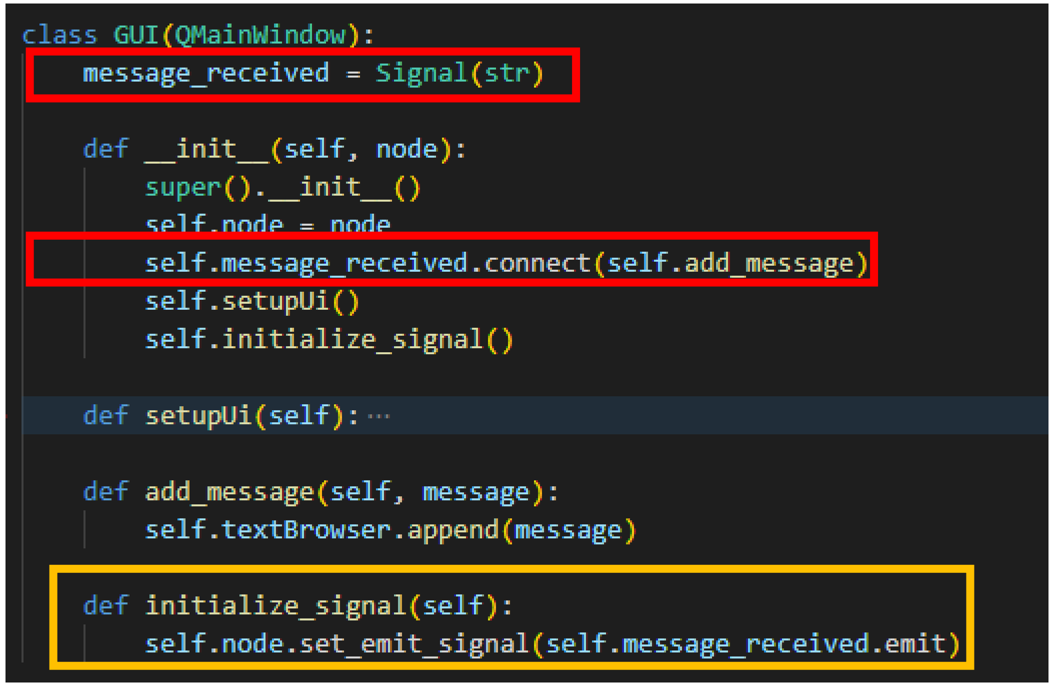


```python
class GUI():
    def __init__(self, node):···

    def setupUi(self):···

    def button_clicked(self):
        self.message = self.lineEdit.text()
        self.node.queue.put(self.message)
        self.lineEdit.clear()
```

Subscriber



```python
class NODE(Node):
    def __init__(self):
        super().__init__('subscriber')
        self.emit_signal = None

        self.subscription = self.create_subscription(
                String, 'message', self.subscription_callback, 10)

    def subscription_callback(self, msg):
        message = msg.data
        self.get_logger().info(f'Received message: {message}')

        if self.emit_signal is not None:
            self.emit_signal(message)
        else:
            self.get_logger().info(f'Node-Gui no connected')

    def set_emit_signal(self, emit_func):
        self.emit_signal = emit_func
```



```python
class GUI(QMainWindow):
    message_received = Signal(str)

    def __init__(self, node):
        super().__init__()
        self.node = node
        self.message_received.connect(self.add_message)
        self.setupUi()
        self.initialize_signal()

    def setupUi(self):...

    def add_message(self, message):
        self.textBrowser.append(message)

    def initialize_signal(self):
        self.node.set_emit_signal(self.message_received.emit)
```

```
$ sudo pip3 install PySide2
```

```
$ designer
```