Pose Subscriber Node

一、程式目的說明

本程式使用 **ROS 2 (rclpy)** 撰寫的 Python 訂閱者節點（Subscriber Node），該節點負責訂閱 `geometry_msgs/msg/Pose` 訊息，並將接收到的位姿資料輸出至終端機。
此程式會建立一個 ROS 2 節點：

- 節點名稱：`pose_subscriber_node`
- 訂閱 Topic：`turtle1/pose`
- 訊息型態：`geometry_msgs/msg/Pose`
- 功能：
- 接收位姿資料（Position 與 Orientation）
- 透過 logger 即時顯示接收到的內容

二、完整程式碼

```
#!/usr/bin/env python3
import rclpy
from rclpy.node import Node
from geometry_msgs.msg import Pose

class PoseSubscriberNode(Node):
    def __init__(self):
        super().__init__('pose_subscriber_node')
        self.subscription = self.create_subscription(
            Pose,
            "turtle1/pose",
            self.pose_callback,
            10)
        self.subscription  # prevent unused variable warning
        self.get_logger().info('Pose Subscriber Node has been started.')

    def pose_callback(self, msg):
        self.get_logger().info(
            f'Received Pose - Position: (x: {msg.position.x}, y:
{msg.position.y}, z: {msg.position.z}), '
            f'Orientation: (x: {msg.orientation.x}, y: {msg.orientation.y}, z:
{msg.orientation.z}, w: {msg.orientation.w})'
```

```
        )

def main(args=None):
    rclpy.init(args=args)
    pose_subscriber_node = PoseSubscriberNode()
    rclpy.spin(pose_subscriber_node)
    rclpy.shutdown()
```

三、程式碼逐段解析

1. Shebang 與匯入模組

```
#!/usr/bin/env python3
import rclpy
from rclpy.node import Node
from geometry_msgs.msg import Pose
```

說明：

- `#!/usr/bin/env python3`
  指定此檔案使用 Python 3 執行。
- `rclpy`
  ROS 2 的 Python Client Library。
- `Node`
  ROS 2 中所有節點的基底類別。
- `Pose`
  來自 `geometry_msgs` 套件的訊息型態，包含位置與方向資訊。

---

2. 定義訂閱者節點類別

```
class PoseSubscriberNode(Node):
    def __init__(self):
        super().__init__('pose_subscriber_node')
```

說明：

- 建立 `PoseSubscriberNode` 類別，繼承自 `Node`。
- 呼叫父類別建構子並設定節點名稱為 `pose_subscriber_node`。

3. 建立 Subscription

```
self.subscription = self.create_subscription(
    Pose,
    "turtle1/pose",
    self.pose_callback,
    10)
```

參數說明：

1. `Pose`
   訂閱的訊息型態。
2. `"turtle1/pose"`
   Topic 名稱。
3. `self.pose_callback`
   當收到訊息時呼叫的 callback 函式。
4. `10`
   QoS queue size（佇列大小）。
   此設定代表節點會持續監聽 `turtle1/pose` topic，並在有新資料時執行 callback。

---

4. Logger 啟動訊息

```
self.get_logger().info('Pose Subscriber Node has been started.')
```

說明：

- 使用 ROS 2 內建 logger 輸出節點啟動訊息。
- 有助於除錯與系統監控。

---

5. Callback 函式

```
def pose_callback(self, msg):
    self.get_logger().info(
        f'Received Pose - Position: (x: {msg.position.x}, y: {msg.position.y},
z: {msg.position.z}), 'f'Orientation: (x: {msg.orientation.x}, y:
```

```
{msg.orientation.y}, 'f'z: {msg.orientation.z}, w: {msg.orientation.w})'
    )
```

說明：

- `msg` 為接收到的 `Pose` 訊息物件。
- `msg.position`
- `x`, `y`, `z`：空間位置。
- `msg.orientation`
- `x`, `y`, `z`, `w`：四元數（Quaternion）表示的姿態。
- 將所有資訊格式化後輸出至終端機。

---

6. 主程式進入點

```
def main(args=None):
    rclpy.init(args=args)
    pose_subscriber_node = PoseSubscriberNode()
    rclpy.spin(pose_subscriber_node)
    rclpy.shutdown()
```

流程說明：

1. `rclpy.init()`
   初始化 ROS 2 通訊。
2. 建立節點實例。
3. `rclpy.spin()`
   讓節點持續運行並等待 callback 觸發。
4. `rclpy.shutdown()`
   程式結束時安全關閉 ROS 2。

四、執行方式說明

1. 確保已安裝 ROS 2 並正確 source 環境。
2. 將此檔案放入 ROS 2 package 的 `scripts` 或 `src` 目錄。
3. 賦予執行權限：

```
chmod +x pose_subscriber.py
```

4. 執行節點：

```
ros2 run <package_name> pose_subscriber.py
```

五、學習重點整理

- ROS 2 Node 的基本結構
- Topic 與 Subscriber 的建立方式
- Callback 機制的運作原理
- `geometry_msgs/Pose` 訊息結構

六、延伸練習建議

- 改為訂閱其他訊息型態（如 `Twist`）。
- 將接收到的資料儲存成 log 檔。
- 搭配 Publisher 節點進行完整通訊測試。