Turtlesim Draw Circle

一、程式目的說明

本程式示範如何在 **ROS 2** 中撰寫一個 Python Node，透過 **Publisher** 發送 `geometry_msgs/Twist` 訊息到 `cmd_vel` topic，控制機器人以固定的線速度與角速度前進，形成畫圓的運動軌跡。

此範例常用於：

- 差速型移動機器人（Differential Drive）
- TurtleBot / Gazebo 模擬

二、完整程式碼

```
from rclpy.node import Node
from geometry_msgs.msg import Twist

class DrawCircleNode(Node):
    def __init__(self):
        super().__init__('draw_circle_node')
        self.publisher_ = self.create_publisher(Twist, 'cmd_vel', 10)
        timer_period = 0.1  # seconds
        self.timer = self.create_timer(timer_period,
self.send_velocity_command)
        self.get_logger().info('Draw Circle Node has been started.')

    def send_velocity_command(self):
        msg = Twist()
        msg.linear.x = 0.2   # 前進線速度（m/s）
        msg.angular.z = 0.5  # 角速度（rad/s），用來形成圓周運動
        self.publisher_.publish(msg)

def main(args=None):
    rclpy.init(args=args)
    draw_circle_node = DrawCircleNode()
    rclpy.spin(draw_circle_node)
    rclpy.shutdown()
```

三、程式架構說明

本程式採用 **物件導向（OOP）** 方式撰寫，主要包含：

- `DrawCircleNode` 類別（繼承自 `Node`）
- Publisher（發送 `Twist`）
- Timer（固定時間週期執行 callback）
- 主程式 `main()`

四、程式逐段講解

1. 匯入必要套件

```
from rclpy.node import Node
from geometry_msgs.msg import Twist
```

- `Node`：ROS 2 中所有節點的基底類別
- `Twist`：
- 用來描述機器人的線速度與角速度
- 常用於 `cmd_vel` topic

---

2. 建立 DrawCircleNode 類別

```
class DrawCircleNode(Node):
```

- 自訂一個 ROS 2 Node
- 繼承 `Node` 類別，才能使用 ROS 2 API

---

3. Node 初始化（`__init__`）

```
super().__init__('draw_circle_node')
```

- 呼叫父類別 `Node` 的建構子
- 設定 Node 名稱為 `draw_circle_node`

4. 建立 Publisher

```
self.publisher_ = self.create_publisher(Twist, 'cmd_vel', 10)
```

- Topic 名稱：`cmd_vel`
- 訊息型態：`Twist`
- Queue size：10
  📌 大多數移動機器人都會訂閱 `cmd_vel` 來接收速度指令

---

5. 建立 Timer

```
timer_period = 0.1
self.timer = self.create_timer(timer_period, self.send_velocity_command)
```

- Timer 週期：0.1 秒（10 Hz）
- 每 0.1 秒自動呼叫 `send_velocity_command()`

---

6. Logger 訊息

```
self.get_logger().info('Draw Circle Node has been started.')
```

- 使用 ROS 標準 logger
- 顯示 Node 啟動成功

---

7. 發送速度指令

```
def send_velocity_command(self):
    msg = Twist()
    msg.linear.x = 0.2
    msg.angular.z = 0.5
    self.publisher_.publish(msg)
```

- `linear.x`：機器人前進速度（m/s）

- `angular.z`：機器人旋轉角速度（rad/s）
- 同時存在 → 機器人會做圓周運動

📐 圓的半徑約為：
`r = linear.x / angular.z = 0.2 / 0.5 = 0.4 m`

---

8. 主程式 main()

```
def main(args=None):
    rclpy.init(args=args)
    draw_circle_node = DrawCircleNode()
    rclpy.spin(draw_circle_node)
    rclpy.shutdown()
```

- `rclpy.init()`：初始化 ROS 2
- `rclpy.spin()`：讓 Node 持續運作（Timer 才會生效）
- `rclpy.shutdown()`：釋放 ROS 資源

五、執行方式（範例）

```
ros2 run <package_name> draw_circle_node
```

六、執行結果說明

- 機器人會持續前進並同時旋轉
- 路徑呈現「圓形」
- 停止程式（Ctrl＋C）後機器人停止

七、重點整理

- `Twist` 是移動機器人最常用的速度控制訊息
- Publisher＋Timer 是 ROS 2 中最典型的控制架構
- 線速度＋角速度＝圓周運動

- `cmd_vel` 是業界慣用的速度控制 topic

八、延伸練習

1. 修改 `linear.x` 或 `angular.z`，觀察圓半徑變化
2. 嘗試只設定 `angular.z`（原地旋轉）
3. 加入 Subscriber，用鍵盤控制速度
4. 加入停止條件（例如 10 秒後停止）