Problem Statement

- The objective of the task is to move the turtle inside the **turtlesim** window in a **vertical lemniscate**(infinity shape) and stop at its initial location. Refer the expected output to understand the maneuver.

Note: Turtle should first cover the below circle and then the upper circle as per the expected output section. Any other orientation/order may not be awarded a total score and may even score zero.

- Teams are supposed to do this by creating a nodes name, `/node_turtle_revolve` within a python script, `node_turtle_revolve.py` . Refer the prerequisites to learn the process.

Procedure

1. First, create a package named `pkg_task0` within your catkin workspace, using ROS package. If you haven't already created a workspace, refer section 1.4.2.1 to know how to do so. Once done, compile and source the packages. The file structuring can be understood from section 1.4.2

2. Within this package, create the `scripts` folder inside which you will have to create a python script named `node_turtle_revolve.py` . By entering the following command (assuming you are in `~/catkin_ws/pkg_task0` already).

```
mkdir scripts && cd scripts && touch node_turtle_revolve.py
```

Write the script with maintaining proper programming ethics. Doing this will help us understand your code better and quicker than usual.

3. Fill in the scripts `node_turtle_revolve.py` . Participants can refer ROS package section to get idea on how to proceed. After completing the **python3** script. Make it executable, if it is not already. To do that, enter the following code.

```
chmod +x ~/catkin_ws/src/pkg_task0/scripts/node_turtle_revolve.py
```

IMPORTANT NOTE:
The referred documentation about *ROS packages*, is in python 2, but remember, **ROS Noetic runs on python 3.** So do follow the learning resource, as it describes well written coding standard too, but don't forget to implement your Task in python 3 syntax.

4. Before executing, make sure that `roscore` is running along with `turtlesim_node` . You can either run them in separate terminals or simply create a `task0.launch` file inside the `~/catkin_ws/src/pkg_task0/launch/` folder. Creating launch is an easy task, participants can refer ROS launch file for hints, plus we have given some suggestions in the recording log section to make things easier for you for creating a valid submission.

Launch file can run multiple nodes, unlike a python/cpp script, at least in ROS. While in ROS2, this is not the case. Do explore it.

Run the launch file, enter,

```
roslaunch pkg_task0 task0.launch
```

- This should run three processes in parallel.
  - `roscore`
  - `turtlesim_node`
  - `node_turtle_revolve.py`

Hints

1. The turtle needs to move in lemniscate motion and divide them into two circular motions with the same radii.
   - This radius should be sufficient to fit within the turtlesim window. However, making it rotate circularly, with only velocities to control, is something to think about.
2. Use linear velocity as well as angular velocity with some combination to get this done.
3. Keep tracking the distance travelled to know when to stop. But do not limit yourself to this. Feel free to use any methods as you wish, as far as it's a valid submission. You can refer to Overview of rospy for more hints.

Next Read: Expected Output