# Purpose

In this project, you will exhibit a basic understanding of ROS2 package fundamentals and the fundamentals of driving a differential drive (diff-drive) robot.

## Task

You must create a small Python script that publishes ROS 2 `Twist` Messages to drive a simulated differential-drive robot (the iRobot Create 3). The robot's behavior must respond to simulated button presses as follows:

→ When simulated "Button 1" (left button) is pressed, the robot must drive in a counter-clockwise spiral and stop.

→ When simulated "Button 2" (right button) is pressed, the robot must drive in a clockwise spiral and stop.

→ When the simulated "Power" button is pressed, the robot must stop immediately.

- Your script must accept one **optional** input parameter, `spiral_scale` ($1.0 \leq$ `spiral_scale` $\leq 2.0$), which defaults to 1.0. Doubling this parameter doubles the end radius of the spiral; a value of 1.5 increases the radius by approximately 50%.
- At the default scalar (`spiral_scale` $= 1.0$), the spiral should terminate approximately at the robot's original outer edge, i.e., extending from $x = 0.0$ to $x \approx \pm 0.20\,\text{m}$ ($\pm 8\,\text{in}$).

After building with `colcon` and sourcing the resulting `install/setup.bash` overlay script. Your package must run as

```
ros2 run drive_spiral main [--ros-args -p spiral_scale:=1.0]
```

# Deliverables

You must provide exactly one file: `main.py`. That is the same file from:

```
drive_spiral/
+-- drive_spiral/
|   +-- main.py
```

# Points

Your code will be tested using the `create3_gazebo` simulation using the R-Viz application and with the dock node deleted/removed.

I will (and will not) provide parameters for `spiral_scale`. You will earn points based on:

## Correctness

`drive_spiral/drive_spiral/main.py` 25% each:

A correct script must publish Twist messages to `cmd_vel` without crashing, ever.

A correct script must subscribe to and correctly accept String messages from `proj1/spiral_cmd` without crashing, ever.

### Performance

`create3_gazebo`

Correctly spiraling in the correct direction. 30%

Ending in the roughly the correct ending position, i.e. facing the same direction as it started and left of its original position if CW and right if CCW. 10%

Correctly increasing spiral magnitude, i.e., 10%

$x \pm 8$in if `spiral_scalar` $== 1.0$
$x \pm 16$in if `spiral_scalar` $== 2.0$.

# Appendix

Useful ROS2 CL commands:

## 1.) Run Project 1a

From the terminal, after sourcing your `install/setup.bash`, use

```
ros2 run drive_spiral main
```

Use `CTRL + c` to halt.

## 2.) Start iRobot Create Gazebo Simulation

```
ROS_DOMAIN_ID=232  \
ros2 launch irobot_create_gazebo_bringup create3_gazebo.launch.py
```

## 3.) Send Button Presses

From the command line, you may simulate button presses using:

### Button 1

```
ROS_DOMAIN_ID=232  \
ros2 topic pub --once proj1/interface_buttons  \
irobot_create_msgs/msg/InterfaceButtons "{  \
  button_1: {is_pressed: true},  \
  button_power: {is_pressed: false},  \
  button_2: {is_pressed: false}  \
}"
```

### Button Power

```
ROS_DOMAIN_ID=232 ros2 topic pub --once  \
proj1/interface_buttons  \
irobot_create_msgs/msg/InterfaceButtons "{  \
  button_1: {is_pressed: false},  \
  button_power: {is_pressed: true},  \
  button_2: {is_pressed: false}  \
}"
```

Cont.

**Button 2**

```
ROS_DOMAIN_ID=232 ros2 topic pub --once  \
proj1/interface_buttons  \
irobot_create_msgs/msg/InterfaceButtons "{  \
  button_1: {is_pressed: false},  \
  button_power: {is_pressed: false},  \
  button_2: {is_pressed: true}  \
}"
```

The End.