AA 274A: Principles of Robot Autonomy I
Section 3: Introduction to Turtlebot and Gazebo

Our goals for this section:

1. Gain a basic understanding of the Turtlebot software.
2. Use basic tools for interacting with the Turtlebot.

1 The Turtlebot Software

Most of the forward-facing Turtlebot software you will work with is located in the `asl_turtlebot` repository on Github. To get it, go to `~/catkin_ws/src` and run:

```
git clone https://github.com/StanfordASL/asl_turtlebot.git
```

Since we downloaded a new catkin package, we need to rebuild the workspace by running the following from the `~/catkin_ws` directory.

```
catkin_make
```

1.1 Turtlebot bring up

Once logged in to Genbu, start roscore:

```
roscore -p $ROS_PORT
```

In a new terminal window, run the Gazebo environment:

```
roslaunch turtlebot3_gazebo turtlebot3_world.launch
```

Problem 1: Once this is all running, which rostopics are available? Paste this list in your submission.

2 TurtleBot Teleoperation

Now, let's explore teleoperation with the TurtleBot.

1. We can start exploring the existing ROS topics. What are all the messages that are being published right now? In particular, look at the odom topic. What is the message type being published to this topic and what information is contained within these messages? HINT: rostopic info odom might help.
2. In a new terminal window, begin teleoperating the robot by running:

```
roslaunch turtlebot3_teleop turtlebot3_teleop_key.launch
```

3. Try to teleop the TurtleBot back to $(0, 0, 0)$.

Problem 2: What is the message type being published to `odom` and what information is contained within these messages?

2.1 Pub to cmd_vel

Using our code from last week's section, create a publisher that publishes to the `cmd_vel` topic and sends a zero velocity signal at every timestep. The skeleton code for this included in this week's code in the `vel_publisher.py` file. In particular, you should send out a message of type `geometry_msgs/Twist`, with information for how to populate it available online. Some resources that help are the ROS documentation on it as well as our own TurtleBot code (look at line 155).

Problem 3: Paste your code in your submission, as well as any of its running output.

2.2 Sub to odom

Similarly, create a subscriber that subscribes to the odom topic and prints out what it receives. The skeleton code for this is located in the `odometry_subscriber.py` file.

Problem 4: Paste your code in your submission, as well as any of its running output.

Cleanup

Towards the end of your section, when you're about to log out, please shut down all of your running processes (like roscore or any publishers/subscribers) and clean up your catkin workspaces for the next groups. In particular, remove the code you wrote for the section as well as any catkin packages you created for the section within `catkin_ws/src`.