Exercises

1. **Performance**: Let's start with the typical ROS example of the publisher and subscriber but a bit overcharged.
2. **Canny-edge your face:** Get images from a camera and apply a canny edge filter to them.

Exercise 1: Performance

The idea is to create ONE package called "performance_tests" with 4 files containing a node each. You will have to implement: (1) One simple publisher written in C++, (2) One simple publisher written in Python, (3) One simple subscriber written in C++ and (4) One simple subscriber written in Python.

The publishers publish a customized message called "SuperAwesome.msg" that just contains a string. The subscribers subscribe to the topic. You have freedom to choose whatever topic name.

The interesting part comes now. You might have heard that C++ is faster than Python but, have you ever see it with you own eyes? You will have to change the loop rate of the publisher somehow (i.e. simple parameter, reconfigurable parameter, service...) to change the speed at which it publishes messages. The subscriber nodes will of course receive the messages at different speeds and thus, you can measure if the desired loop rate is as in reality.

There are 4 different combinations:

- C++ publisher to Python subscriber
- C++ publisher to C++ subscriber
- Python publisher to Python subscriber
- Python publisher to C++ subscriber

You will have to perform the performance test to find our where is approximately the limit at which the desired rate does not coincide with real life for the 4 combinations. You are free to choose how to show this information (i.e. just the number, a graph, picture...).

Exercise 2: Canny-edge your face

This will tests your ability with OpenCV. The idea is to grab images with your computer's camera and apply a canny edge filter to the frames. To publish images from the camera you can use the node "cv_camera" (thus you will save time, resources and pain).

Create a new package called "canny_edge_my_face" that subscribes to the raw images from the camera and apply a canny edge to them. Then publish that image so you can see it with rqt. You can use whatever programming language you want.