简单的Turtlesim模拟

目标:

你将了解简单的消息发布-接收机制，从turtlesim节点接收信息，并发布回turtlesim节点以控制海龟。

任务:

练习1:

我们在四个终端运行以下命令启动并驱动一个 turtlebot，

```
% roscore
% rosrun turtlesim turtlesim_node
% rosrun turtlesim turtle_teleop_key
% rqt_graph
```

我们可以通过键盘控制海龟，当然也可以用 python 或 C++编程控制它。

你也可以通过命令行加入一只新海龟。

```
% rosservice call /spawn 2 2 0.2 “”
% rosnode info /turtlesim
% rostopic pub /turtle1/cmd_vel geometry_msgs/Twist -r 10 -- '[2.0, 0.0, 0.0]' '[0.0, 0.0, 0.0]'
```

以上命令中的参数：

- `/turtle1/cmd_vel` 是 TurtleBot 接收速度指令的话题。
- `geometry_msgs/Twist` 是用于发布速度指令的消息类型。
- `-r 10` 表示每秒发布十次消息。
- 接下来的两个列表分别代表线速度（`Linear`）和角速度（`Angular`）

此时打开可视化图，会看到新增的消息等。

```
% rqt_graph
```

现在，进入你的 catkin 空间的 src 子目录，用 git 获取一个 stack，在此 stack 中包含了 4 个 packages。

```
% cd ~/catkin_ws/src
% git clone https://github.com/Zhijun2/roslab1.git
% roscd autoturtle && cd scripts
```

进入 scripts 子目录后，分析 python 程序 turtlesim_move.py 和 turtlesim_move1.py，此二个程序分别使用 python class 和 functions 编写，但是都做同一件事情: 生成 ROS 节点，发布消息 /turtle1/cmd_vel 去驱动海龟沿一条直线以固定速度运动。请用 rosrun 运行这二个程序(你可尝试改变速度)，观察这只海龟如何移动。

练习 2 （课后作业）：
请花费一些时间考虑以下问题。

以程序 turtlesim_move1.py 为模板，修改此模板程序以产生一个新节点，完成以下二个任务。

(1) 用户输入海龟的线速度和角速度，驱动海龟以设定的速度行动（例如，沿圆弧运动）；
(2) 用户输入海龟需要运动的距离信息，使得海龟一旦达到该距离终点即停止运动。

请至少花费半个小时思考设计这个任务，尝试独立修改、测试代码。在花费足够时间解决此任务后，你可参考下面的解决方案。

http://wiki.ros.org/turtlesim/Tutorials/Moving%20in%20a%20Straight%20Line
http://wiki.ros.org/turtlesim/Tutorials/Rotating%20Left%20and%20Right
http://wiki.ros.org/turtlesim/Tutorials/Go%20to%20Goal

备注:不要仅拷贝黏贴代码，需要分析理解代码并进行实验。