Before writing source code

- package := namespace, structure, compiling unit

workspace
- build
- devel
- src
  - package_name
    - Cmakelists.txt
    - package.xml
    - src

catkin_init_workspace

Creating own package

```
catkin_create_pkg pkg_name <dependencies>
```

- CmakeLists.txt and package.xml are generated
- Source code should be located inside the folder of package
- To compile the source code use `catkin_make`

Creating own publisher

- The node that will publish the commands for turtlesim_node
- `rostopic info /turtle1/cmd_vel`
- `rosmsg show geometry_msgs/Twist`
- In python code

```
from geometry_msgs import Twist
```

Example of simple publisher in python

```python
#!/usr/bin/env python

import rospy
from geometry_msgs.msg import Twist

if __name__ == '__main__':
    message = Twist()
    message.linear.x = 1
    message.angular.z = 1

    rospy.init_node('pub_staff', anonymous=True)

    pub = rospy.Publisher('/turtle1/cmd_vel', Twist, queue_size=10)

    rate = rospy.Rate(1)
    while not rospy.is_shutdown():
        pub.publish(message)
        rate.sleep()
```

Creating own subscriber

- To subscribe to a topic means to call a function that will handle the messages.
- Calling a function is a task of ROS — it has a threadpool for dealing with the message queue.
- You need to point a callback for a function.

Templates for publisher and subscriber in C++

```
#include <ros/ros.h>
#include <geometry_msgs/Twist.h>

void reader (const geometry_msgs::Twist & message) {
        // handle message here
}

int  main (int argc, char **argv) {


        ros::init(argc, argv, "writer");

        ros::NodeHandle n;
        ros::Publisher pub = n.advertise<geometry_msgs::Twist>("Name", 10);

        ros::Subscriber sub = n.subscribe("Name", 10, reader);

        ros::spin();
        ROS_INFO("Publishing is finished! \n");
        return 0;
}
```

How to edit CmakeListst.txt

```
cmake_minimum_required(VERSION 2.8.3)
project(<project_name>)
find_package(catkin REQUIRED COMPONENTS
    roscpp
)
catkin_package()
include_directories(include ${catkin_INCLUDE_DIRS})
add_executable(reader src/reader.cpp)
add_executable(writer src/writer.cpp)
target_link_libraries(reader ${catkin_LIBRARIES})
target_link_libraries(writer ${catkin_LIBRARIES})
```

Creating own message

- It is possible to create a package that consists of message and doesn`t contain any executable nodes.
- Package that uses this message should have corresponding dependecies in CmakeLists.txt and package.xml.