Lecture 07.02 Subscribing to topics

Subscribing to topics with `rospy` involves two steps:

callback

1. defining a *callback* function that is called every time a message arrives (on the topics specified in a moment) and
2. registering the subscription with `roscore`.

The name of the callback function can be anything—say, `callback`, but its argument should be handled as a message of the correct type (i.e. the message type of the topic to which we are subscribing). Registering the subscription with `roscore` is accomplished with the `Subscriber` method as follows.

```
rospy.Subscriber(
  <topic name string>,        # e.g. 'cool_topic_bro'
  <message_type>,             # e.g. Int32 from std_msgs
  <callback function handle>  # e.g. callback
)
```

The first two arguments are the same as those of `rospy.Publisher`. The final argument is simply the name of the callback function from above.

07.02.1 Creating a simple subscriber node

The code accompanying the text has a simple subscriber node in the `rico_topics` package. You should use have used `catkin_create_pkg` in Lecture 07.01 to create a parallel package in your own code repository—we'll call it `my_topics`. Create a new Python file in `my_topics/src` with the following.

```
touch my_topics/src/topic_subscriber.py     # create file
chmod u+x my_topics/src/topic_subscriber.py # make executable
```

Open the empty `topic_subscriber.py` in a text editor. You'll want to enter here the same code as appears in the sample `topic_subscriber.py` from `robotics-book-code/rico_topics/src`, which is listed in Figure 07.2.

We see that the callback function definition `def callback(msg)` simply `print`s the message's data to the Terminal running the node. The call to to `rospy.Subscriber` register's (with `roscore`) this node's

```
#!/usr/bin/env python
import rospy
from std_msgs.msg import Int32

def callback(msg):   # callback for receiving messages
  print(msg.data)    # print to Terminal

rospy.init_node('topic_subscriber') # initialize node

sub = rospy.Subscriber('counter', Int32, callback) # subscribe

rospy.spin() # wait for node to be shut down
```

Figure 07.2: `rico_topics/src/topic_subscriber.py` listing.

subscription to the topic `'counter'`, with its message type `Int32`, and directs messages to the callback function `callback`, just defined.

Finally, there's a call to `rospy.spin`. This function here acts to keep the node running (so it can receive messages) until it is explicitly shut down. It's doing something like the following.

```
while not rospy.core.is_shutdown():
  rospy.rostime.wallsleep(0.5) # seconds
```

07.02.2 Running and verifying the node

Now that we have created `my_topics/src/topic_subscriber.py`, we need to `catkin_make` and `source` our workspace.

```
cd ros_ws_01 # if needed
```

```
catkin_make
```

Now we can source our workspace.

```
source devel/setup.bash
```

Now, make sure you've started a `roscore` service running. If not, start it with the following.

```
roscore
```

Also make sure you still have the `topic_publisher.py` node running. If not, start it with the following.

```
rosrun my_topics topic_publisher.py
```

And now we're ready to launch the new `topic_subscriber.py` node.

```
rosrun my_topics topic_subscriber.py
```

```
100
101
102
```

The terminal prints the counter, as expected. To see who's publishing and subscribing to `counter`, we can use `rostopic` as follows.

```
rostopic info counter
```

```
Type: std_msgs/Int32

Publishers:
 * /topic_publisher (http://socrates:35309/)

Subscribers:
 * /topic_subscriber (http://socrates:40387/)
```

Just as we expected: `topic_publisher` is publishing to and `topic_subscriber` is subscribed to the topic `counter`.

07.02.3 Latched topics

latched topic

Sometimes a topic will have messages published so infrequently that it could be problematic if a subscriber misses a message because it was not-yet subscribed to the topic. In this case, we can publish a *latched topic*, which makes it so that every new subscriber gets the last message published to the topic. Latched topics are created with the `rospy.Publisher` named argument `latched=True`, which is by default `False`.