Hochschule
Bonn-Rhein-Sieg
University of Applied Sciences

b-it Bonn-Aachen International Center for Information Technology

ROS Nodes, Topics, and Messages

Foundation Course

September 1, 2019

Hassan Umari

Recap

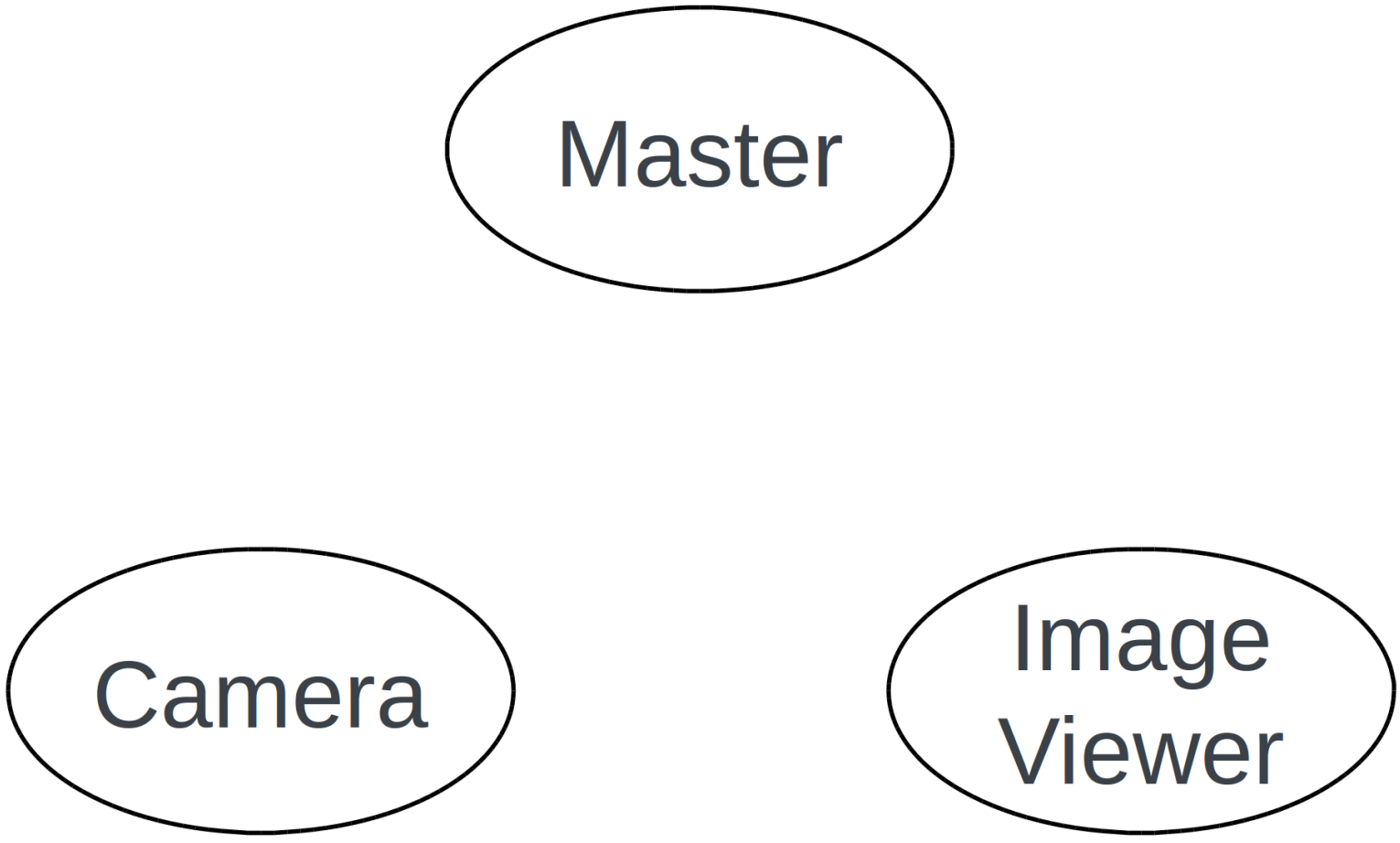
Summary of yesterday's session

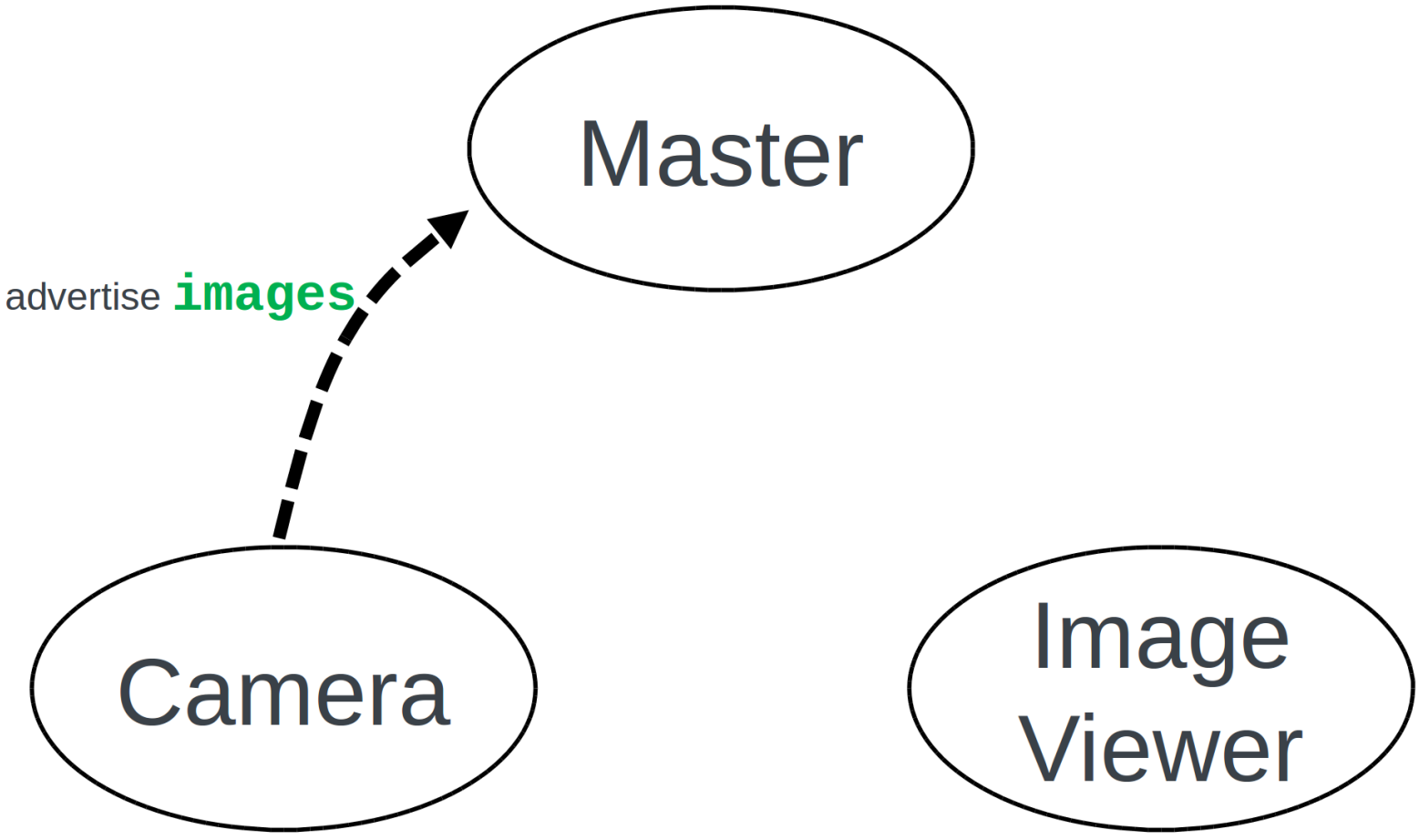
- ROS is a collection of libraries and tools that helps you when you develop software for robots.
- ROS provides several ways to transfer data between nodes:
  1. ROS topics and messages (**publish/subscribe**).
  2. ROS services (**request/reply**).
  3. ROS actions (**request/reply**).
  4. Parameter server.

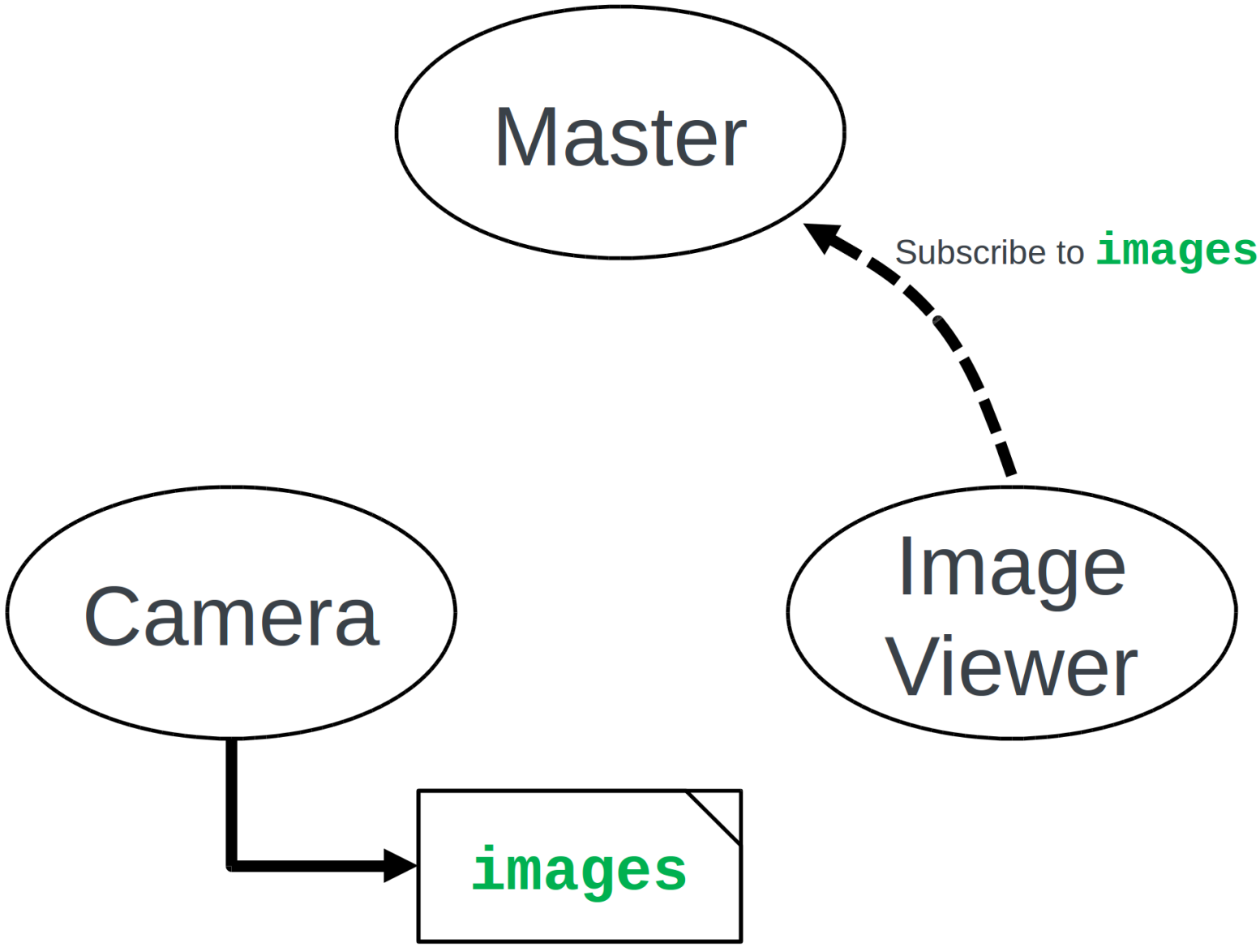
Recap

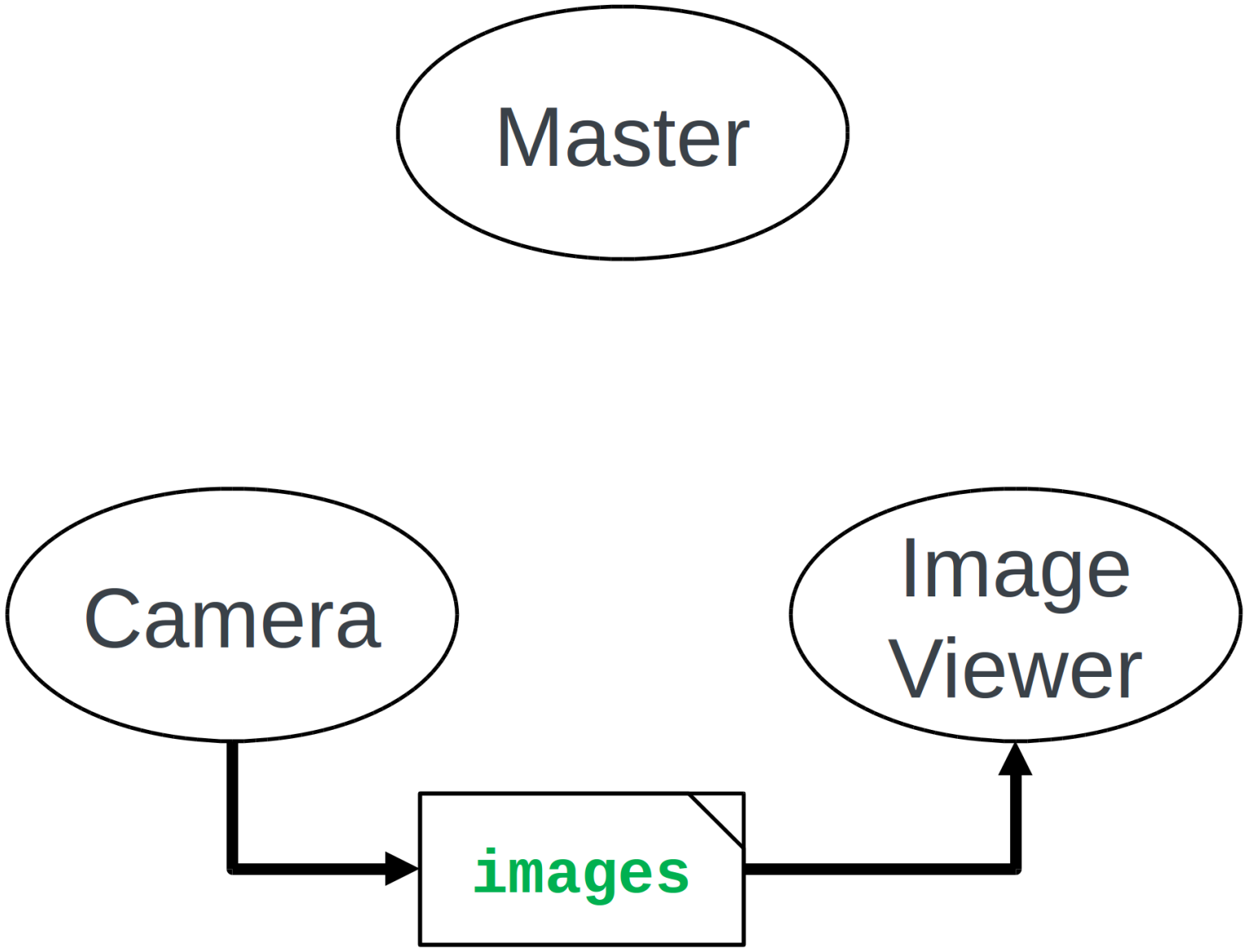
Summary of yesterday's session

- We will focus today on ROS topics and messages..


Master
Camera
Image
Viewer


Master
advertise images
Camera
Image
Viewer


Master
Subscribe to images
Camera
Image Viewer
images


Master
Camera
Image
Viewer
images

A simple ROS node

../scripts/00_simple_node.py

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


import rospy
from time import sleep

rospy.init_node("print_text")


while True:
    print "Hello world!"
    sleep(1)
```

A simple ROS node

../scripts/01_simple_node.py

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

import rospy


rospy.init_node("print_text")
rate = rospy.Rate(1)

while not rospy.is_shutdown():
    print "Hello world!"
    rate.sleep()
```

Writing a publisher node in Python

../scripts/02_simple_publisher.py

```
rospy.init_node('node name')
```

- nodes name must be unique. If you want to make sure the name of the node is unique:

```
rospy.init_node('node name', anonymous= True)
```

- node name will look like this:
  `/print_text_19637_1567065017476`

Three ways to run a node

ROS Nodes

There are 3 ways to run a node:

1. Like you normally do (not recommended). Example (in case of python node):

```
python <file name>
```

2. using rosrun command:

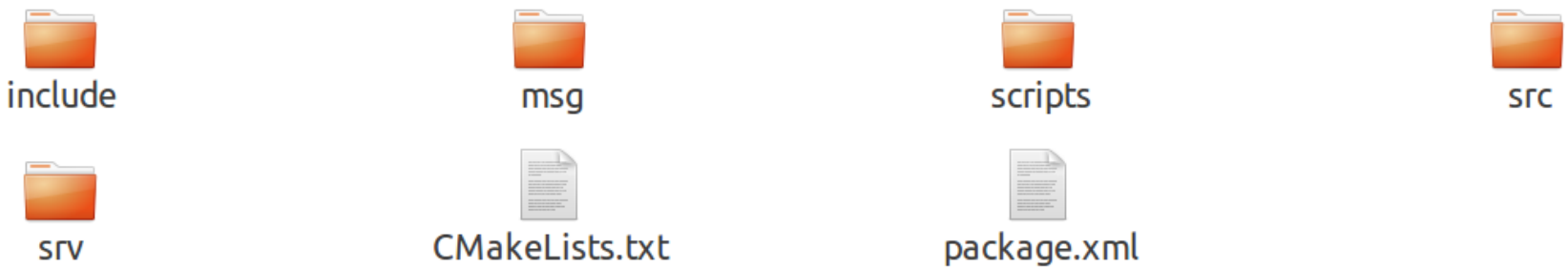
```
rosrun <package name> <node name>
```

3. Using launch files. (we'll see it later)

Let's create a package first!

ROS Nodes

- ROS commands find your files (python scripts, cpp files, launch files, message definitions) if they are located in a package inside the workspace.
- Normally, a package looks like this:



Let's create a package first!

ROS Nodes

- go to the README and do the steps for **creating a package**.

ROS commands

ROS Nodes

- Navigate to a ROS package directly:

```
roscd <package name>
```

- run a node without navigating to it's directory:

```
rosrun <package name> <executable>
```

Let's create a package first!

ROS Nodes

- go to the README and do the steps for **running a node**.

More ROS commands

ROS Nodes, Topics, and Messages

- List all the running nodes:

```
rosnode list
```

- Get more info. about a certain node:

```
rosnode info <node name>
```

Writing a publisher node in Python

ROS Nodes, Topics, and Messages

- Let's extend our previous node and make it publish a String ROS message.

Writing a publisher node in Python

../scripts/02_simple_publisher.py

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

import rospy

from std_msgs.msg import String

rospy.init_node('talker')

pub = rospy.Publisher('myFirstTopic', String, queue_size=10)

rate = rospy.Rate(1)

my_message = String()
my_message.data = "Hello there! How are you?"

while not rospy.is_shutdown():
    pub.publish(my_message)
    rate.sleep()
```

Writing a publisher node in Python

../scripts/02_simple_publisher.py

```
rospy.Publisher(name, data_class, queue_size)
```

- `name`: Name of the topic to publish on.
- `data_class`: The type of message. It is a ROS message class.
- `queue_size`: The size of the outgoing message queue.

Writing a publisher node in Python

../scripts/02_simple_publisher.py

```
rospy.Publisher(
name,
data_class,
subscriber_listener=None,
tcp_nodelay=False,
latch=False,
headers=None,
queue_size=None
)
```

Writing a publisher node in Python

Things to note..

- ROS messages are implemented as classes.
- To publish a message you also need to define a **Publisher** class.
- Most of ROS concepts and functionalities are implemented as classes. This is why understanding OOP helps you understand ROS better.

Writing a publisher node in Python

ROS Nodes, Topics, and Messages

- Go to the README file and do the instructions of section: **some of ROS commands**.

More ROS commands

ROS Nodes, Topics, and Messages

- Get the current list of topics:

```
rostopic list
```

- Print published messages:

```
rostopic echo <topic name>
```

More ROS commands

ROS Nodes, Topics, and Messages

- Publish a message from terminal:

```
rostopic pub <topic name> <msg type> <msg>
```

- Get message type of a topic:

```
rostopic type <topic name>
```

How to use ROS messages

ROS Nodes, Topics, and Messages

- ROS messages are just classes with attributes you can fill.
- ROS messages are defined in separate files and have to be placed in a package. (will be covered today).
- The following command can be used to see the class attributes, or the description, of a ROS message:

```
rosmsg show <package/msg>
```

How to use ROS messages

Example

```
rosmsg show std_msgs/String
```

The output is:

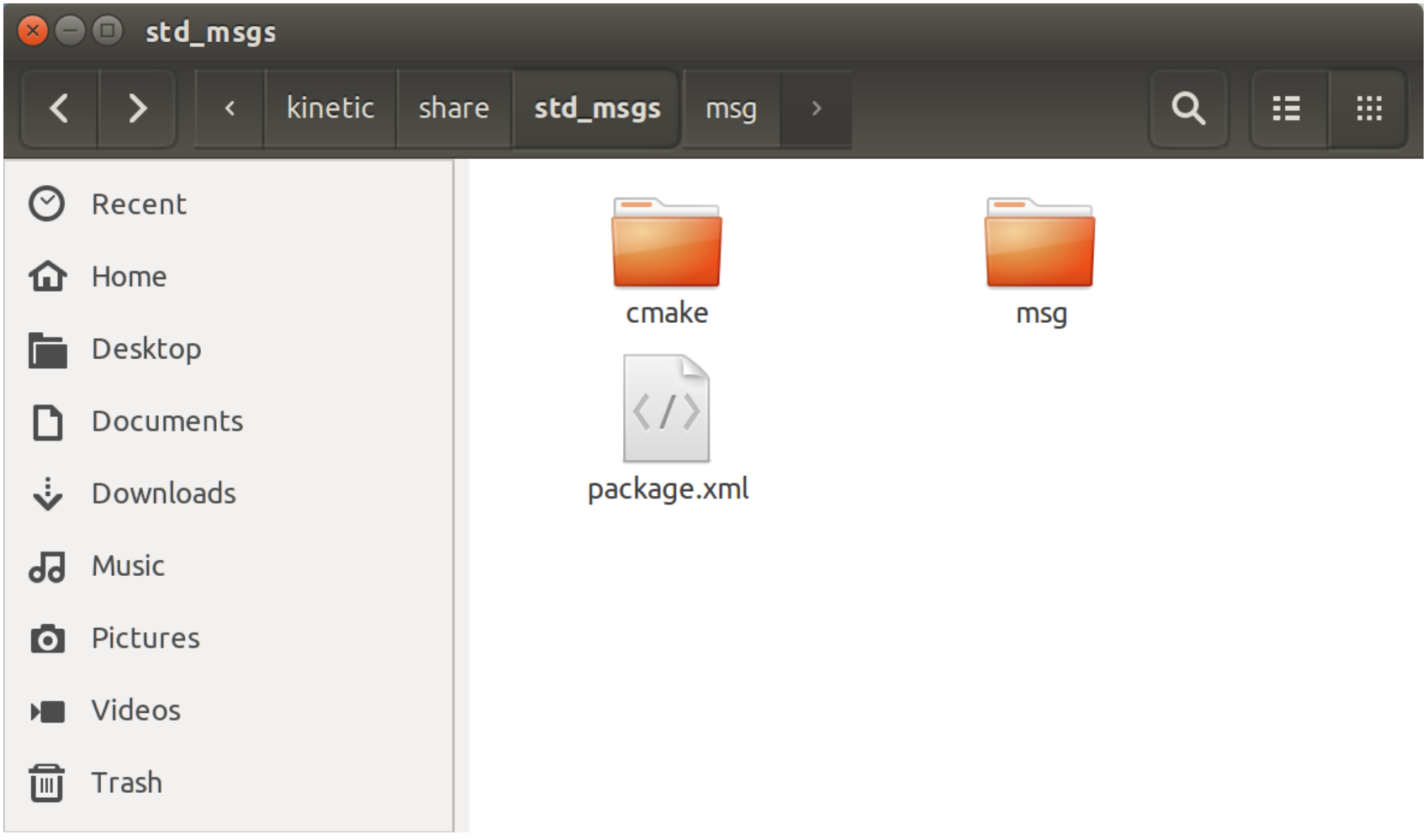
```
string data
```

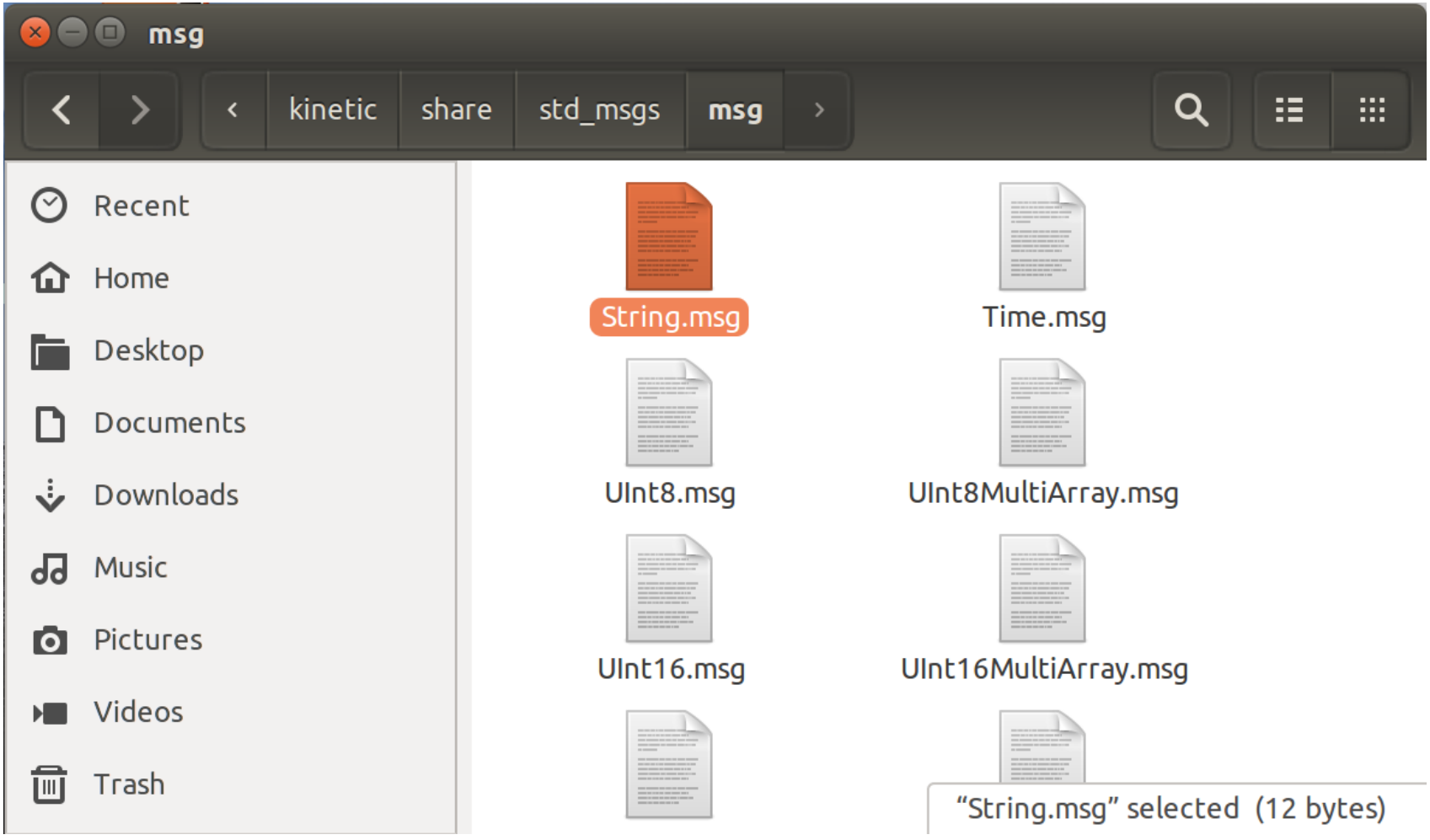
- It means ROS `String` message is a class with an attribute named `data` of type string (Python string).

How to use ROS messages

Importing a ROS message

- `String` message is located in the `std_msgs` package.


std_msgs
kinetic
share
std_msgs
msg
Recent
Home
Desktop
Documents
Downloads
Music
Pictures
Videos
Trash
cmake
msg
package.xml



Note: this is not the file that is imported in our script though! this file is not even Python nor C++. We will see later what this file is.

How to use ROS messages

Importing a ROS message

Python

```
from std_msgs.msg import String
```

C++

```
#include "std_msgs/String.h"
```

Exercise 1

Writing a subscriber node in Python

../scripts/03_simple_subscriber.py

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

import rospy
from std_msgs.msg import String

def my_callback_function(msg):
    print msg

rospy.init_node('listener')
rate = rospy.Rate(100)
sub = rospy.Subscriber('myFirstTopic', String,
                       callback=my_callback_function)

while not rospy.is_shutdown():
    pass
```

Writing a subscriber node in Python

../scripts/04_simple_subscriber.py

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

import rospy
from std_msgs.msg import String


def my_callback_function(msg):
    print msg

rospy.init_node('listener')
rate = rospy.Rate(100)

rospy.Subscriber('myFirstTopic', String, callback=my_callback_function)

rospy.spin()
```

Writing a subscriber node in Python

Subscriber class

- go to the README, and let's do section:
  Simple subscriber

Writing a subscriber node in Python

Subscriber class

```
rospy.Subscriber(
name,
data_class,
callback=None,
callback_args=None,
queue_size=None,
buff_size=65536,
tcp_nodelay=False
)
```

Writing a subscriber node in Python

Subscriber class

- go to the README, and let's do section:
  Simple subscriber

General notes

more on ROS nodes

- You can define multiple publishers and subscribers in a node.
- You can call `init_node` function once only!
- To make your code look neat, you can wrap ROS stuff in a class and hide them!

Let's see an example with multiple publishers/subscribers and let's define them in a class...

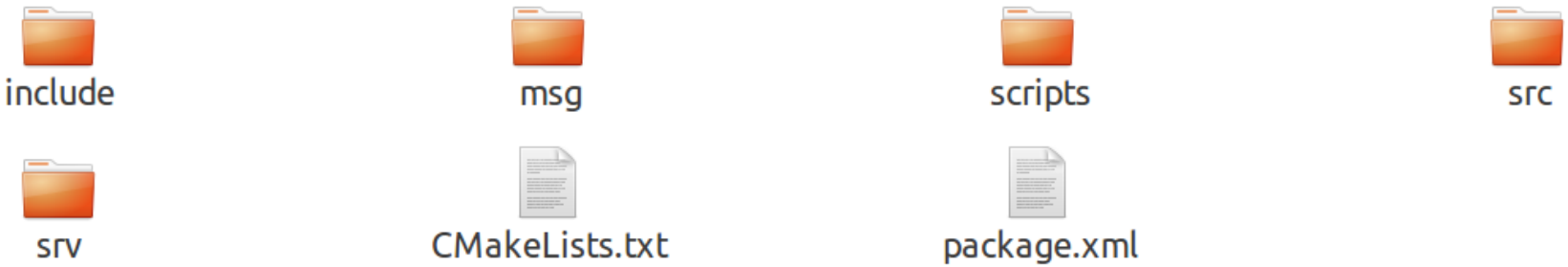
Let’s see script
05_node_example.py

Catkin

- In ROS, packages are combiled and built using Catkin.

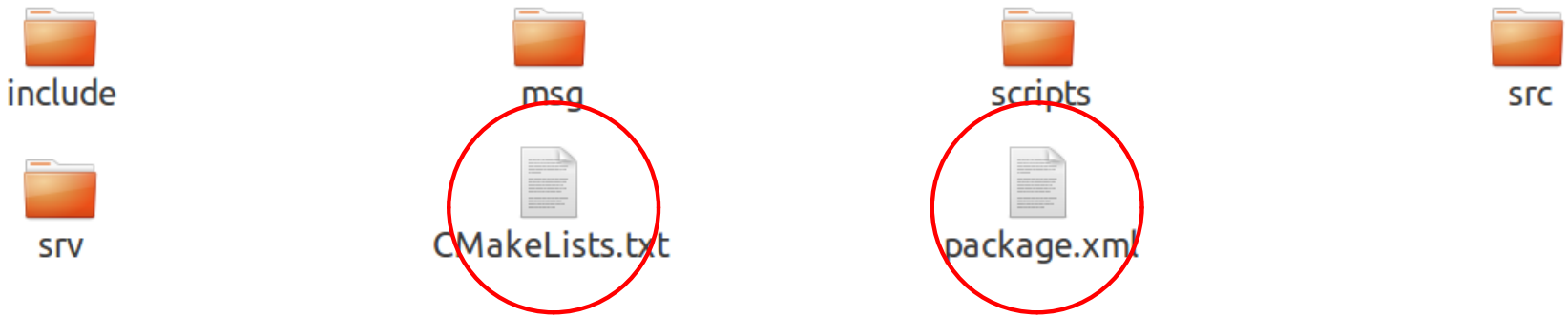
A ROS package

- A ROS package might look like this:



A ROS package

- A ROS package might look like this:



rospy reference

- rospy full documentation (all the classes, all the functions ..etc):

  `http://docs.ros.org/kinetic/api/rospy/html/`