

# AI & Robotics

ROS: Publisher & Subscriber

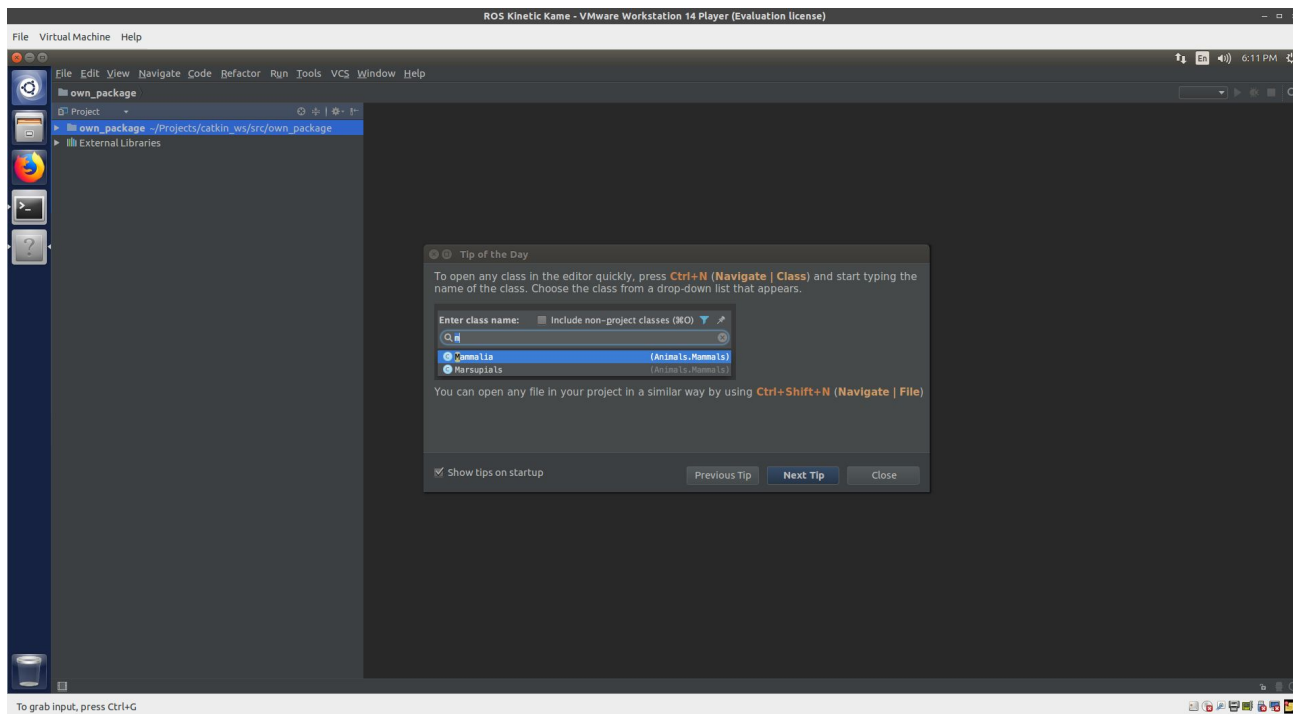
# Goals



## The **junior-colleague**

- can create and run their own ROS publisher using Python
- can create and run their own ROS subscriber using Python

# Open own\_package



[INFO]

Start “charm own\_package”, add a directory named “scripts” and create publish.py.

# Publisher Node: publisher.py

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

if __name__ == '__main__':
    publisherName = "test_publisher_" + str(os.getpid())
    topicName     = "test_topic"

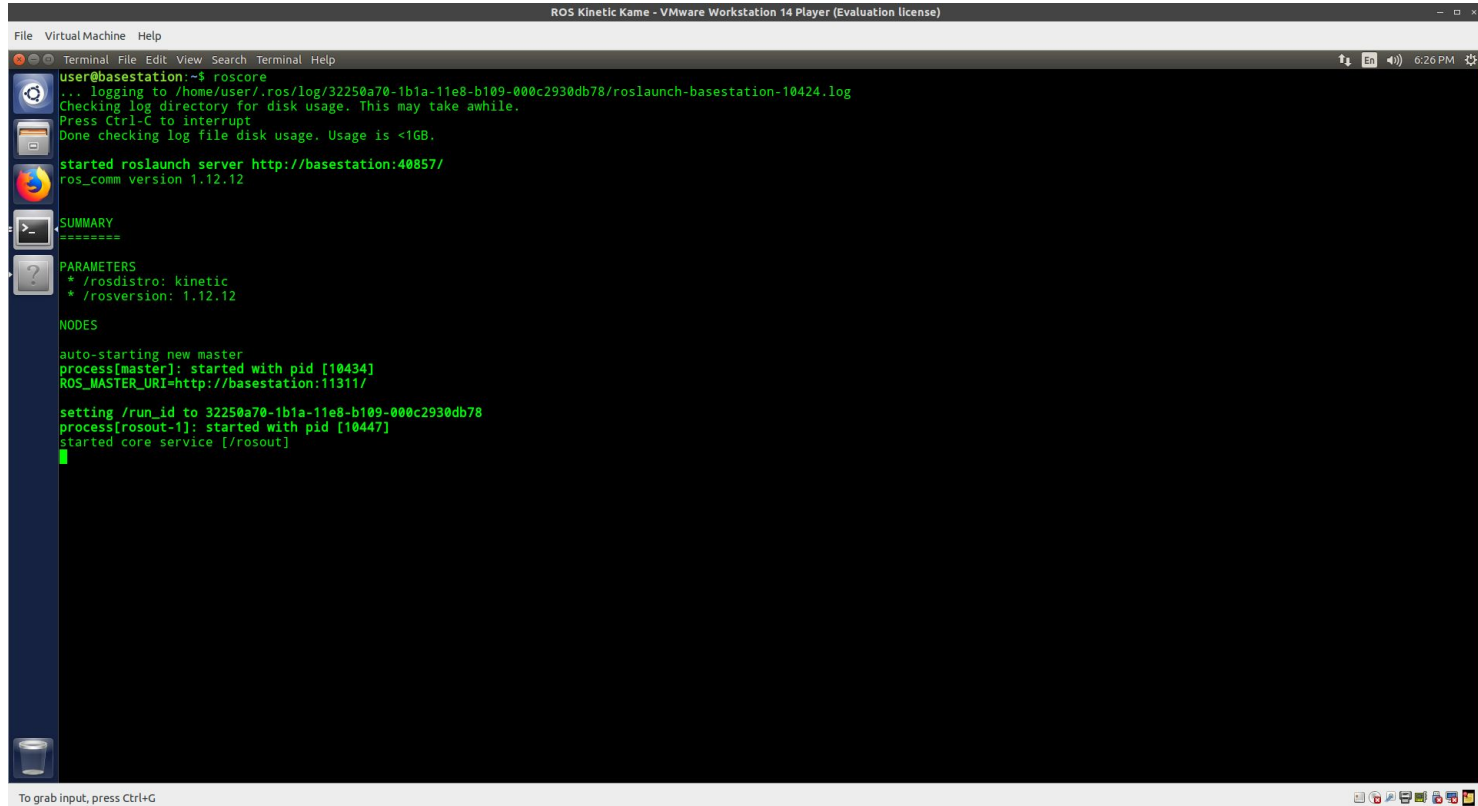
    try:
        rospy.init_node(publisherName, anonymous=False)
        publisher = rospy.Publisher(topicName, String, queue_size=1)

        counter = 0
        while True:
            rawInput = raw_input("Ready to send a data message . . . <ENTER>\n")

            counter += 1
            data = publisherName + ":data:" + str(counter)
            rospy.loginfo("Sending: " + data)
            publisher.publish(data)
            rospy.loginfo("Sent\n")

    except rospy.ROSInterruptException:
        print "An interrupt occurred."
```

# Run roscore



```
ROS Kinetic Kame - VMware Workstation 14 Player (Evaluation license)
File VirtualMachine Help
Terminal File Edit View Search Terminal Help
user@basestation:~$ roscore
... logging to /home/user/.ros/log/32250a70-1b1a-11e8-b109-000c2930db78/roslaunch-basestation-10424.log
Checking log directory for disk usage. This may take awhile.
Press Ctrl-C to interrupt
Done checking log file disk usage. Usage is <1GB.

started roslaunch server http://basestation:40857/
ros_comm version 1.12.12

SUMMARY
=====
PARAMETERS
* /rostdistro: kinetic
* /rosversion: 1.12.12

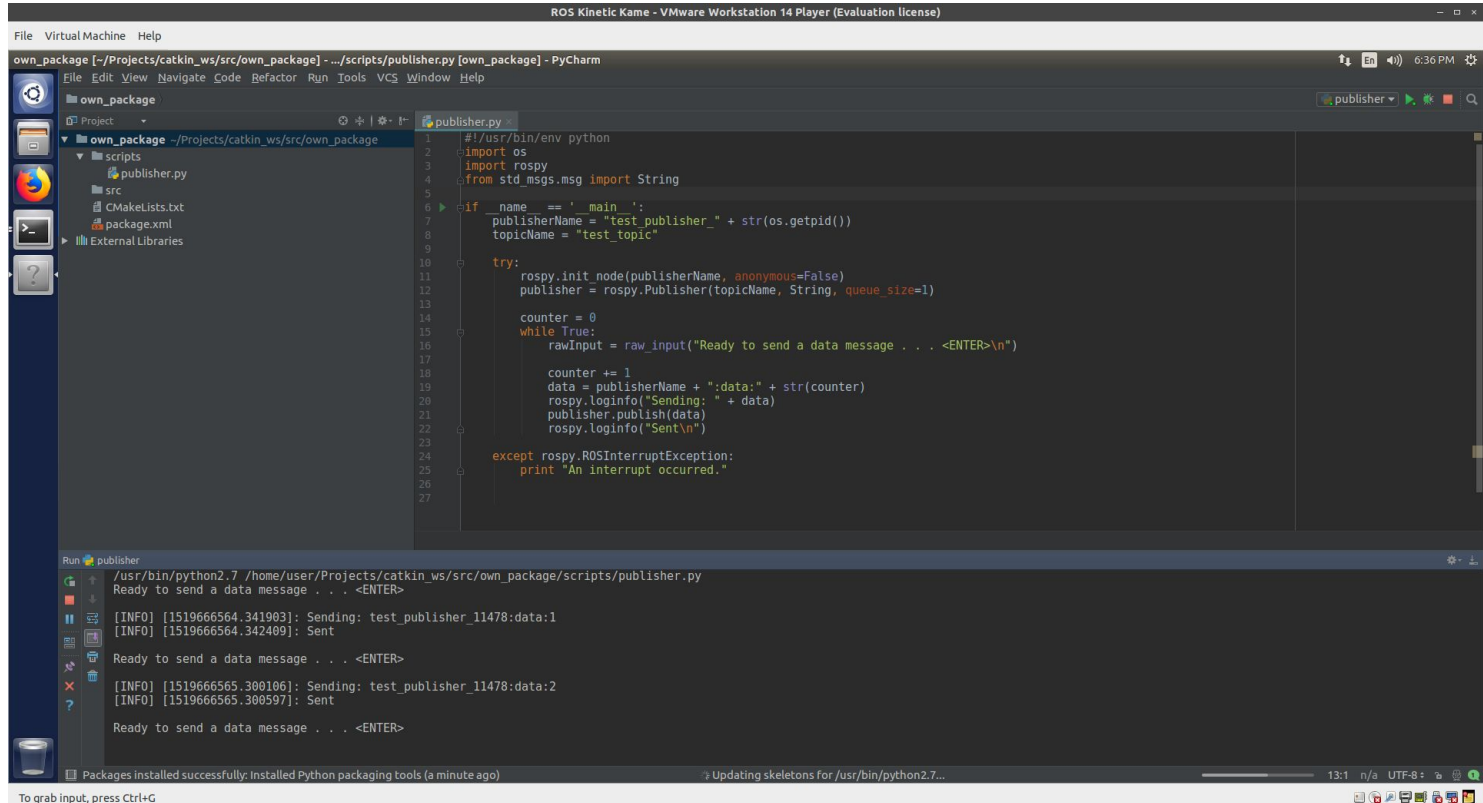
NODES

auto-starting new master
process[master]: started with pid [10434]
ROS_MASTER_URI=http://basestation:11311/

setting /run_id to 32250a70-1b1a-11e8-b109-000c2930db78
process[rosout-1]: started with pid [10447]
started core service [/rosout]
```

To grab input, press Ctrl+G

# Run publisher.py



The screenshot shows a PyCharm IDE window titled "ROS Kinetic Kame - VMware Workstation 14 Player (Evaluation license)". The main editor displays the file `publisher.py` within the `own_package` project. The code is a ROS publisher script that initializes a node, subscribes to a topic, and publishes data messages. The terminal at the bottom shows the execution of the script, with output indicating that the node is running and publishing data messages.

```
1 #!/usr/bin/env python
2 import os
3 import rospy
4 from std_msgs.msg import String
5
6 if __name__ == '__main__':
7     publisherName = "test_publisher_" + str(os.getpid())
8     topicName = "test_topic"
9
10    try:
11        rospy.init_node(publisherName, anonymous=False)
12        publisher = rospy.Publisher(topicName, String, queue_size=1)
13
14        counter = 0
15        while True:
16            rawInput = raw_input("Ready to send a data message . . . <ENTER>\n")
17
18            counter += 1
19            data = publisherName + ":data:" + str(counter)
20            rospy.loginfo("Sending: " + data)
21            publisher.publish(data)
22            rospy.loginfo("Sent\n")
23
24    except rospy.ROSInterruptException:
25        print "An interrupt occurred."
26
27
```

Run publisher

```
/usr/bin/python2.7 /home/user/Projects/catkin_ws/src/own_package/scripts/publisher.py
Ready to send a data message . . . <ENTER>
[INFO] [1519666564.341903]: Sending: test_publisher_11478:data:1
[INFO] [1519666564.342409]: Sent
Ready to send a data message . . . <ENTER>
[INFO] [1519666565.300106]: Sending: test_publisher_11478:data:2
[INFO] [1519666565.300597]: Sent
Ready to send a data message . . . <ENTER>
```

⊞ Packages installed successfully: Installed Python packaging tools (a minute ago) ⊞ Updating skeletons for /usr/bin/python2.7...

13:1 n/a UTF-8

To grab input, press Ctrl+G

# Subscriber Node

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

import os
import rospy
from std_msgs.msg import String

def callbackFunction(data):
    rospy.loginfo(data.data)

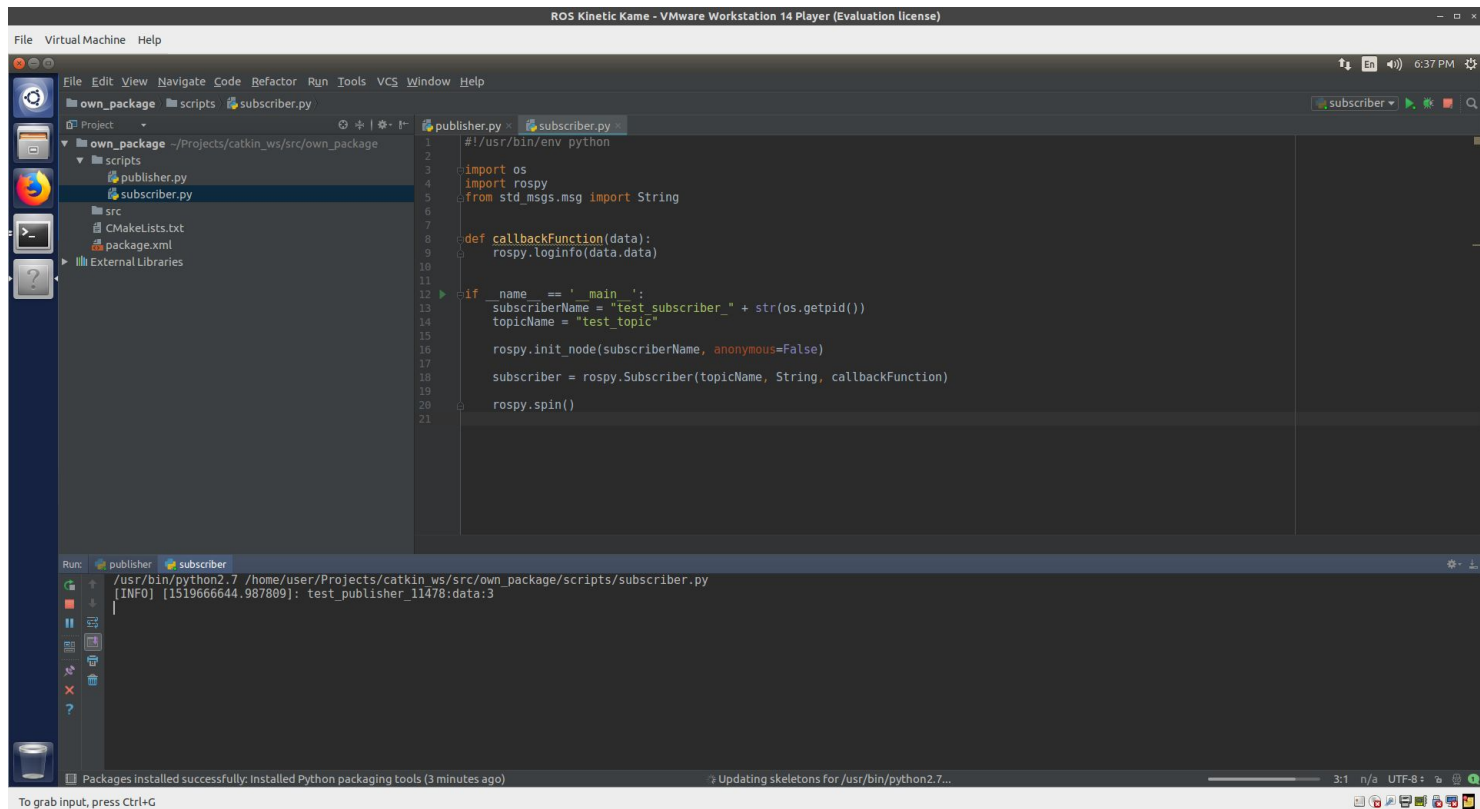
if __name__ == '__main__':
    subscriberName = "test_subscriber_" + str(os.getpid())
    topicName      = "test_topic"

    rospy.init_node(subscriberName, anonymous=False)

    subscriber = rospy.Subscriber(topicName, String, callbackFunction)

    rospy.spin()
```

# Run subscriber.py





# Building & Running

Terminal 1

```
$ roscore # You already had this running, no?
```

Terminal 2

```
$ chmod +x ~/Projects/catkin_ws/src/own_package/scripts/publisher.py
$ rosrun own_package publisher.py

. . .

Ready to send a data message . . . <ENTER>

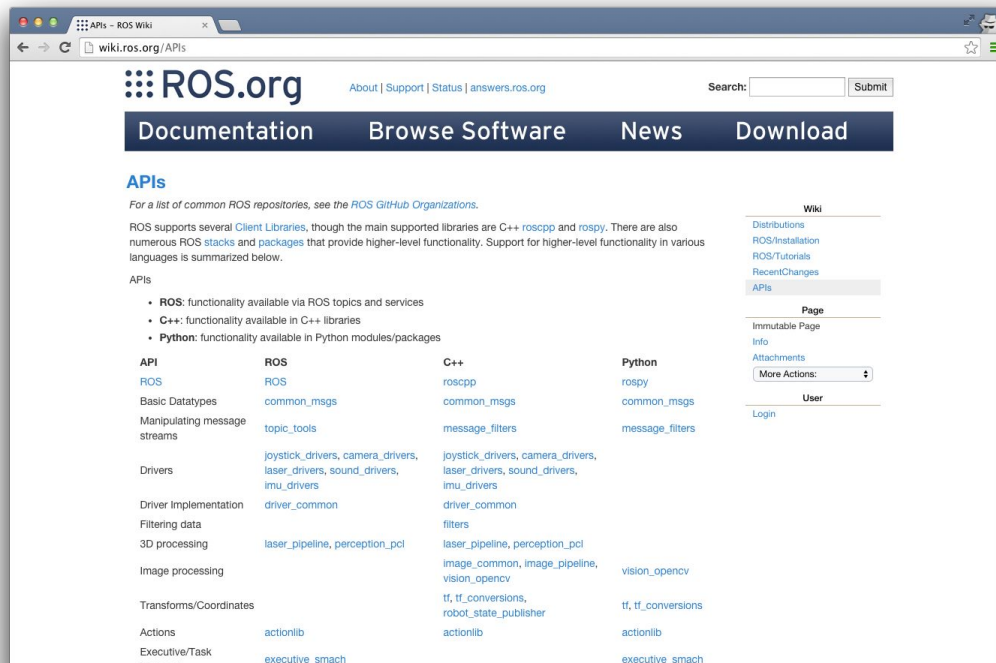
[INFO] [WallTime: 1426278799.760896] Sending: test_publisher_14645:data:2
[INFO] [WallTime: 1426278799.761684] Sent

Ready to send a data message . . . <ENTER>
```

Terminal 3

```
$ chmod +x ~/Projects/catkin_ws/src/own_package/scripts/subscriber.py
$ rosrun own_package subscriber.py # After Sending: . . . data:1
[INFO] [WallTime: 1426278799.792494] test_publisher_14645:data:2
```

# Application Programming Interface



The screenshot shows the ROS.org website with the 'APIs' section selected in the navigation bar. The page title is 'APIs' and it provides a list of common ROS repositories. The main content area is divided into four columns: ROS, C++, Python, and a Wiki section. The ROS column lists various packages like 'common\_msgs', 'topic\_tools', 'joystick\_drivers', etc. The C++ column lists 'roscpp', 'common\_msgs', 'message\_filters', etc. The Python column lists 'rospy', 'common\_msgs', 'message\_filters', etc. The Wiki section on the right has links to 'Distributions', 'ROS/Installation', 'ROS/Tutorials', 'RecentChanges', and 'APIs'.

**ROS.org** About | Support | Status | answers.ros.org Search:  Submit

**Documentation** Browse Software News Download

## APIs

For a list of common ROS repositories, see the [ROS GitHub Organizations](#).

ROS supports several [Client Libraries](#), though the main supported libraries are C++ [roscpp](#) and [rospy](#). There are also numerous ROS [stacks](#) and [packages](#) that provide higher-level functionality. Support for higher-level functionality in various languages is summarized below.

APIs

- **ROS**: functionality available via ROS topics and services
- **C++**: functionality available in C++ libraries
- **Python**: functionality available in Python modules/packages

API	ROS	C++	Python
ROS	ROS	roscpp	rospy
Basic Datatypes	common_msgs	common_msgs	common_msgs
Manipulating message streams	topic_tools	message_filters	message_filters
Drivers	joystick_drivers, camera_drivers, laser_drivers, sound_drivers, imu_drivers	joystick_drivers, camera_drivers, laser_drivers, sound_drivers, imu_drivers	
Driver Implementation	driver_common	driver_common	
Filtering data		filters	
3D processing	laser_pipeline, perception_pcl	laser_pipeline, perception_pcl	
Image processing		image_common, image_pipeline, vision_opencv	vision_opencv
Transforms/Coordinates		tf, tf_conversions, robot_state_publisher	tf, tf_conversions
Actions	actionlib	actionlib	actionlib
Executive/Task Manager	executive_smach		executive_smach

**Wiki**

- [Distributions](#)
- [ROS/Installation](#)
- [ROS/Tutorials](#)
- [RecentChanges](#)
- [APIs](#)

**Page**

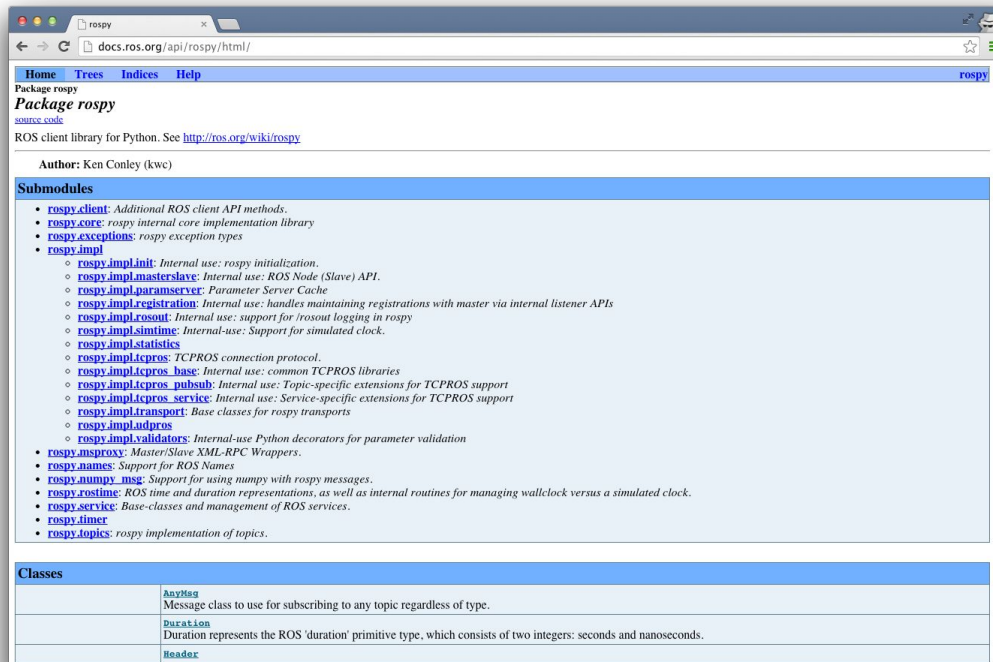
- [Immutable Page](#)
- [Info](#)
- [Attachments](#)
- More Actions:

**User**

- [Login](#)

<http://wiki.ros.org/APIs>

# Application Programming Interface



The screenshot shows a web browser window displaying the ROS API documentation for the `rospy` package. The browser's address bar shows `docs.ros.org/api/rospy/html/`. The page has a navigation bar with links for `Home`, `Trees`, `Indices`, and `Help`. The main content area is titled `Package rospy` and includes a `source code` link. Below this, it states that `rospy` is the ROS client library for Python, with a link to the ROS wiki. The `Author` is listed as Ken Conley (kwc). The `Submodules` section lists various submodules with brief descriptions, including `rospy_client`, `rospy_core`, `rospy_exceptions`, and `rospy_impl` (which contains many sub-submodules like `rospy_impl_init`, `rospy_impl_masterslave`, etc.). The `Classes` section lists `AnyMsg`, `Duration`, and `Reader` with their respective descriptions.

Package rospy  
**Package rospy**  
[source code](#)

ROS client library for Python. See <http://ros.org/wiki/rospy>.

Author: Ken Conley (kwc)

**Submodules**

- [rospy\\_client](#): Additional ROS client API methods.
- [rospy\\_core](#): rospy internal core implementation library
- [rospy\\_exceptions](#): rospy exception types
- [rospy\\_impl](#)
  - [rospy\\_impl\\_init](#): Internal use: rospy initialization.
  - [rospy\\_impl\\_masterslave](#): Internal use: ROS Node (Slave) API.
  - [rospy\\_impl\\_paramserver](#): Parameter Server Cache
  - [rospy\\_impl\\_registration](#): Internal use: handles maintaining registrations with master via internal listener APIs
  - [rospy\\_impl\\_rosout](#): Internal use: support for /rosout logging in rospy
  - [rospy\\_impl\\_simtime](#): Internal-use: Support for simulated clock.
  - [rospy\\_impl\\_statistics](#)
  - [rospy\\_impl\\_tcpros](#): TCPROS connection protocol.
  - [rospy\\_impl\\_tcpros\\_base](#): Internal use: common TCPROS libraries
  - [rospy\\_impl\\_tcpros\\_pubsub](#): Internal use: Topic-specific extensions for TCPROS support
  - [rospy\\_impl\\_tcpros\\_service](#): Internal use: Service-specific extensions for TCPROS support
  - [rospy\\_impl\\_transport](#): Base classes for rospy transports
  - [rospy\\_impl\\_adapters](#)
  - [rospy\\_impl\\_validators](#): Internal-use Python decorators for parameter validation
- [rospy\\_msgsproxy](#): Master/Slave XML-RPC Wrappers.
- [rospy\\_names](#): Support for ROS Names
- [rospy\\_numpy\\_msg](#): Support for using numpy with rospy messages.
- [rospy\\_rostime](#): ROS time and duration representations, as well as internal routines for managing wallclock versus a simulated clock.
- [rospy\\_service](#): Base-classes and management of ROS services.
- [rospy\\_linter](#)
- [rospy\\_topics](#): rospy implementation of topics.

**Classes**

<a href="#">AnyMsg</a>	Message class to use for subscribing to any topic regardless of type.
<a href="#">Duration</a>	Duration represents the ROS 'duration' primitive type, which consists of two integers: seconds and nanoseconds.
<a href="#">Reader</a>	

<http://docs.ros.org/api/rospy/html>