ROS Python Node Development Guide

Welcome to guide on developing Python nodes in ROS (Robot Operating System). This README will introduce you to the basics of writing publisher and subscriber nodes using Python. Understanding these concepts is crucial for developing applications in ROS.

Prerequisites

Ensure you have ROS installed on your system, along with Python 3.8, as we will be using this version for our examples. The choice of Python version is important to avoid potential errors in systems with multiple Python versions installed.

Example 1: Basic Publisher Node

This example demonstrates how to create a simple publisher node that sends string messages.

```
#!/usr/bin/env python3.8

import rospy
from std_msgs.msg import String

if __name__ == '__main__':
    rospy.init_node('publisher_node')
    rospy.loginfo('Hello world')

    rospy.sleep(1.0)
    rate = rospy.Rate(1) # 1 Hz
    test_pub = rospy.Publisher("/data", String, queue_size=10)

    while not rospy.is_shutdown():
        test_pub.publish("Hello world")
        rate.sleep()
    else:
        rospy.loginfo("Node SHUTDOWNED")
```

Key Concepts:

- `rospy.Publisher()`: Initializes a publisher node. The `queue_size` parameter is set to 10, meaning if an 11th message arrives before the first one is sent, the oldest message is dropped.
- `rospy.Rate(1)`: Controls the rate at which messages are published, in this case, 1Hz (one message per second).
- `rospy.is_shutdown()`: Keeps the node running until ROS is shut down.

Example 2: Logging Node

This example introduces logging within a ROS node, showcasing how to log info, warnings, and errors.

```
#!/usr/bin/env python3.8

import rospy

if __name__ == '__main__':
    rospy.init_node('logger_node')
    rospy.loginfo('Hello world')
    rospy.logwarn('This is warning')
    rospy.logerr('This is error')

    rospy.sleep(1.0)
    rate = rospy.Rate(1) # 1 Hz

    while not rospy.is_shutdown():
        rospy.loginfo('In the loop')
        rate.sleep()
```

Key Concepts:

- `rospy.loginfo()`, `rospy.logwarn()`, `rospy.logerr()`: Used for logging different levels of messages to the ROS console.
- The loop structure is similar to Example 1 but focuses on logging instead of publishing messages.

Example 3: Simple Subscriber Node

This final example covers creating a subscriber node that listens to string messages and executes a callback function upon receiving a message.

```
#!/usr/bin/env python3.8

import rospy
from std_msgs.msg import String

def callback(data):
    rospy.loginfo("Received data: %s", data.data)

if __name__ == '__main__':
    rospy.init_node('subscriber_node')
    rospy.Subscriber("/data", String, callback)
    rospy.loginfo('Hello from sub')

    rospy.spin() # Keeps the node alive to listen for incoming data
```

Key Concepts:

- `rospy.Subscriber()`: Initializes a subscriber node that listens to messages on a specified topic and calls a function (`callback`) when a message is received.
- `rospy.spin()`: Unlike the previous examples with explicit loops, `spin()` keeps your node running and waiting for callbacks to execute.

Running Your Nodes

After writing your nodes, remember to make them executable:

```
chmod +x your_node_name.py
```

To run a node, use the `rosrun` command:

```
rosrun your_package_name your_node_name.py
```

Conclusion

This guide introduced you to basic concepts of writing, running, and debugging Python nodes in ROS. By following these examples, you can start building more complex ROS applications tailored to your needs. Remember, the key to mastering ROS is practice and experimentation. Happy coding!

✒️ Crafted by

Sadegh-Kalami

🔗 **Connect with me:**

- **GitHub Profile**: collaborations at Sadegh-Kalami.
- **Personal Site**: My projects and portfolio at Sadegh-Kalami's GitHub Page.

Your journey through my digital craftsmanship begins here.
