Example #2: Simple Action Client

Aim

- To write a ROS Node which will act as Simple Action Client.
- It should send **angle** by which the turtle should rotate and **distance** by which it should move as **Goal** to the Simple Action Server discussed in previous example.
- The client should send goals at an interval of 5 seconds such that the turtle in the `turtlesim_node` traces a square shape of edge 2 units in length.
- The name of the action use by this Simple Action Server should be `/action_client`.

NOTE: Same action file discussed at **Create a action message file** section can be used here also.

Code

node_simple_action_client_turtle.py

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

# ROS Node - Simple Action Client - Turtle

import rospy
import actionlib
import time

from pkg_ros_actions.msg import myActionMsgAction       # Message Class that is used by F
from pkg_ros_actions.msg import myActionMsgGoal         # Message Class that is used for


class SimpleActionClientTurtle:

    # Constructor
    def __init__(self):
        self._ac = actionlib.SimpleActionClient('/action_turtle',
                                                myActionMsgAction)
        self._ac.wait_for_server()
        rospy.loginfo("Action server is up, we can send new goals!")

    # Function to send Goals to Action Servers
    def send_goal(self, arg_dis, arg_angle):

        # Create Goal message for Simple Action Server
        goal = myActionMsgGoal(distance=arg_dis, angle=arg_angle)

        '''
            * done_cb is set to the function pointer of the function which should be call
              the Goal is processed by the Simple Action Server.

            * feedback_cb is set to the function pointer of the function which should be
              the goal is being processed by the Simple Action Server.
        '''
        self._ac.send_goal(goal, done_cb=self.done_callback,
                           feedback_cb=self.feedback_callback)

        rospy.loginfo("Goal has been sent.")
    # Function print result on Goal completion
    def done_callback(self, status, result):
        rospy.loginfo("Status is : " + str(status))
        rospy.loginfo("Result is : " + str(result))

    # Function to print feedback while Goal is being processed
    def feedback_callback(self, feedback):
        rospy.loginfo(feedback)


# Main Function
def main():
    # 1. Initialize ROS Node
    rospy.init_node('node_simple_action_client_turtle')

    # 2. Create a object for Simple Action Client.
    obj_client = SimpleActionClientTurtle()

    # 3. Send Goals to Draw a Square
    obj_client.send_goal(2, 0)
    rospy.sleep(5)

    obj_client.send_goal(2, 90)
    rospy.sleep(5)

    obj_client.send_goal(2, 90)
    rospy.sleep(5)

    obj_client.send_goal(2, 90)
    rospy.sleep(5)

    obj_client.send_goal(2, 90)

    # 4. Loop forever
    rospy.spin()


if __name__ == '__main__':
    main()
```

Download

Run Command

Now this server do the following,

```
roscd pkg_ros_actions

cd srcipts

sudo chmod +x node_simple_action_client_turtle.py

rosrun pkg_ros_actions node_simple_action_server_turtle.py

rosrun pkg_ros_actions node_simple_action_client_turtle.py
```

NOTE: For the Action Client to work properly Action Server needs to be running. Hence we need to start it before starting an Action Client.

NOTE: `roscore` should be running if you want to run a ROS Node.