Slide

https://kasets.art/nOW45Q

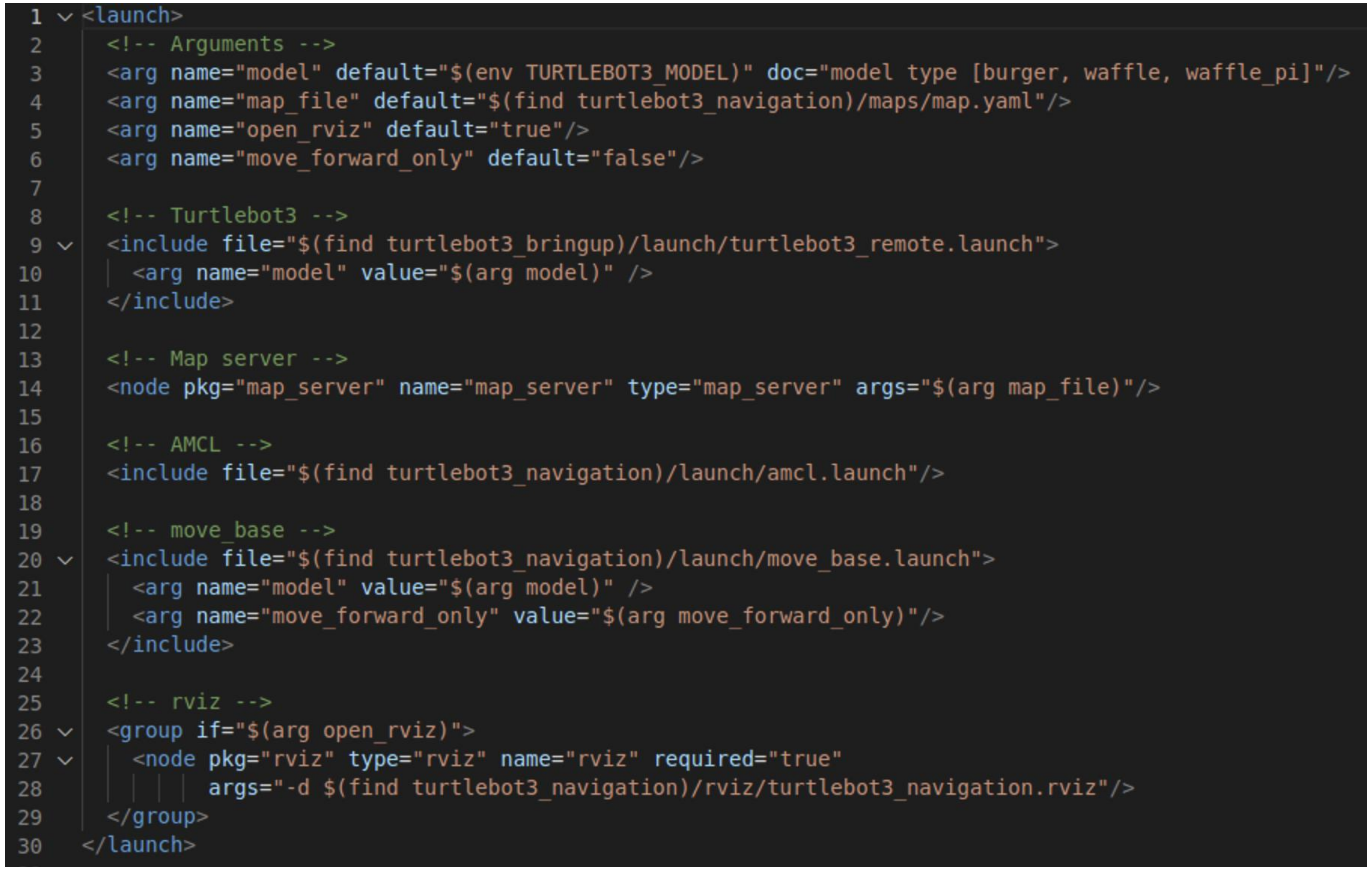
Navigation workshop

Noppanut Thongton

Navigation with Turtlebot3

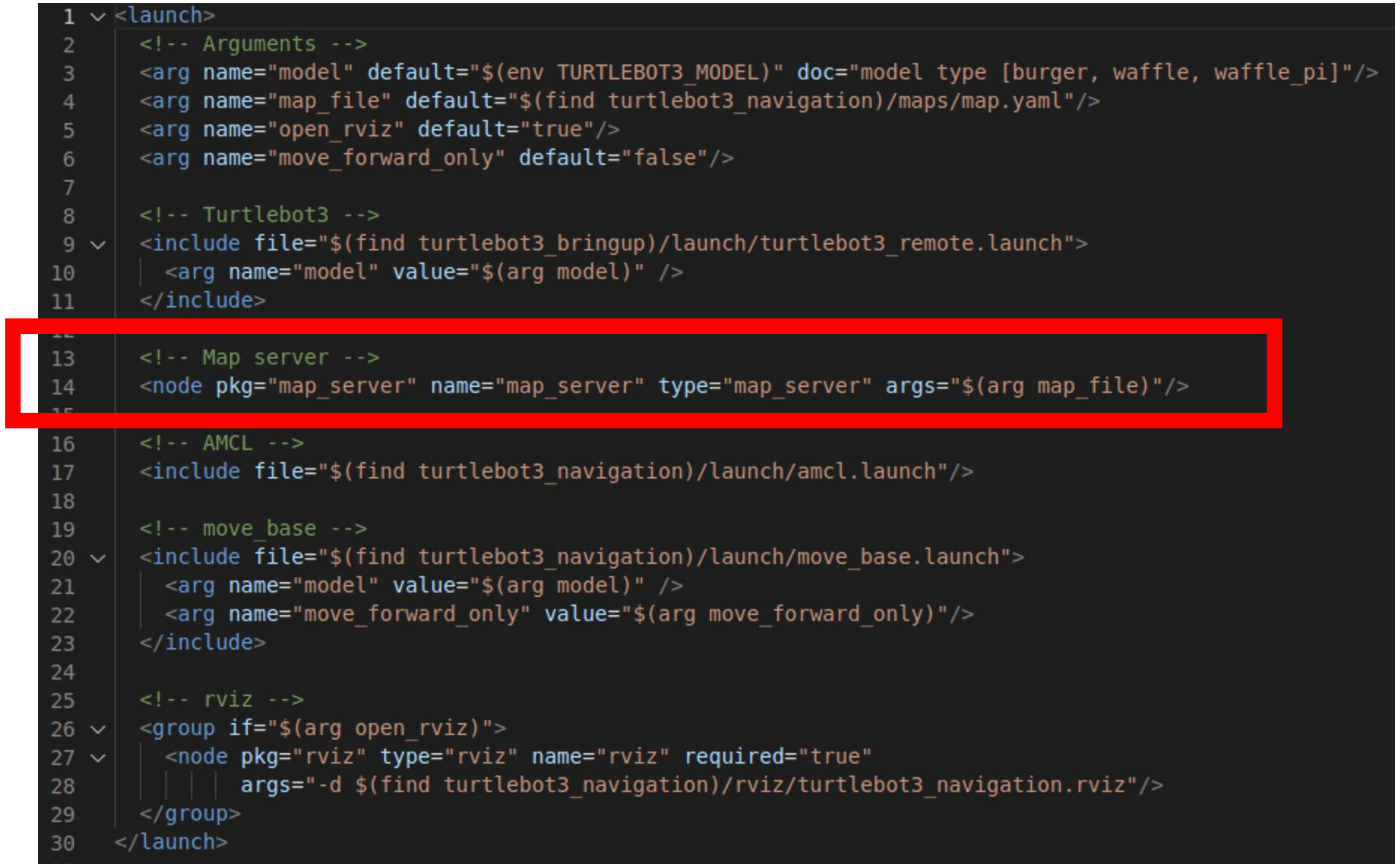
Using simulation of Turtlebot3 in Gazebo with navigation.

Turtlebot3 - Navigation



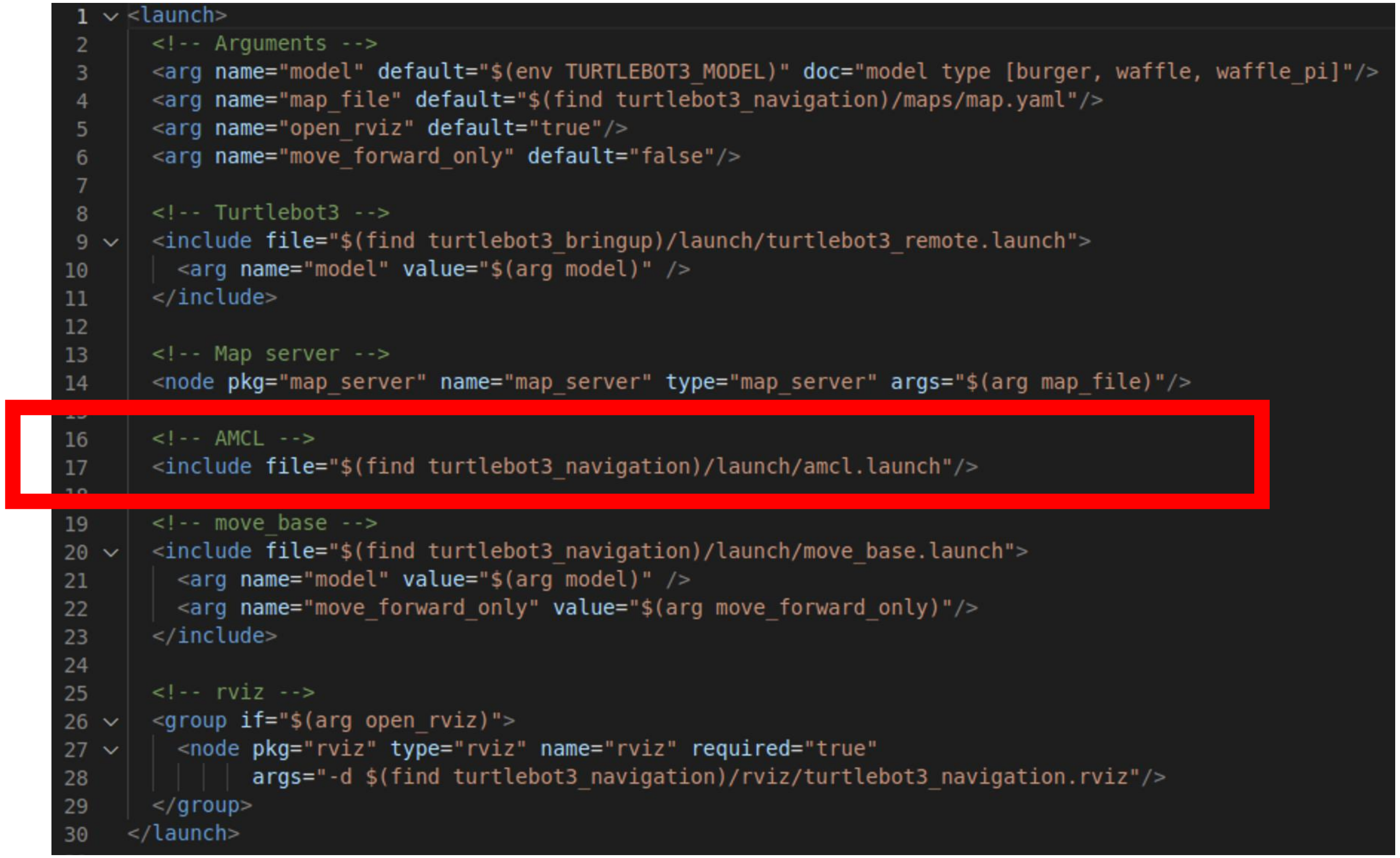
```
<launch>
  <!-- Arguments -->
  <arg name="model" default="$(env TURTLEBOT3_MODEL)" doc="model type [burger, waffle, waffle_pi]"/>
  <arg name="map_file" default="$(find turtlebot3_navigation)/maps/map.yaml"/>
  <arg name="open_rviz" default="true"/>
  <arg name="move_forward_only" default="false"/>

  <!-- Turtlebot3 -->
  <include file="$(find turtlebot3_bringup)/launch/turtlebot3_remote.launch">
    <arg name="model" value="$(arg model)" />
  </include>

  <!-- Map server -->
  <node pkg="map_server" name="map_server" type="map_server" args="$(arg map_file)"/>

  <!-- AMCL -->
  <include file="$(find turtlebot3_navigation)/launch/amcl.launch"/>

  <!-- move_base -->
  <include file="$(find turtlebot3_navigation)/launch/move_base.launch">
    <arg name="model" value="$(arg model)" />
    <arg name="move_forward_only" value="$(arg move_forward_only)"/>
  </include>

  <!-- rviz -->
  <group if="$(arg open_rviz)">
    <node pkg="rviz" type="rviz" name="rviz" required="true"
          args="-d $(find turtlebot3_navigation)/rviz/turtlebot3_navigation.rviz"/>
  </group>
</launch>
```

Turtlebot3 - Navigation



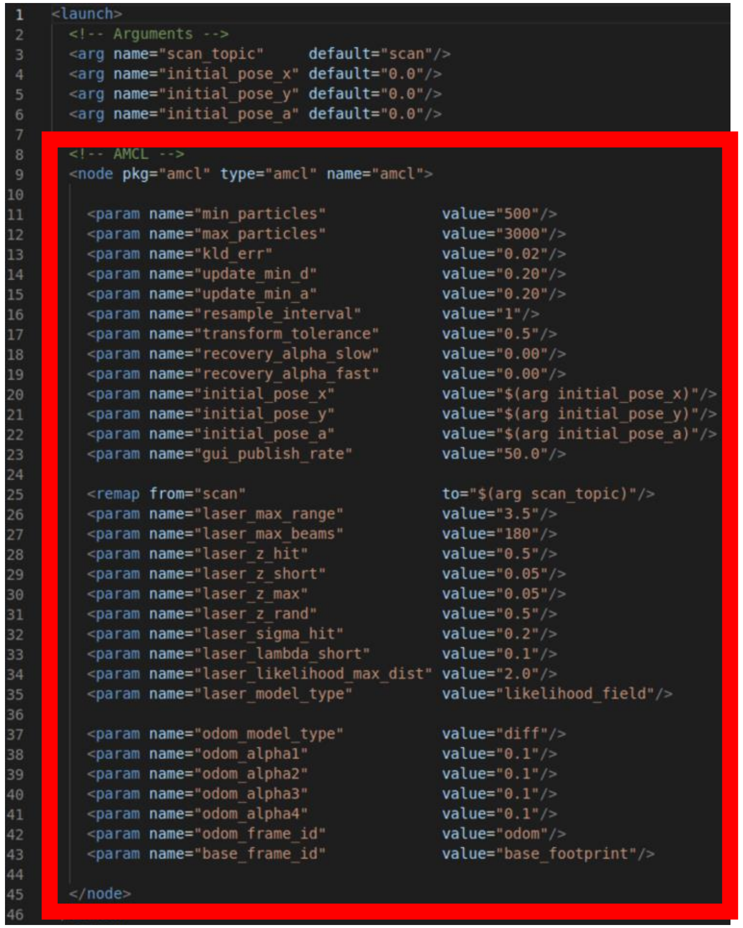
```xml
<launch>
  <!-- Arguments -->
  <arg name="model" default="$(env TURTLEBOT3_MODEL)" doc="model type [burger, waffle, waffle_pi]"/>
  <arg name="map_file" default="$(find turtlebot3_navigation)/maps/map.yaml"/>
  <arg name="open_rviz" default="true"/>
  <arg name="move_forward_only" default="false"/>

  <!-- Turtlebot3 -->
  <include file="$(find turtlebot3_bringup)/launch/turtlebot3_remote.launch">
    <arg name="model" value="$(arg model)" />
  </include>

  <!-- Map server -->
  <node pkg="map_server" name="map_server" type="map_server" args="$(arg map_file)"/>

  <!-- AMCL -->
  <include file="$(find turtlebot3_navigation)/launch/amcl.launch"/>

  <!-- move_base -->
  <include file="$(find turtlebot3_navigation)/launch/move_base.launch">
    <arg name="model" value="$(arg model)" />
    <arg name="move_forward_only" value="$(arg move_forward_only)"/>
  </include>

  <!-- rviz -->
  <group if="$(arg open_rviz)">
    <node pkg="rviz" type="rviz" name="rviz" required="true"
          args="-d $(find turtlebot3_navigation)/rviz/turtlebot3_navigation.rviz"/>
  </group>
</launch>
```

Turtlebot3 - Navigation



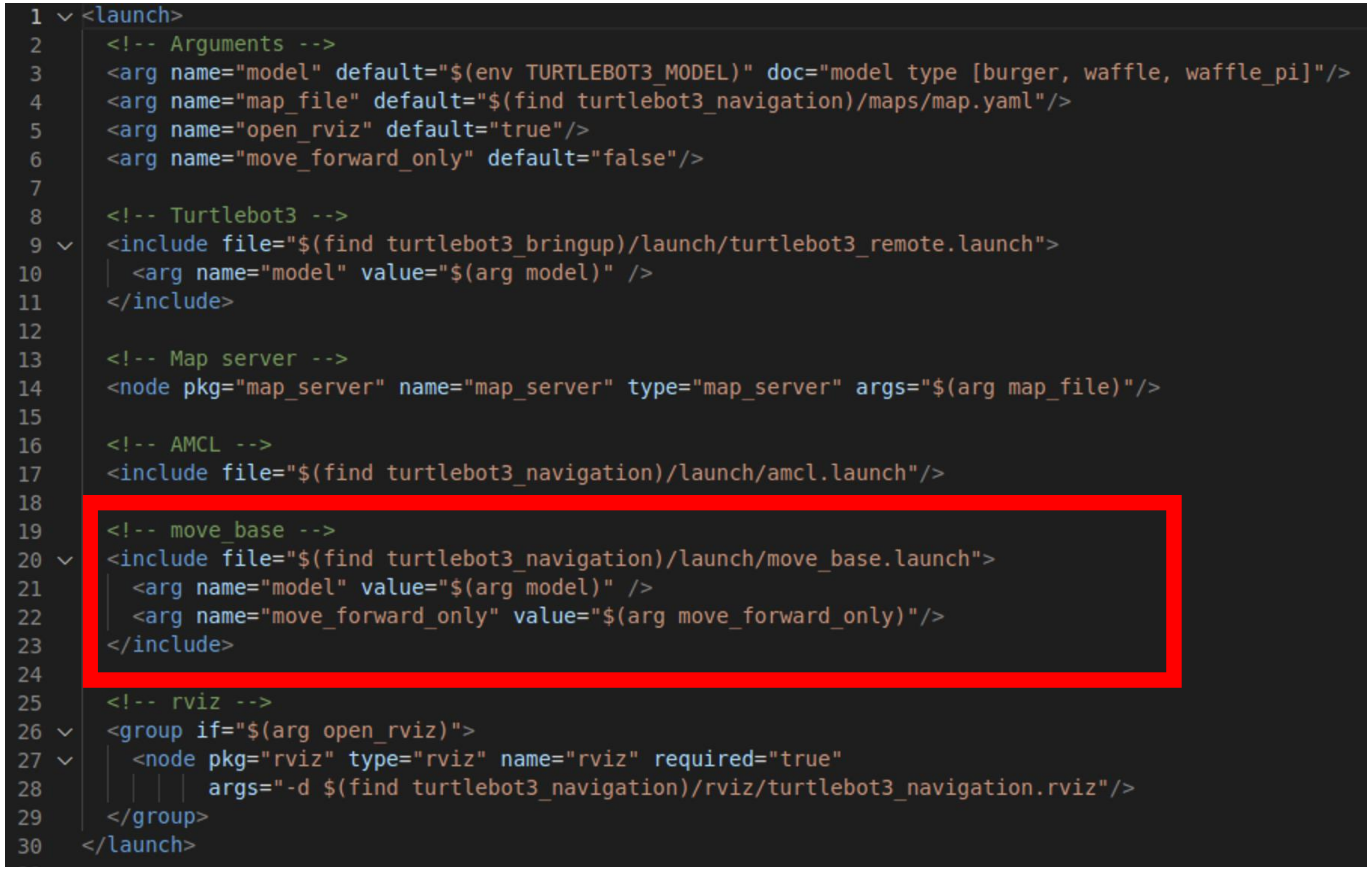
```xml
<launch>
  <!-- Arguments -->
  <arg name="model" default="$(env TURTLEBOT3_MODEL)" doc="model type [burger, waffle, waffle_pi]"/>
  <arg name="map_file" default="$(find turtlebot3_navigation)/maps/map.yaml"/>
  <arg name="open_rviz" default="true"/>
  <arg name="move_forward_only" default="false"/>

  <!-- Turtlebot3 -->
  <include file="$(find turtlebot3_bringup)/launch/turtlebot3_remote.launch">
    <arg name="model" value="$(arg model)" />
  </include>

  <!-- Map server -->
  <node pkg="map_server" name="map_server" type="map_server" args="$(arg map_file)"/>

  <!-- AMCL -->
  <include file="$(find turtlebot3_navigation)/launch/amcl.launch"/>

  <!-- move_base -->
  <include file="$(find turtlebot3_navigation)/launch/move_base.launch">
    <arg name="model" value="$(arg model)" />
    <arg name="move_forward_only" value="$(arg move_forward_only)"/>
  </include>

  <!-- rviz -->
  <group if="$(arg open_rviz)">
    <node pkg="rviz" type="rviz" name="rviz" required="true"
          args="-d $(find turtlebot3_navigation)/rviz/turtlebot3_navigation.rviz"/>
  </group>
</launch>
```

Turtlebot3 - Navigation



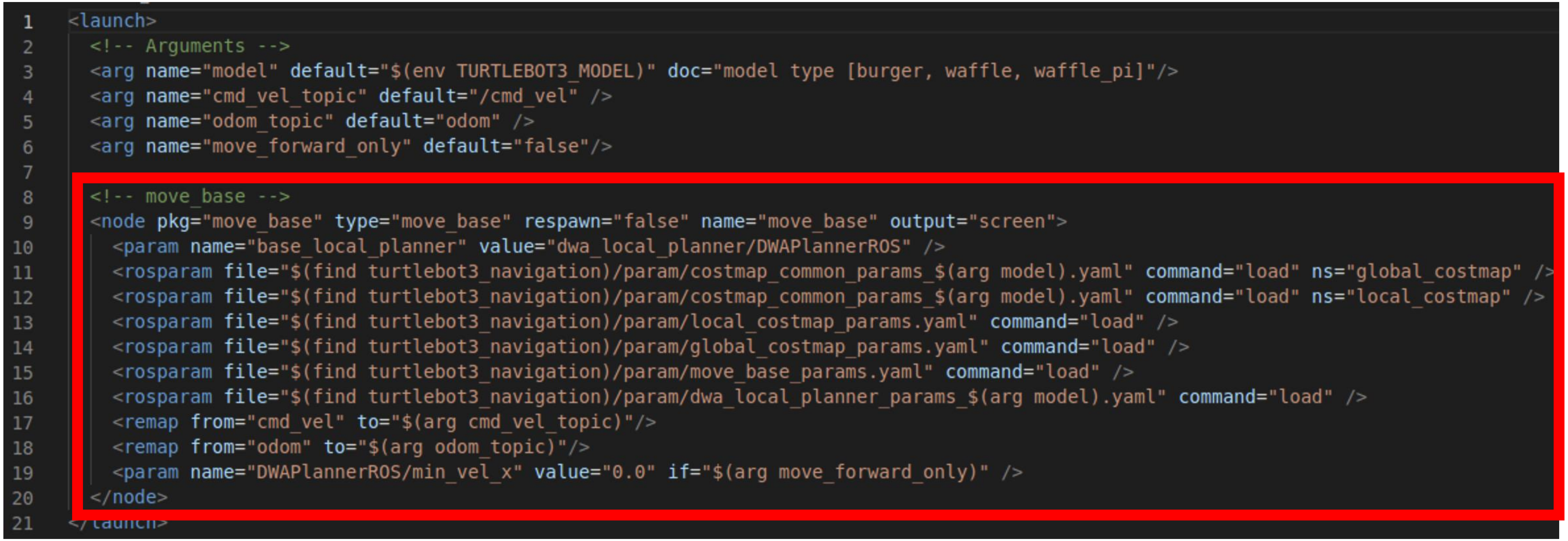
```xml
<launch>
  <!-- Arguments -->
  <arg name="scan_topic"     default="scan"/>
  <arg name="initial_pose_x" default="0.0"/>
  <arg name="initial_pose_y" default="0.0"/>
  <arg name="initial_pose_a" default="0.0"/>

  <!-- AMCL -->
  <node pkg="amcl" type="amcl" name="amcl">

    <param name="min_particles"             value="500"/>
    <param name="max_particles"             value="3000"/>
    <param name="kld_err"                   value="0.02"/>
    <param name="update_min_d"              value="0.20"/>
    <param name="update_min_a"              value="0.20"/>
    <param name="resample_interval"         value="1"/>
    <param name="transform_tolerance"       value="0.5"/>
    <param name="recovery_alpha_slow"       value="0.00"/>
    <param name="recovery_alpha_fast"       value="0.00"/>
    <param name="initial_pose_x"            value="$(arg initial_pose_x)"/>
    <param name="initial_pose_y"            value="$(arg initial_pose_y)"/>
    <param name="initial_pose_a"            value="$(arg initial_pose_a)"/>
    <param name="gui_publish_rate"          value="50.0"/>

    <remap from="scan"                      to="$(arg scan_topic)"/>
    <param name="laser_max_range"           value="3.5"/>
    <param name="laser_max_beams"           value="180"/>
    <param name="laser_z_hit"               value="0.5"/>
    <param name="laser_z_short"             value="0.05"/>
    <param name="laser_z_max"               value="0.05"/>
    <param name="laser_z_rand"              value="0.5"/>
    <param name="laser_sigma_hit"           value="0.2"/>
    <param name="laser_lambda_short"        value="0.1"/>
    <param name="laser_likelihood_max_dist" value="2.0"/>
    <param name="laser_model_type"          value="likelihood_field"/>

    <param name="odom_model_type"           value="diff"/>
    <param name="odom_alpha1"               value="0.1"/>
    <param name="odom_alpha2"               value="0.1"/>
    <param name="odom_alpha3"               value="0.1"/>
    <param name="odom_alpha4"               value="0.1"/>
    <param name="odom_frame_id"             value="odom"/>
    <param name="base_frame_id"             value="base_footprint"/>

  </node>
```

Turtlebot3 - Navigation



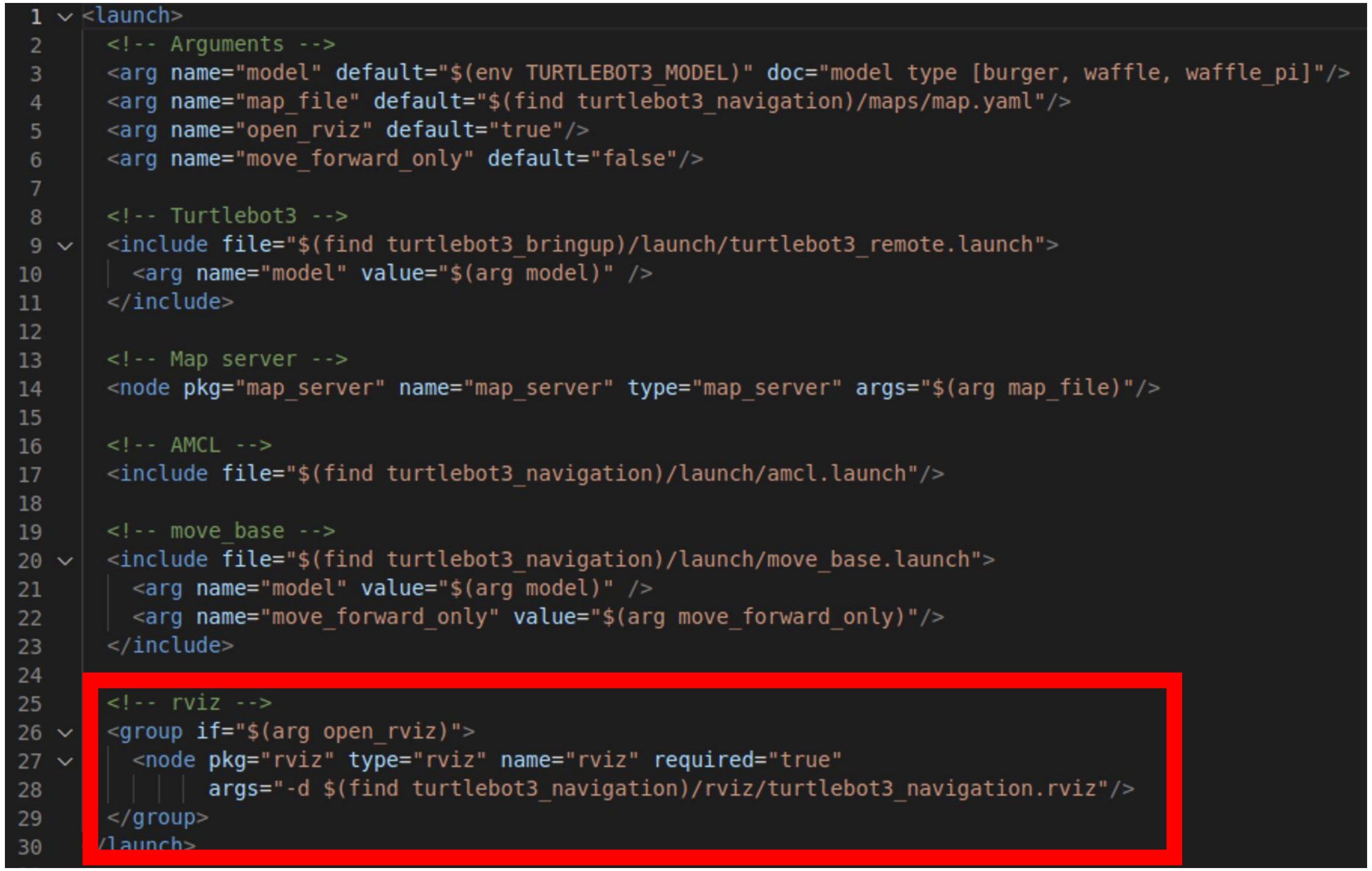
```xml
<launch>
  <!-- Arguments -->
  <arg name="model" default="$(env TURTLEBOT3_MODEL)" doc="model type [burger, waffle, waffle_pi]"/>
  <arg name="map_file" default="$(find turtlebot3_navigation)/maps/map.yaml"/>
  <arg name="open_rviz" default="true"/>
  <arg name="move_forward_only" default="false"/>

  <!-- Turtlebot3 -->
  <include file="$(find turtlebot3_bringup)/launch/turtlebot3_remote.launch">
    <arg name="model" value="$(arg model)" />
  </include>

  <!-- Map server -->
  <node pkg="map_server" name="map_server" type="map_server" args="$(arg map_file)"/>

  <!-- AMCL -->
  <include file="$(find turtlebot3_navigation)/launch/amcl.launch"/>

  <!-- move_base -->
  <include file="$(find turtlebot3_navigation)/launch/move_base.launch">
    <arg name="model" value="$(arg model)" />
    <arg name="move_forward_only" value="$(arg move_forward_only)"/>
  </include>

  <!-- rviz -->
  <group if="$(arg open_rviz)">
    <node pkg="rviz" type="rviz" name="rviz" required="true"
          args="-d $(find turtlebot3_navigation)/rviz/turtlebot3_navigation.rviz"/>
  </group>
</launch>
```

Turtlebot3 - Navigation



```
<launch>
  <!-- Arguments -->
  <arg name="model" default="$(env TURTLEBOT3_MODEL)" doc="model type [burger, waffle, waffle_pi]"/>
  <arg name="cmd_vel_topic" default="/cmd_vel" />
  <arg name="odom_topic" default="odom" />
  <arg name="move_forward_only" default="false"/>

  <!-- move_base -->
  <node pkg="move_base" type="move_base" respawn="false" name="move_base" output="screen">
    <param name="base_local_planner" value="dwa_local_planner/DWAPlannerROS" />
    <rosparam file="$(find turtlebot3_navigation)/param/costmap_common_params_$(arg model).yaml" command="load" ns="global_costmap" />
    <rosparam file="$(find turtlebot3_navigation)/param/costmap_common_params_$(arg model).yaml" command="load" ns="local_costmap" />
    <rosparam file="$(find turtlebot3_navigation)/param/local_costmap_params.yaml" command="load" />
    <rosparam file="$(find turtlebot3_navigation)/param/global_costmap_params.yaml" command="load" />
    <rosparam file="$(find turtlebot3_navigation)/param/move_base_params.yaml" command="load" />
    <rosparam file="$(find turtlebot3_navigation)/param/dwa_local_planner_params_$(arg model).yaml" command="load" />
    <remap from="cmd_vel" to="$(arg cmd_vel_topic)"/>
    <remap from="odom" to="$(arg odom_topic)"/>
    <param name="DWAPlannerROS/min_vel_x" value="0.0" if="$(arg move_forward_only)" />
  </node>
</launch>
```

Turtlebot3 - Navigation



```
<launch>
  <!-- Arguments -->
  <arg name="model" default="$(env TURTLEBOT3_MODEL)" doc="model type [burger, waffle, waffle_pi]"/>
  <arg name="map_file" default="$(find turtlebot3_navigation)/maps/map.yaml"/>
  <arg name="open_rviz" default="true"/>
  <arg name="move_forward_only" default="false"/>

  <!-- Turtlebot3 -->
  <include file="$(find turtlebot3_bringup)/launch/turtlebot3_remote.launch">
    <arg name="model" value="$(arg model)" />
  </include>

  <!-- Map server -->
  <node pkg="map_server" name="map_server" type="map_server" args="$(arg map_file)"/>

  <!-- AMCL -->
  <include file="$(find turtlebot3_navigation)/launch/amcl.launch"/>

  <!-- move_base -->
  <include file="$(find turtlebot3_navigation)/launch/move_base.launch">
    <arg name="model" value="$(arg model)" />
    <arg name="move_forward_only" value="$(arg move_forward_only)"/>
  </include>

  <!-- rviz -->
  <group if="$(arg open_rviz)">
    <node pkg="rviz" type="rviz" name="rviz" required="true"
          args="-d $(find turtlebot3_navigation)/rviz/turtlebot3_navigation.rviz"/>
  </group>
</launch>
```

Turtlebot3 - Navigation

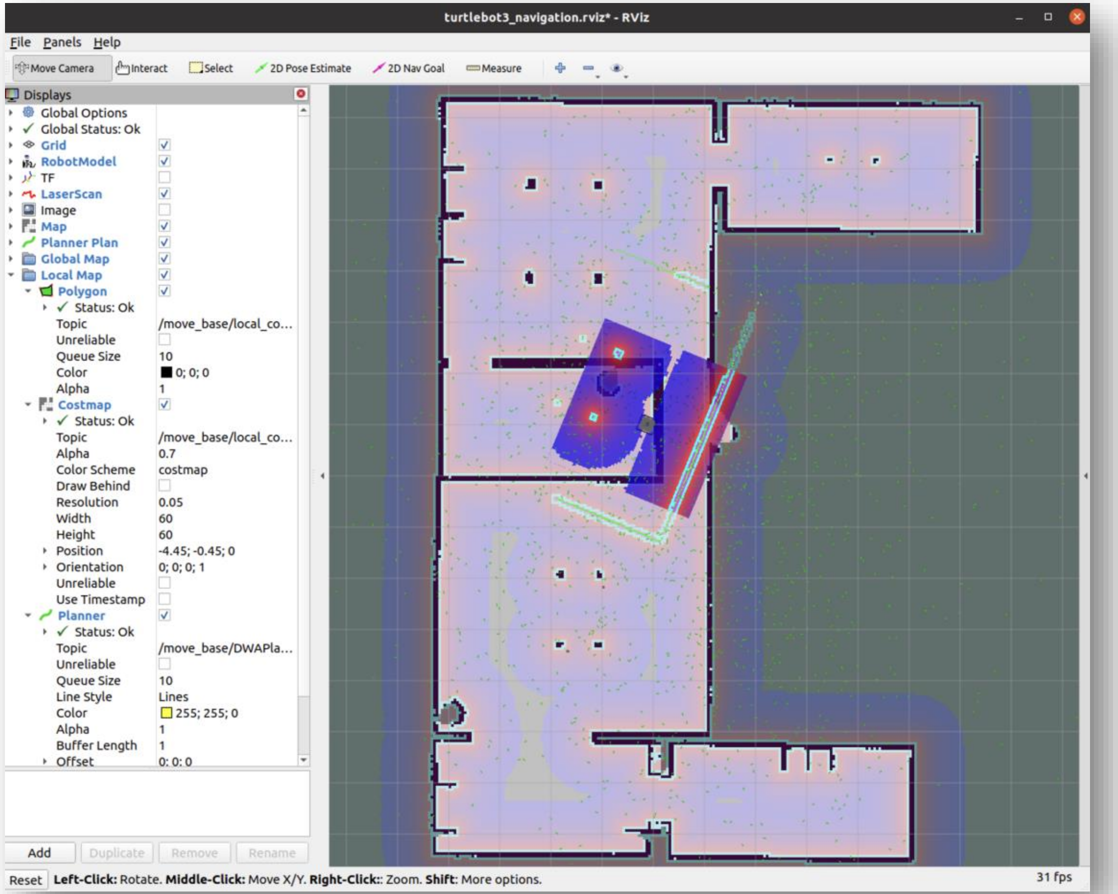
```
$ export TURTLEBOT3_MODEL=burger
```

```
$ roslaunch turtlebot3_navigation turtlebot3_navigation.launch map_file:=$HOME/map.yaml
```

Turtlebot3 - Navigation

```
$ sudo apt install ros-noetic-dwa-local-planner
```

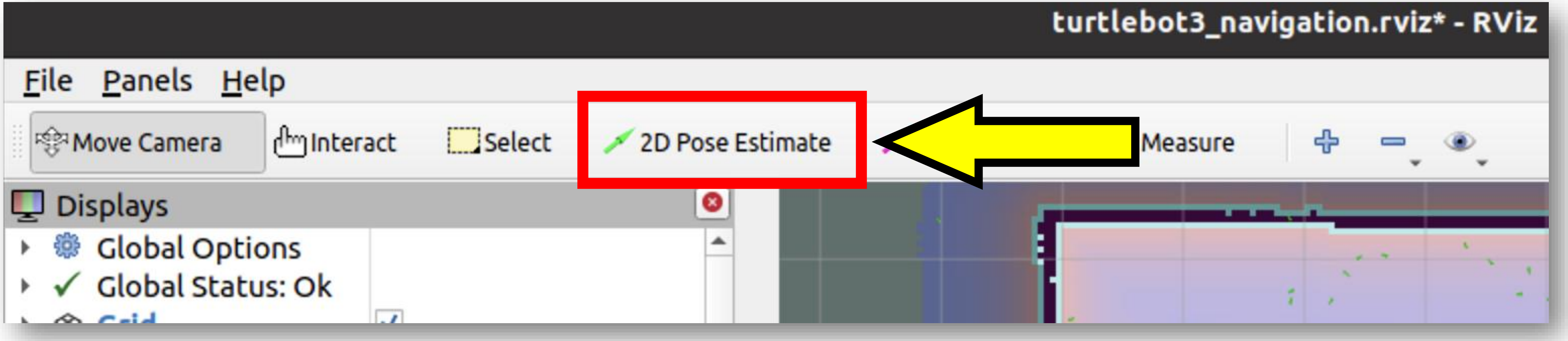
Turtlebot3 - Navigation



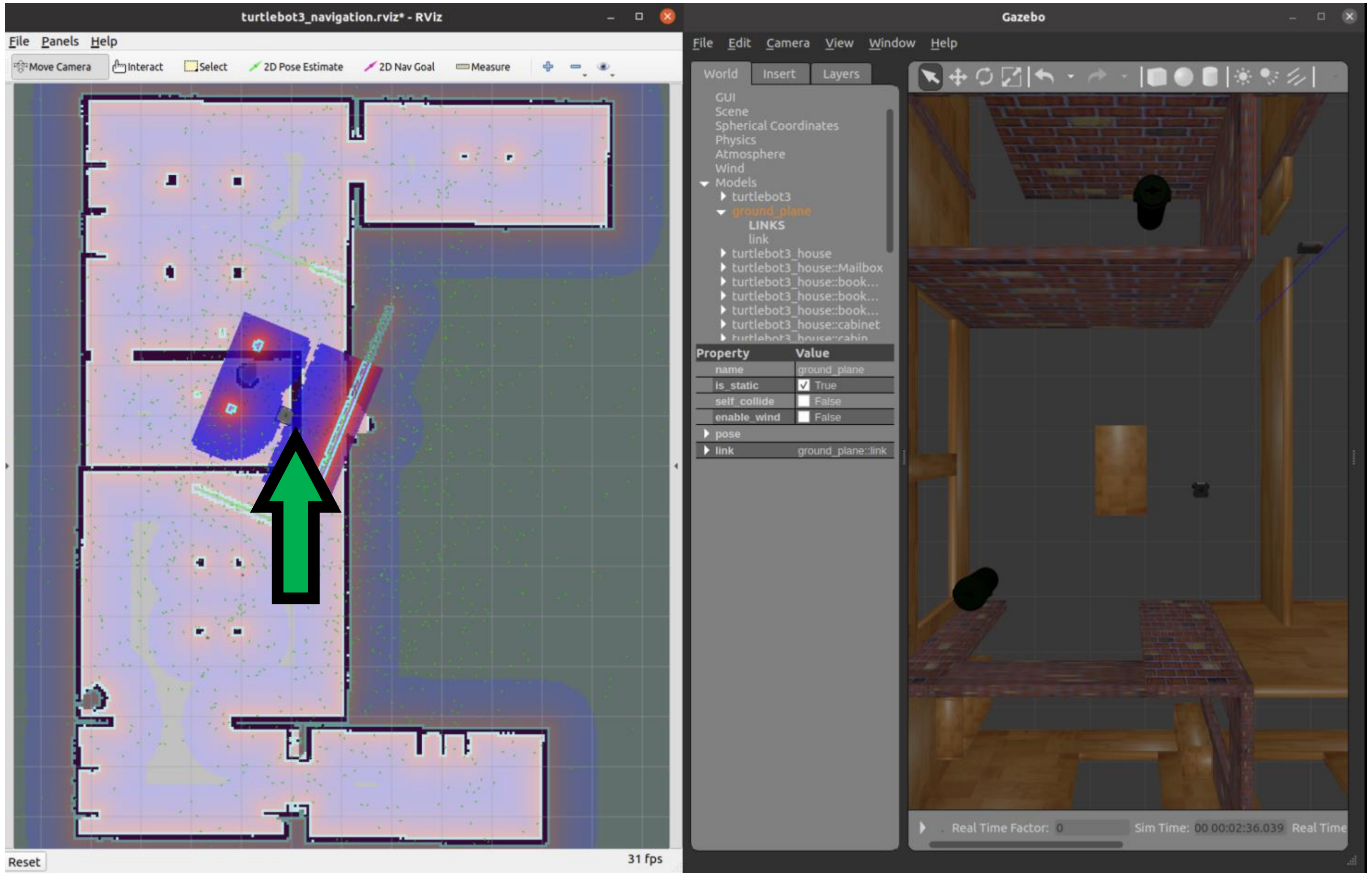
Global Options	
✓ Global Status: Ok	
Grid	☑
RobotModel	☑
TF	□
LaserScan	☑
Image	□
Map	☑
Planner Plan	☑
Global Map	☑
Local Map	☑
Polygon	☑
✓ Status: Ok	
Topic	/move_base/local_co...
Unreliable	□
Queue Size	10
Color	■ 0; 0; 0
Alpha	1
Costmap	☑
✓ Status: Ok	
Topic	/move_base/local_co...
Alpha	0.7
Color Scheme	costmap
Draw Behind	□
Resolution	0.05
Width	60
Height	60
Position	-4.45; -0.45; 0
Orientation	0; 0; 0; 1
Unreliable	□
Use Timestamp	□
Planner	☑
✓ Status: Ok	
Topic	/move_base/DWAPla...
Unreliable	□
Queue Size	10
Line Style	Lines
Color	□ 255; 255; 0
Alpha	1
Buffer Length	1
Offset	0; 0; 0

Left-Click: Rotate. **Middle-Click:** Move X/Y. **Right-Click:**: Zoom. **Shift:** More options.

Turtlebot3 - Navigation



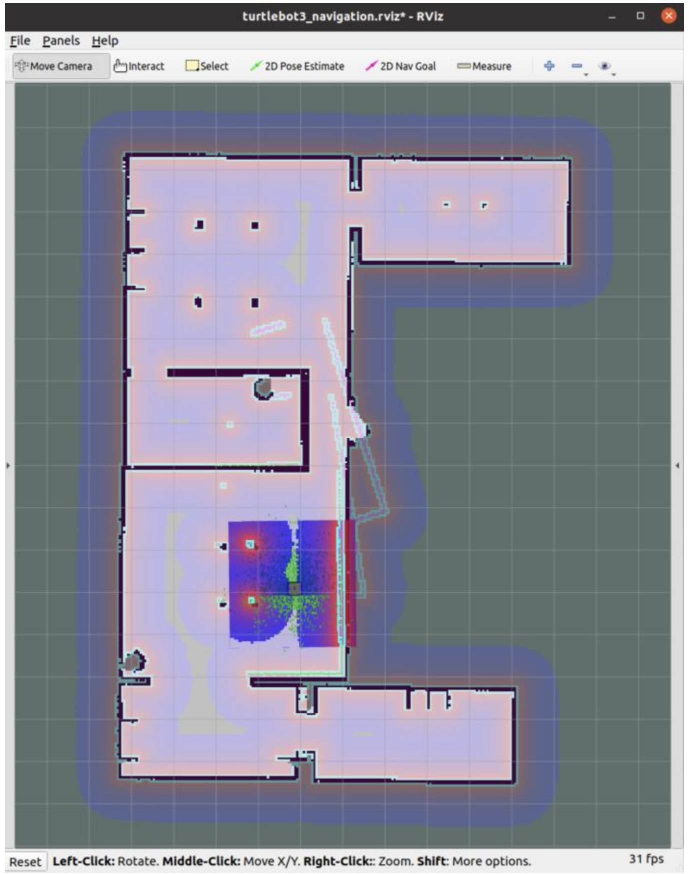
Turtlebot3 - Navigation



Property	Value
name	ground_plane
is_static	☑ True
self_collide	□ False
enable_wind	□ False
pose	
link	ground_plane::link

Real Time Factor: 0 Sim Time: 00 00:02:36.039 Real Time

Turtlebot3 - Navigation



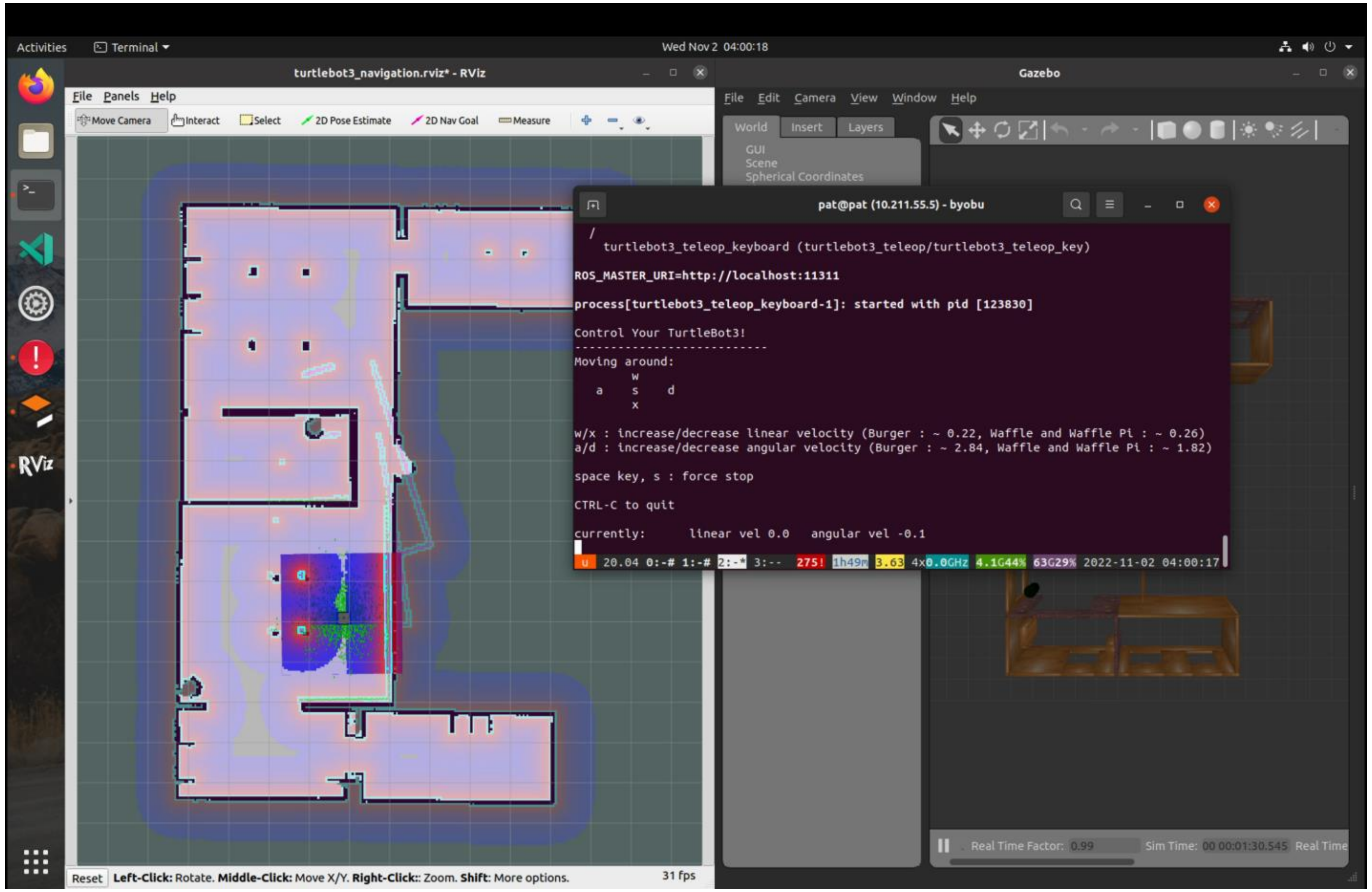
turtlebot3_navigation.rviz* - RViz
File Panels Help
Move Camera
Interact
Select
2D Pose Estimate
2D Nav Goal
Measure
Reset
Left-Click: Rotate. Middle-Click: Move X/Y. Right-Click:: Zoom. Shift: More options.
31 fps

Turtlebot3 - Navigation

Try to rotate your robot.

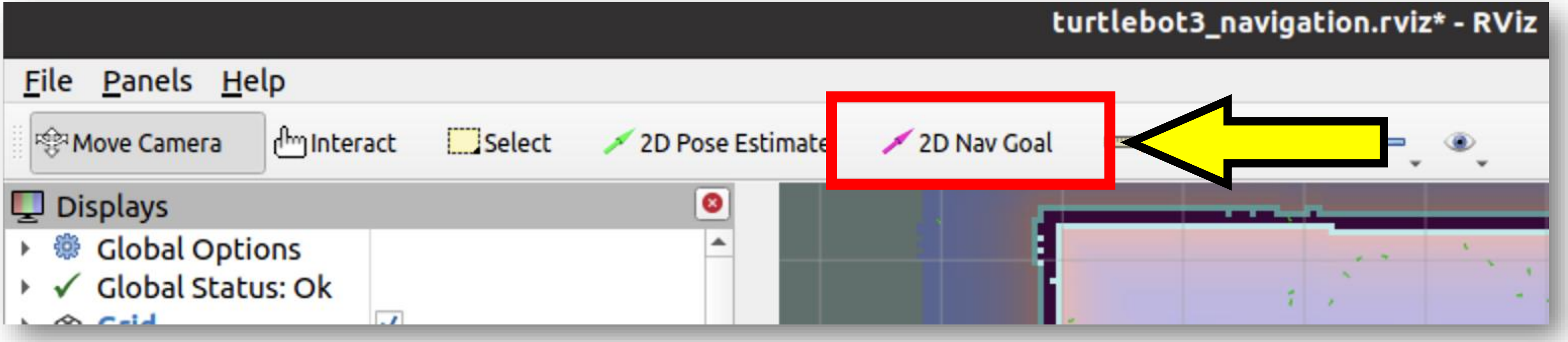
```
$ roslaunch turtlebot3_teleop turtlebot3_teleop_key.launch
```

Turtlebot3 - Navigation

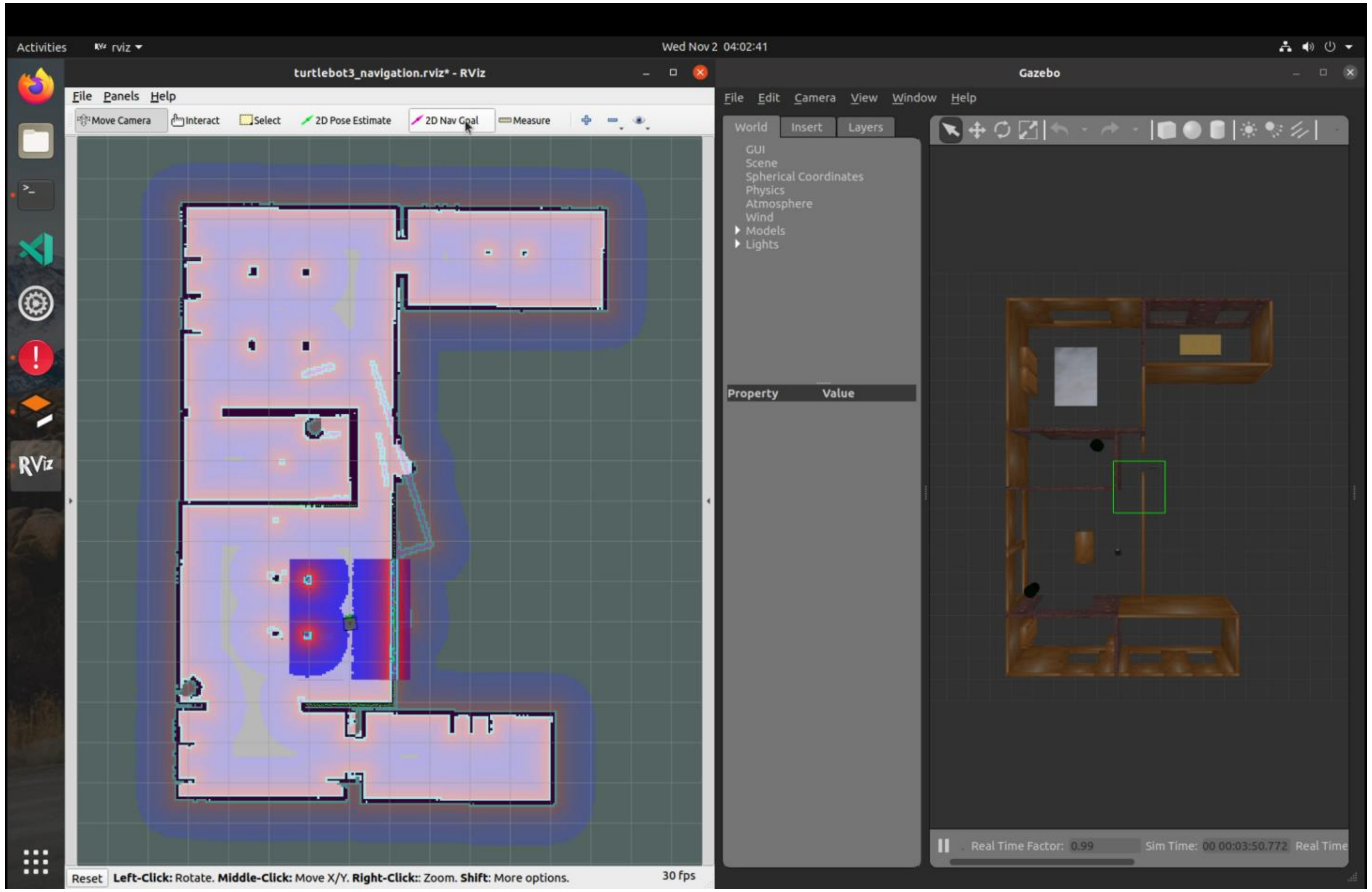


```
 /
   turtlebot3_teleop_keyboard (turtlebot3_teleop/turtlebot3_teleop_key)

ROS_MASTER_URI=http://localhost:11311

process[turtlebot3_teleop_keyboard-1]: started with pid [123830]

Control Your TurtleBot3!
---------------------------
Moving around:
        w
   a    s    d
        x

w/x : increase/decrease linear velocity (Burger : ~ 0.22, Waffle and Waffle Pi : ~ 0.26)
a/d : increase/decrease angular velocity (Burger : ~ 2.84, Waffle and Waffle Pi : ~ 1.82)

space key, s : force stop

CTRL-C to quit

currently:      linear vel 0.0   angular vel -0.1

u 20.04 0:-# 1:-# 2:-* 3:--  275! 1h49m 3.63 4x0.0GHz 4.1G44% 63G29% 2022-11-02 04:00:17
```

Turtlebot3 - Navigation



Turtlebot3 - Navigation



Left-Click: Rotate. Middle-Click: Move X/Y. Right-Click:: Zoom. Shift: More options.

Navigation in Python3

Developing node for navigation

Navigation in Python3

- Create directory csv in your_package

```
$ roscd your_package
```

```
$ mkdir csv
```

- Create empty file name location.csv inside directory csv

```
$ cd csv
```

```
$ touch location.csv
```

Navigation in Python3

- Create directory nodes in your_package

```
$ roscd your_package
```

```
$ mkdir nodes
```

- Add python file name navigation_node.py

```
$ cd nodes
```

```
$ gedit navigation_node.py
```

Navigation in Python3

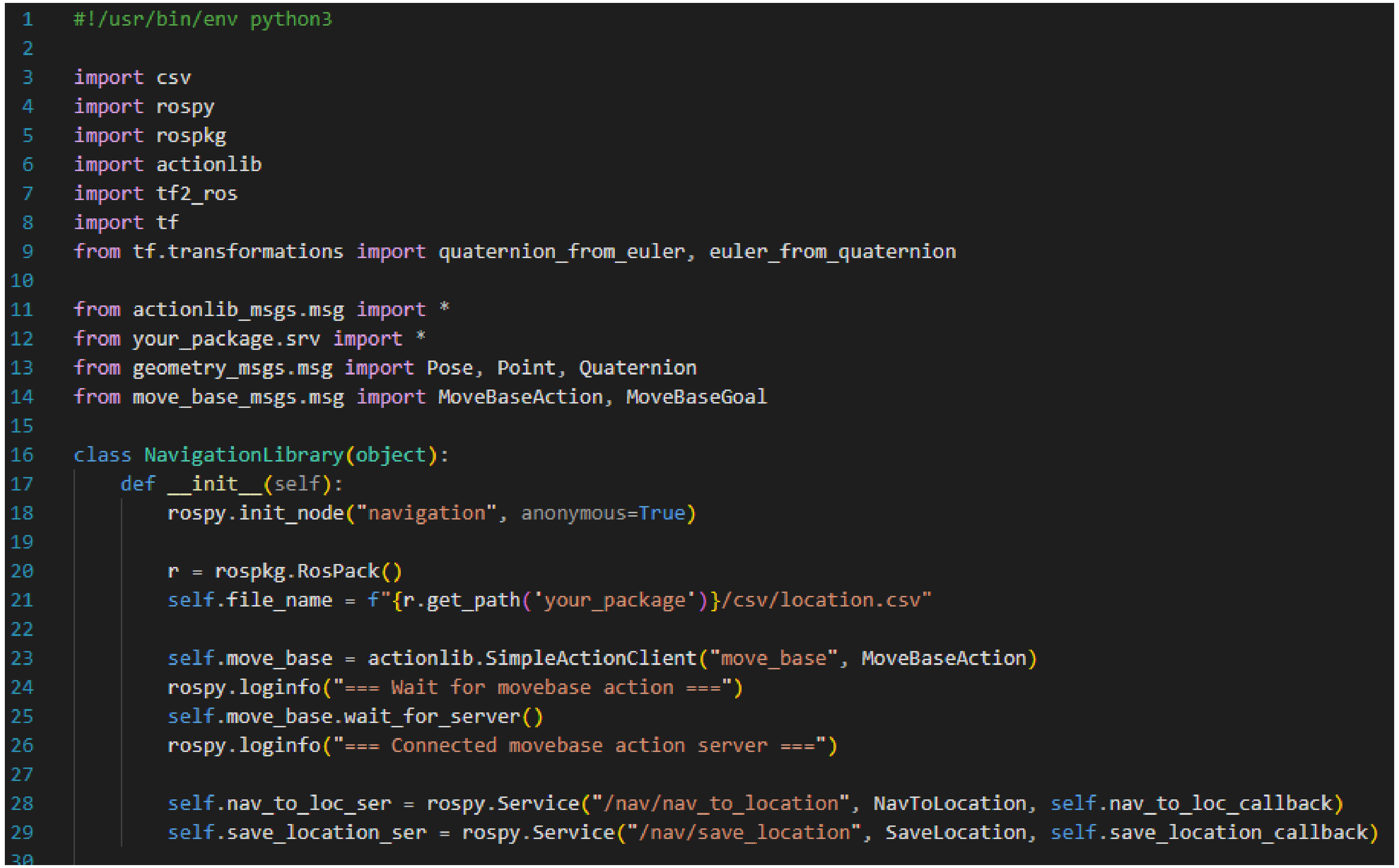
- Copy code from this link and paste to navigation_node.py

navigation_node.py

- Save the file and exit
- Change navigation_node.py to executable file

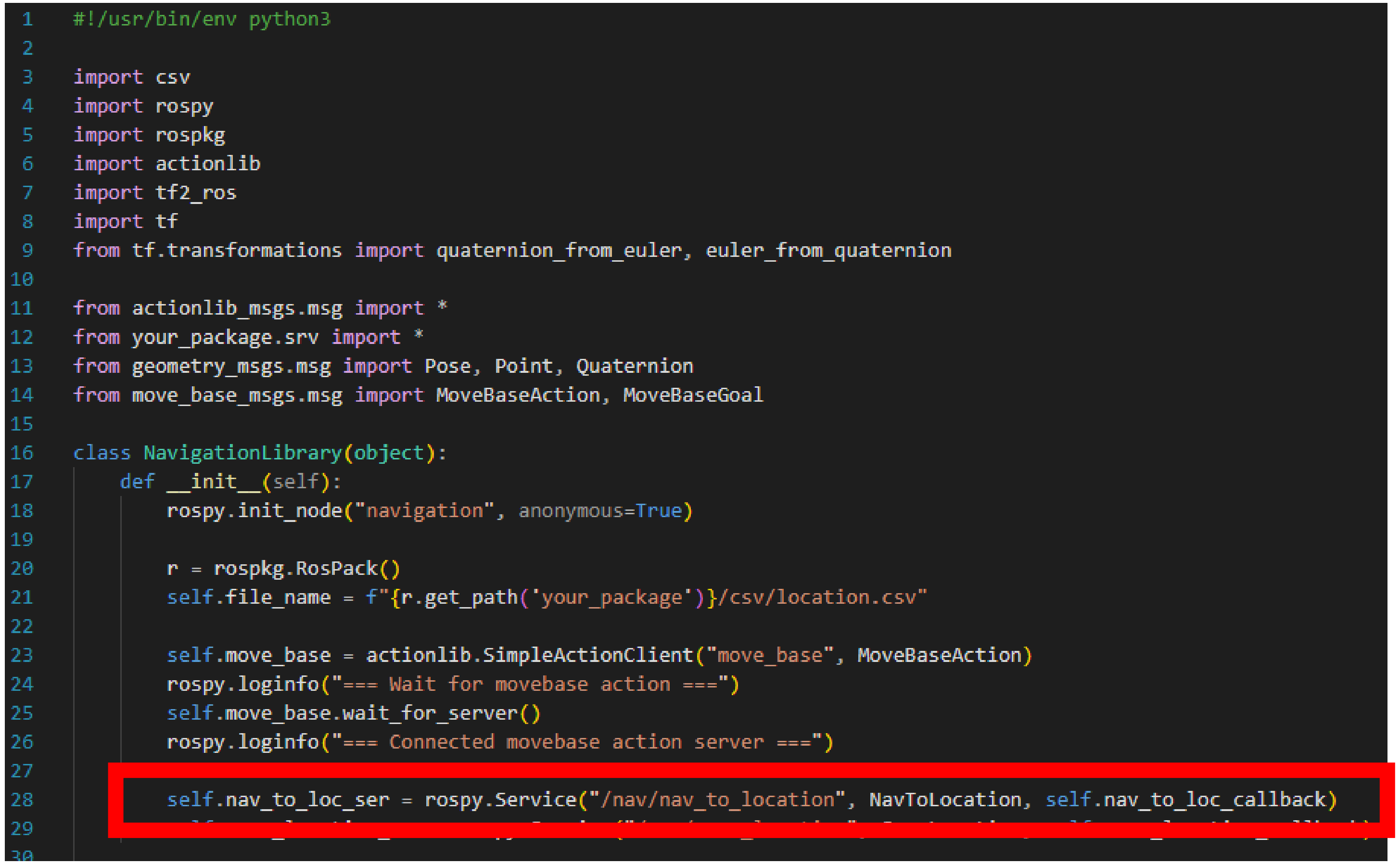
```
$ chmod +x navigation_node.py
```

Navigation in Python3



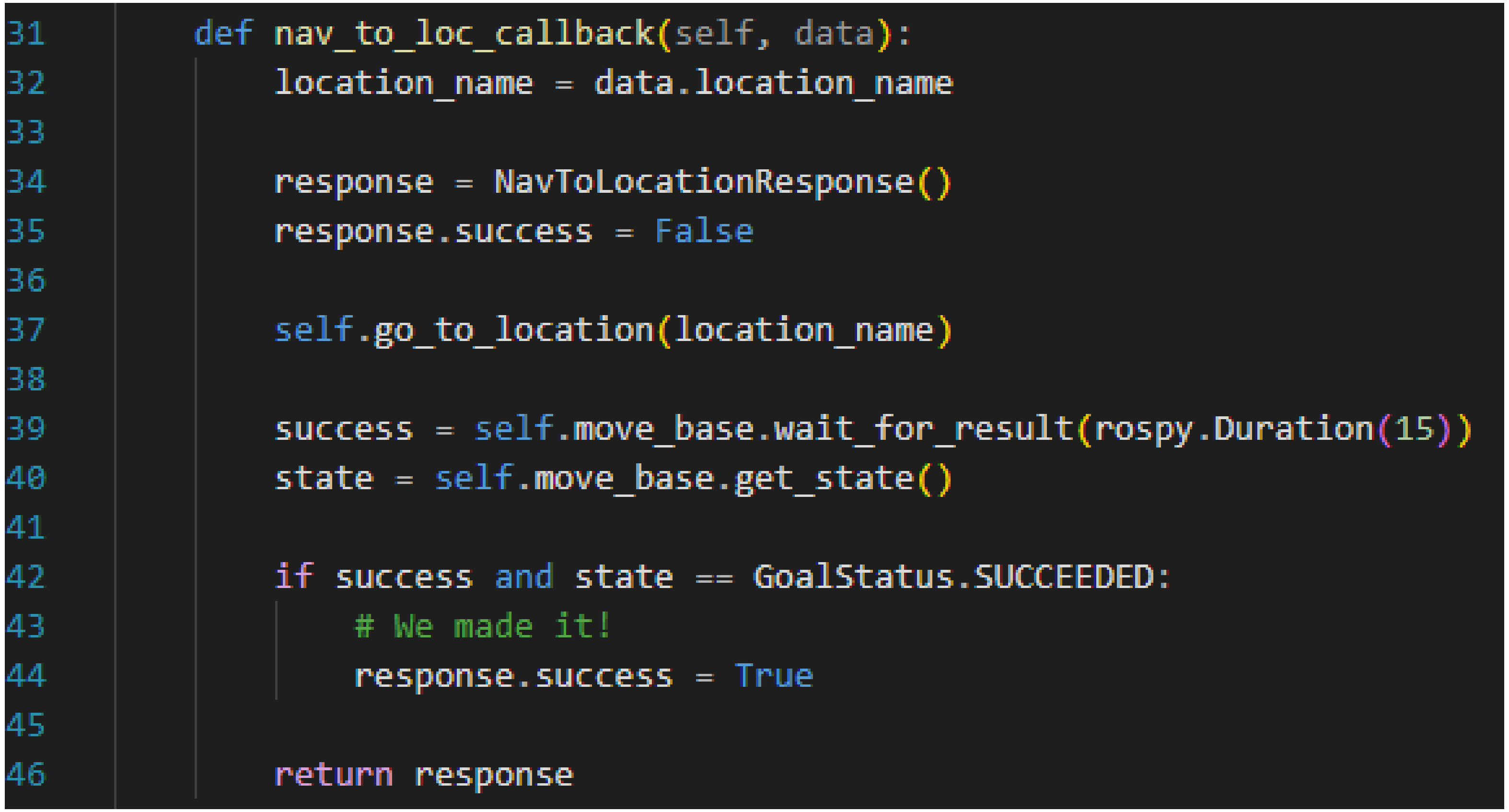
```python
#!/usr/bin/env python3

import csv
import rospy
import rospkg
import actionlib
import tf2_ros
import tf
from tf.transformations import quaternion_from_euler, euler_from_quaternion

from actionlib_msgs.msg import *
from your_package.srv import *
from geometry_msgs.msg import Pose, Point, Quaternion
from move_base_msgs.msg import MoveBaseAction, MoveBaseGoal

class NavigationLibrary(object):
    def __init__(self):
        rospy.init_node("navigation", anonymous=True)

        r = rospkg.RosPack()
        self.file_name = f"{r.get_path('your_package')}/csv/location.csv"

        self.move_base = actionlib.SimpleActionClient("move_base", MoveBaseAction)
        rospy.loginfo("=== Wait for movebase action ===")
        self.move_base.wait_for_server()
        rospy.loginfo("=== Connected movebase action server ===")

        self.nav_to_loc_ser = rospy.Service("/nav/nav_to_location", NavToLocation, self.nav_to_loc_callback)
        self.save_location_ser = rospy.Service("/nav/save_location", SaveLocation, self.save_location_callback)
```

Navigation in Python3



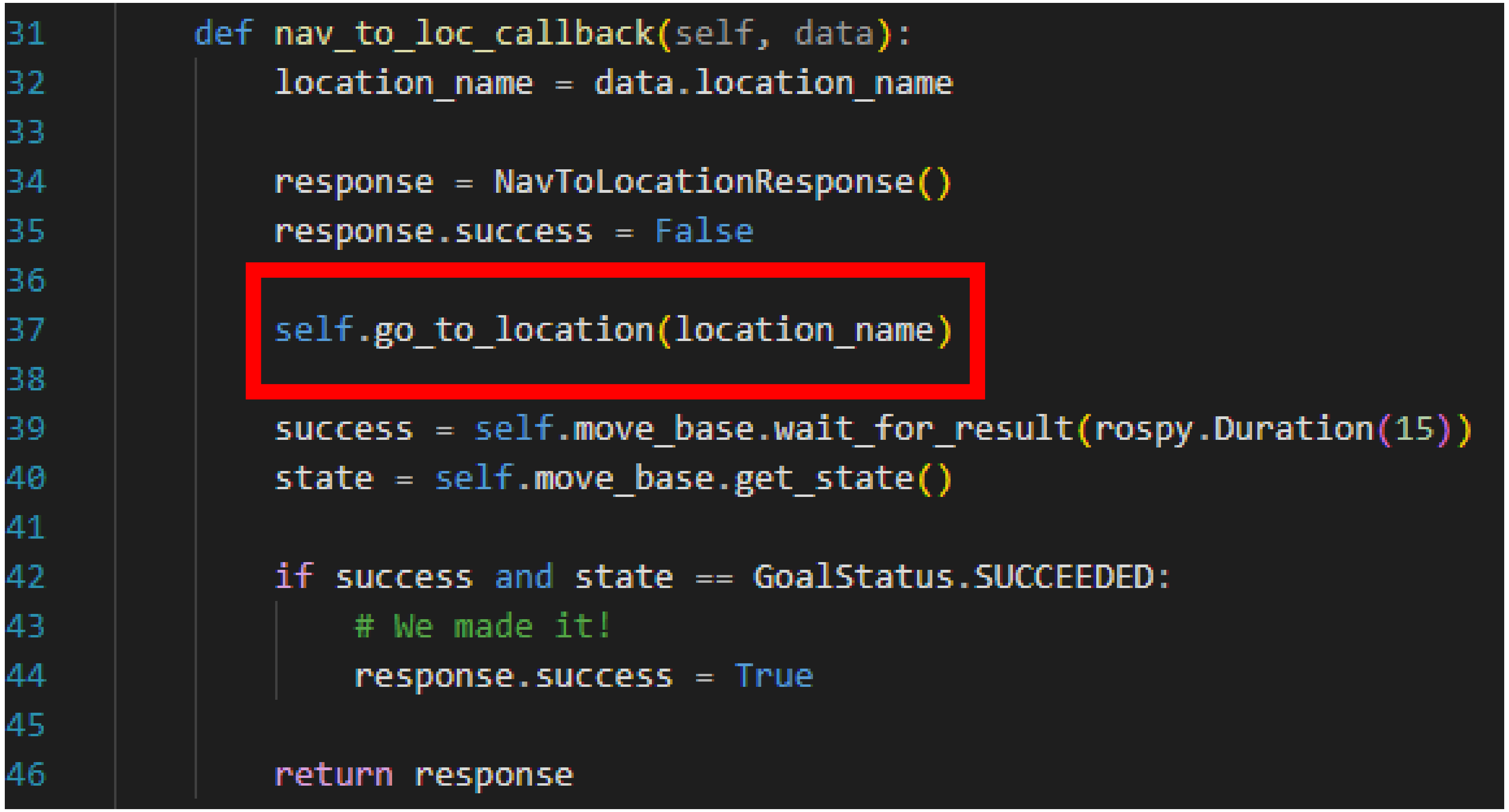
```python
#!/usr/bin/env python3

import csv
import rospy
import rospkg
import actionlib
import tf2_ros
import tf
from tf.transformations import quaternion_from_euler, euler_from_quaternion

from actionlib_msgs.msg import *
from your_package.srv import *
from geometry_msgs.msg import Pose, Point, Quaternion
from move_base_msgs.msg import MoveBaseAction, MoveBaseGoal

class NavigationLibrary(object):
    def __init__(self):
        rospy.init_node("navigation", anonymous=True)

        r = rospkg.RosPack()
        self.file_name = f"{r.get_path('your_package')}/csv/location.csv"

        self.move_base = actionlib.SimpleActionClient("move_base", MoveBaseAction)
        rospy.loginfo("=== Wait for movebase action ===")
        self.move_base.wait_for_server()
        rospy.loginfo("=== Connected movebase action server ===")

        self.nav_to_loc_ser = rospy.Service("/nav/nav_to_location", NavToLocation, self.nav_to_loc_callback)
```

Navigation in Python3



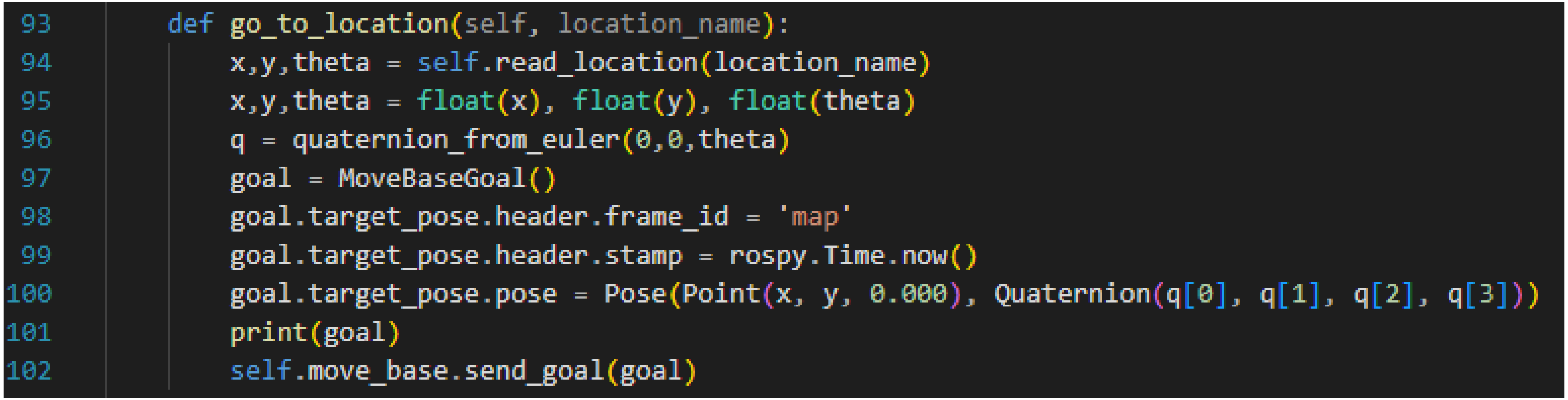
```python
    def nav_to_loc_callback(self, data):
        location_name = data.location_name

        response = NavToLocationResponse()
        response.success = False

        self.go_to_location(location_name)

        success = self.move_base.wait_for_result(rospy.Duration(15))
        state = self.move_base.get_state()

        if success and state == GoalStatus.SUCCEEDED:
            # We made it!
            response.success = True

        return response
```

Navigation in Python3



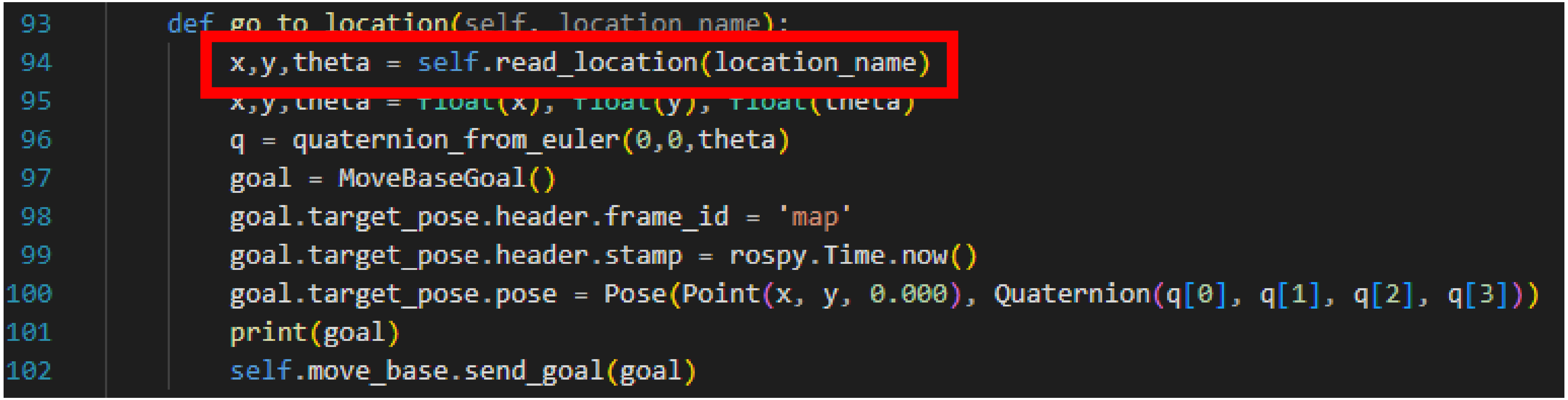
```python
    def nav_to_loc_callback(self, data):
        location_name = data.location_name

        response = NavToLocationResponse()
        response.success = False

        self.go_to_location(location_name)

        success = self.move_base.wait_for_result(rospy.Duration(15))
        state = self.move_base.get_state()

        if success and state == GoalStatus.SUCCEEDED:
            # We made it!
            response.success = True

        return response
```

Navigation in Python3



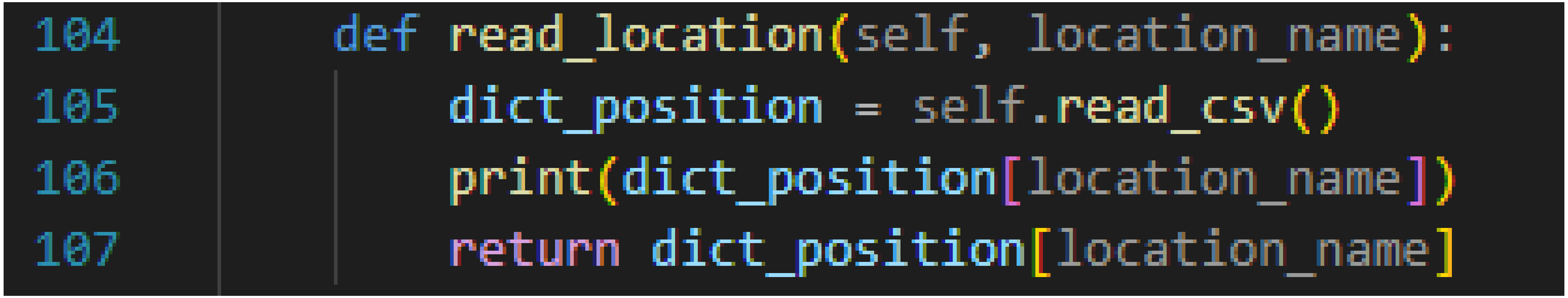
```
    def go_to_location(self, location_name):
        x,y,theta = self.read_location(location_name)
        x,y,theta = float(x), float(y), float(theta)
        q = quaternion_from_euler(0,0,theta)
        goal = MoveBaseGoal()
        goal.target_pose.header.frame_id = 'map'
        goal.target_pose.header.stamp = rospy.Time.now()
        goal.target_pose.pose = Pose(Point(x, y, 0.000), Quaternion(q[0], q[1], q[2], q[3]))
        print(goal)
        self.move_base.send_goal(goal)
```

Navigation in Python3



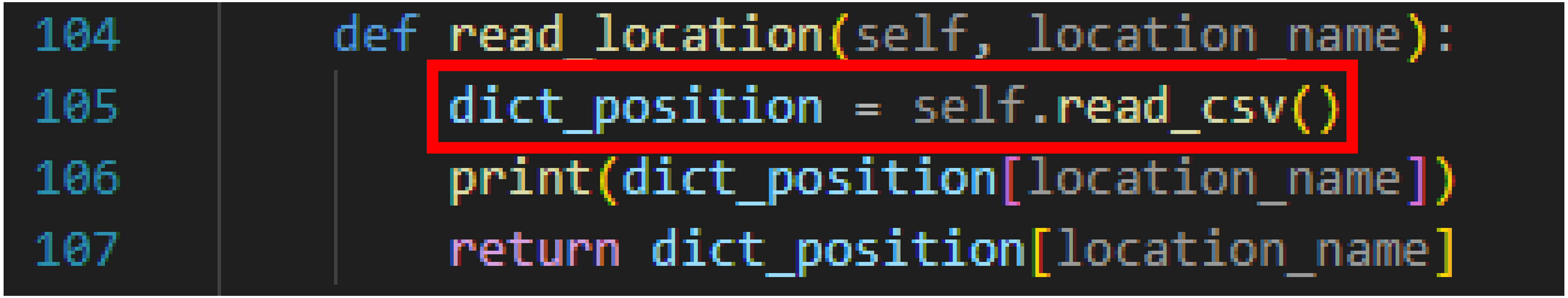
```python
    def go_to_location(self, location_name):
        x,y,theta = self.read_location(location_name)
        x,y,theta = float(x), float(y), float(theta)
        q = quaternion_from_euler(0,0,theta)
        goal = MoveBaseGoal()
        goal.target_pose.header.frame_id = 'map'
        goal.target_pose.header.stamp = rospy.Time.now()
        goal.target_pose.pose = Pose(Point(x, y, 0.000), Quaternion(q[0], q[1], q[2], q[3]))
        print(goal)
        self.move_base.send_goal(goal)
```

Navigation in Python3



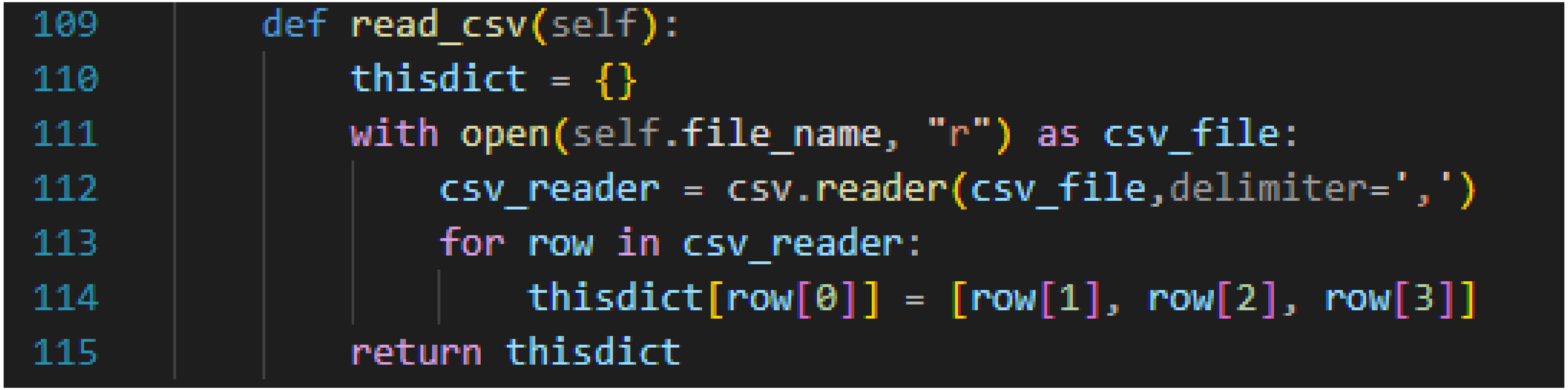
```python
    def read_location(self, location_name):
        dict_position = self.read_csv()
        print(dict_position[location_name])
        return dict_position[location_name]
```

Navigation in Python3



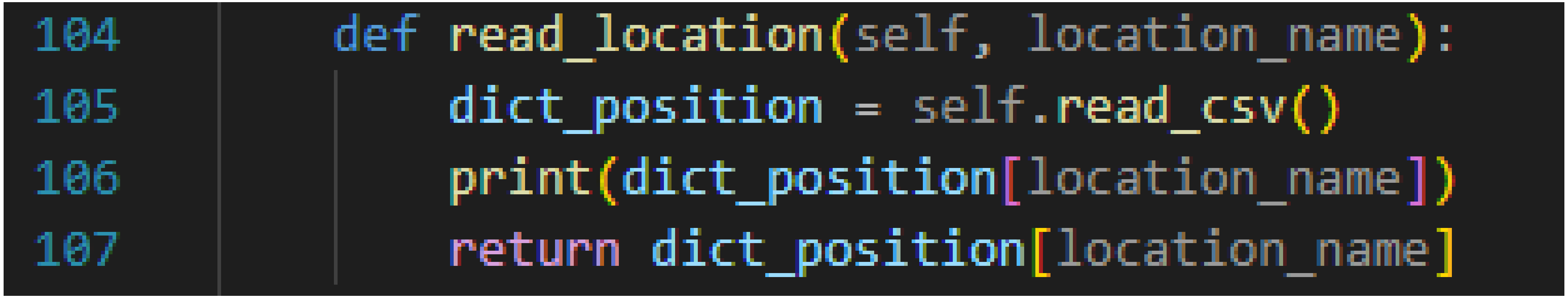
```
    def read_location(self, location_name):
        dict_position = self.read_csv()
        print(dict_position[location_name])
        return dict_position[location_name]
```

Navigation in Python3



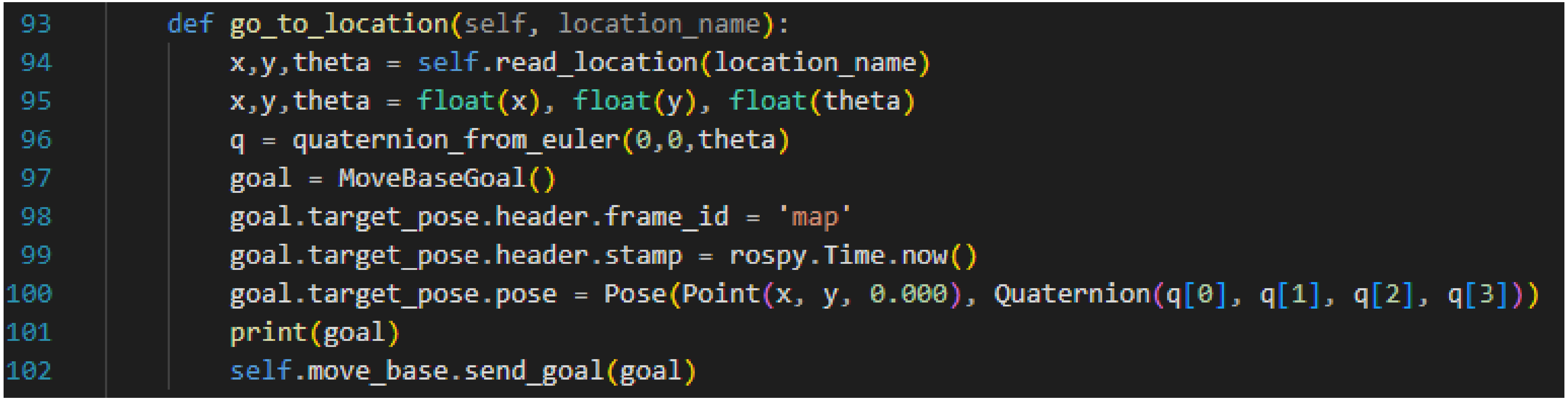
```
    def read_csv(self):
        thisdict = {}
        with open(self.file_name, "r") as csv_file:
            csv_reader = csv.reader(csv_file,delimiter=',')
            for row in csv_reader:
                thisdict[row[0]] = [row[1], row[2], row[3]]
        return thisdict
```

Navigation in Python3



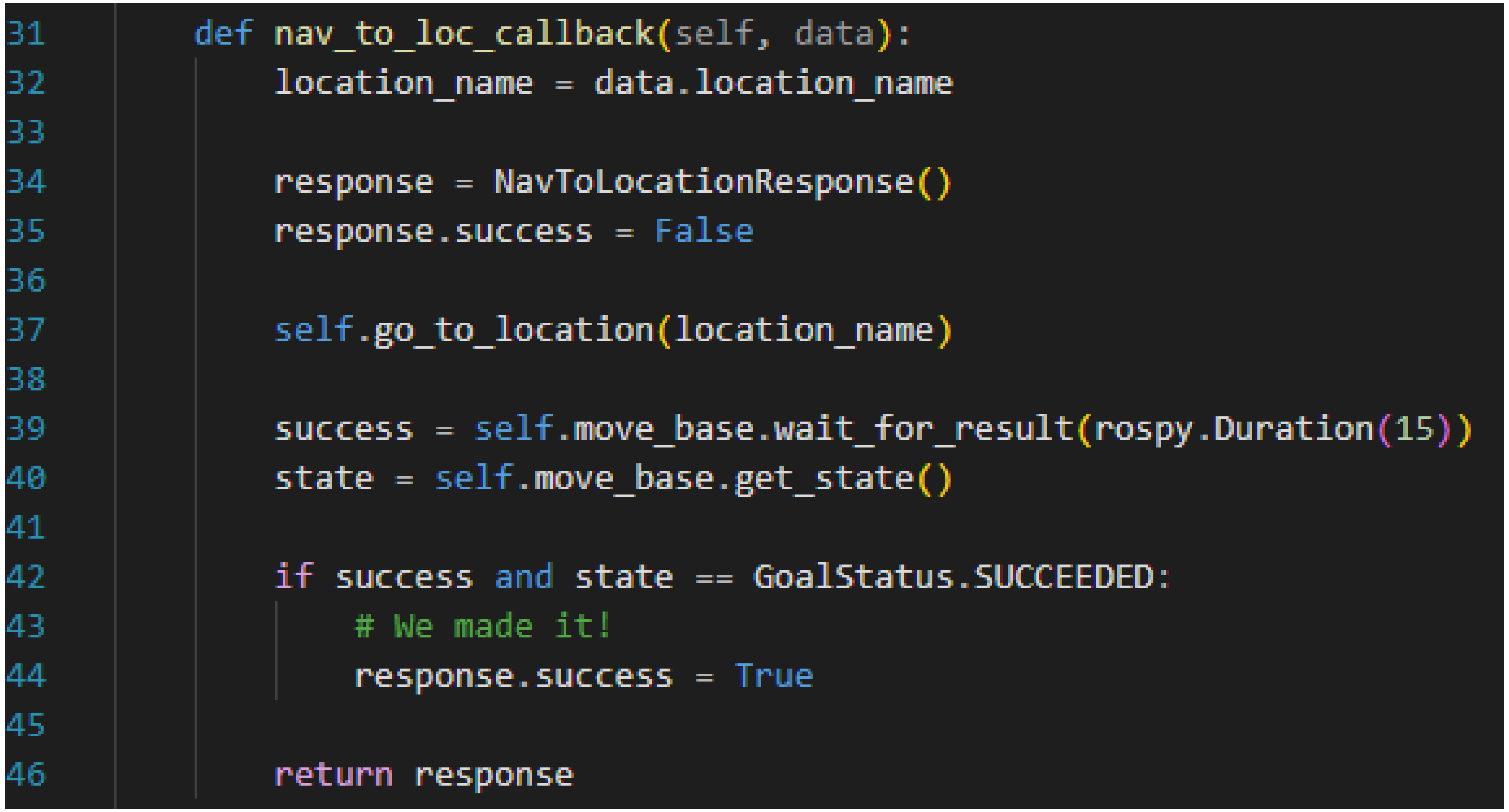
```python
    def read_location(self, location_name):
        dict_position = self.read_csv()
        print(dict_position[location_name])
        return dict_position[location_name]
```

Navigation in Python3



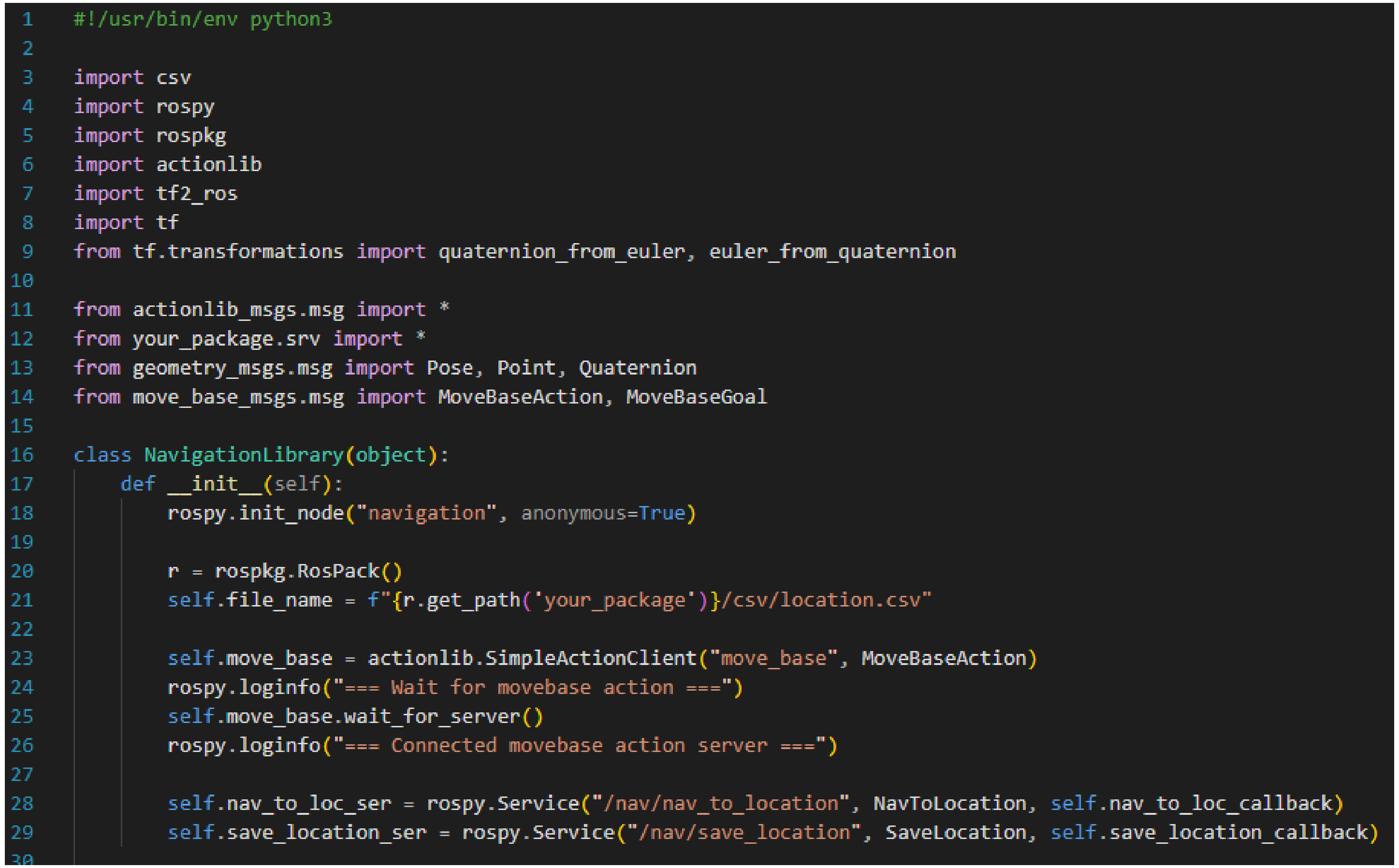
```
    def go_to_location(self, location_name):
        x,y,theta = self.read_location(location_name)
        x,y,theta = float(x), float(y), float(theta)
        q = quaternion_from_euler(0,0,theta)
        goal = MoveBaseGoal()
        goal.target_pose.header.frame_id = 'map'
        goal.target_pose.header.stamp = rospy.Time.now()
        goal.target_pose.pose = Pose(Point(x, y, 0.000), Quaternion(q[0], q[1], q[2], q[3]))
        print(goal)
        self.move_base.send_goal(goal)
```

Navigation in Python3



```python
    def nav_to_loc_callback(self, data):
        location_name = data.location_name

        response = NavToLocationResponse()
        response.success = False

        self.go_to_location(location_name)

        success = self.move_base.wait_for_result(rospy.Duration(15))
        state = self.move_base.get_state()

        if success and state == GoalStatus.SUCCEEDED:
            # We made it!
            response.success = True

        return response
```

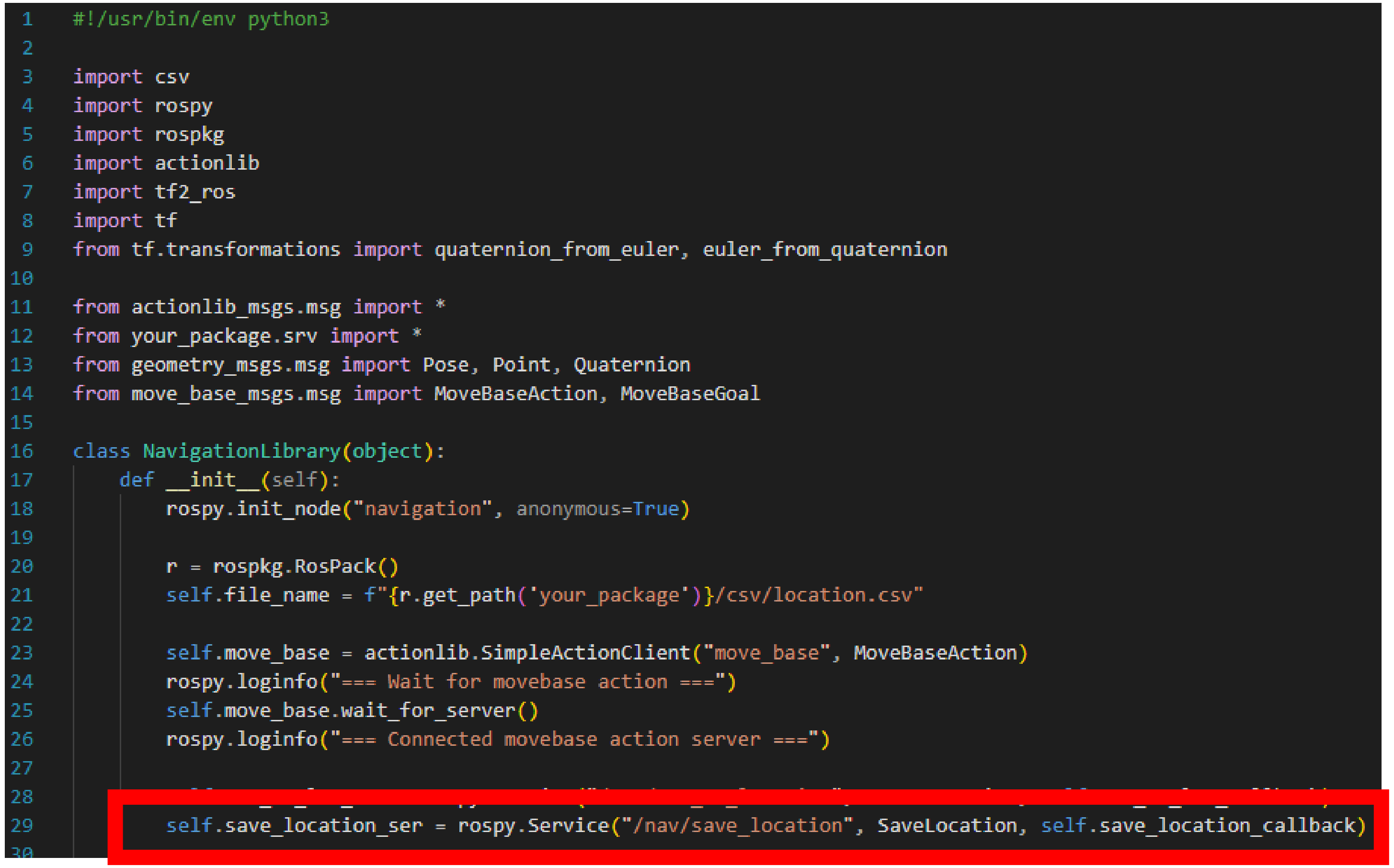
Navigation in Python3



```python
#!/usr/bin/env python3

import csv
import rospy
import rospkg
import actionlib
import tf2_ros
import tf
from tf.transformations import quaternion_from_euler, euler_from_quaternion

from actionlib_msgs.msg import *
from your_package.srv import *
from geometry_msgs.msg import Pose, Point, Quaternion
from move_base_msgs.msg import MoveBaseAction, MoveBaseGoal

class NavigationLibrary(object):
    def __init__(self):
        rospy.init_node("navigation", anonymous=True)

        r = rospkg.RosPack()
        self.file_name = f"{r.get_path('your_package')}/csv/location.csv"

        self.move_base = actionlib.SimpleActionClient("move_base", MoveBaseAction)
        rospy.loginfo("=== Wait for movebase action ===")
        self.move_base.wait_for_server()
        rospy.loginfo("=== Connected movebase action server ===")

        self.nav_to_loc_ser = rospy.Service("/nav/nav_to_location", NavToLocation, self.nav_to_loc_callback)
        self.save_location_ser = rospy.Service("/nav/save_location", SaveLocation, self.save_location_callback)
```

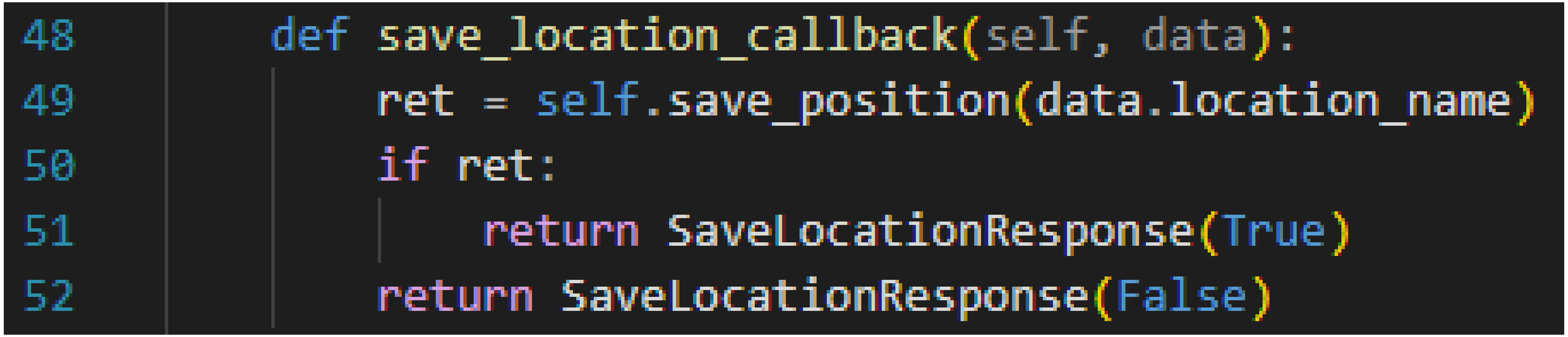
robotcitizens.org

Navigation in Python3



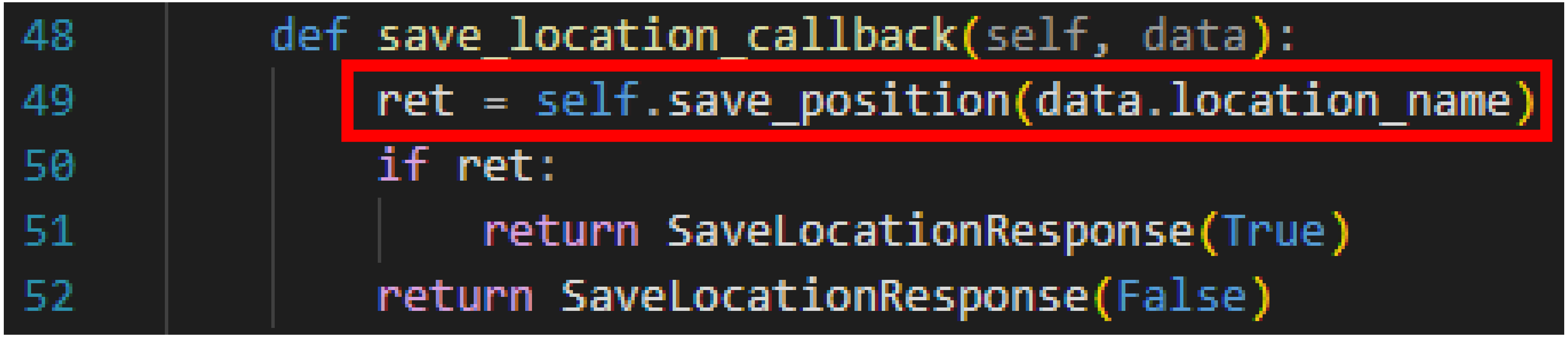
```
#!/usr/bin/env python3

import csv
import rospy
import rospkg
import actionlib
import tf2_ros
import tf
from tf.transformations import quaternion_from_euler, euler_from_quaternion

from actionlib_msgs.msg import *
from your_package.srv import *
from geometry_msgs.msg import Pose, Point, Quaternion
from move_base_msgs.msg import MoveBaseAction, MoveBaseGoal

class NavigationLibrary(object):
    def __init__(self):
        rospy.init_node("navigation", anonymous=True)

        r = rospkg.RosPack()
        self.file_name = f"{r.get_path('your_package')}/csv/location.csv"

        self.move_base = actionlib.SimpleActionClient("move_base", MoveBaseAction)
        rospy.loginfo("=== Wait for movebase action ===")
        self.move_base.wait_for_server()
        rospy.loginfo("=== Connected movebase action server ===")

        self.save_location_ser = rospy.Service("/nav/save_location", SaveLocation, self.save_location_callback)
```

Navigation in Python3



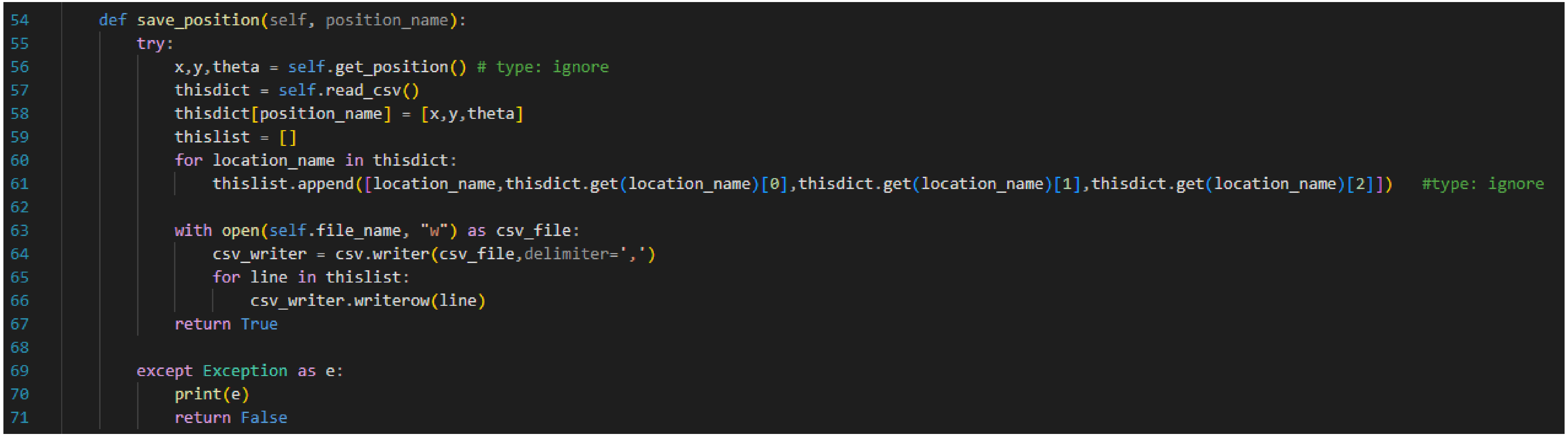
```python
    def save_location_callback(self, data):
        ret = self.save_position(data.location_name)
        if ret:
            return SaveLocationResponse(True)
        return SaveLocationResponse(False)
```

Navigation in Python3



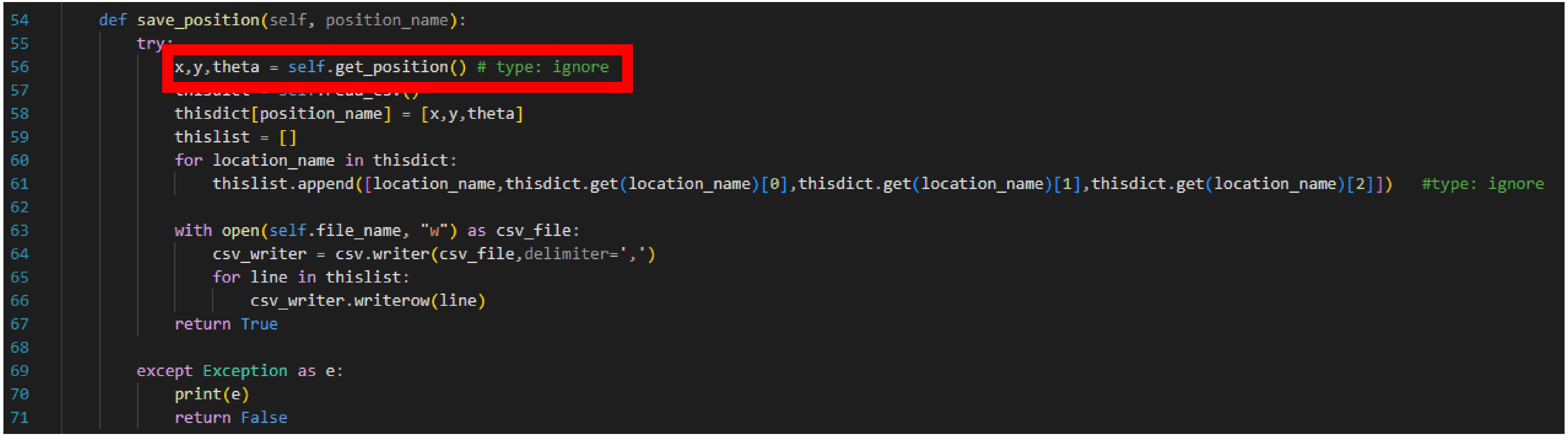
```python
    def save_location_callback(self, data):
        ret = self.save_position(data.location_name)
        if ret:
            return SaveLocationResponse(True)
        return SaveLocationResponse(False)
```

Navigation in Python3



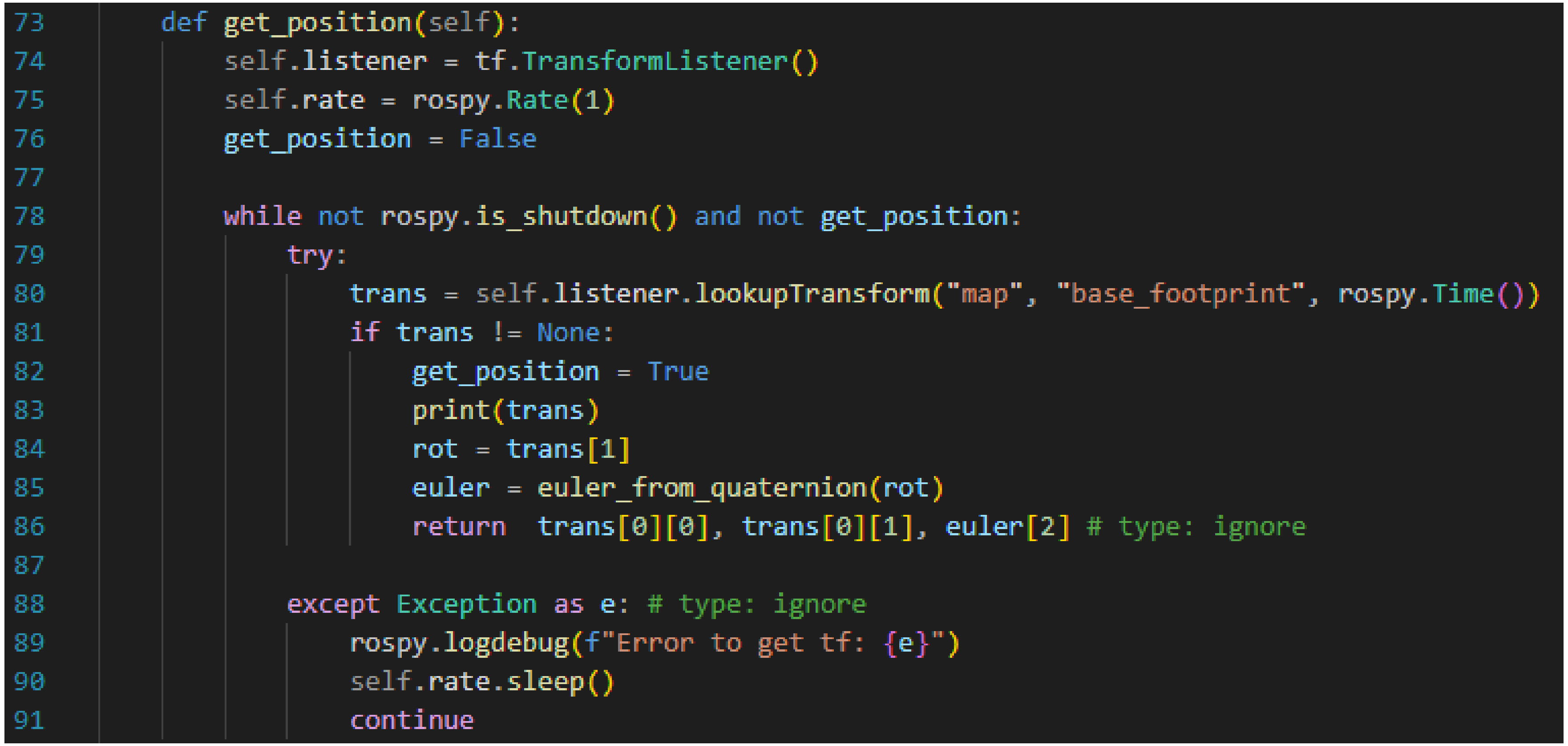
```python
    def save_position(self, position_name):
        try:
            x,y,theta = self.get_position() # type: ignore
            thisdict = self.read_csv()
            thisdict[position_name] = [x,y,theta]
            thislist = []
            for location_name in thisdict:
                thislist.append([location_name,thisdict.get(location_name)[0],thisdict.get(location_name)[1],thisdict.get(location_name)[2]])   #type: ignore

            with open(self.file_name, "w") as csv_file:
                csv_writer = csv.writer(csv_file,delimiter=',')
                for line in thislist:
                    csv_writer.writerow(line)
            return True

        except Exception as e:
            print(e)
            return False
```

Navigation in Python3



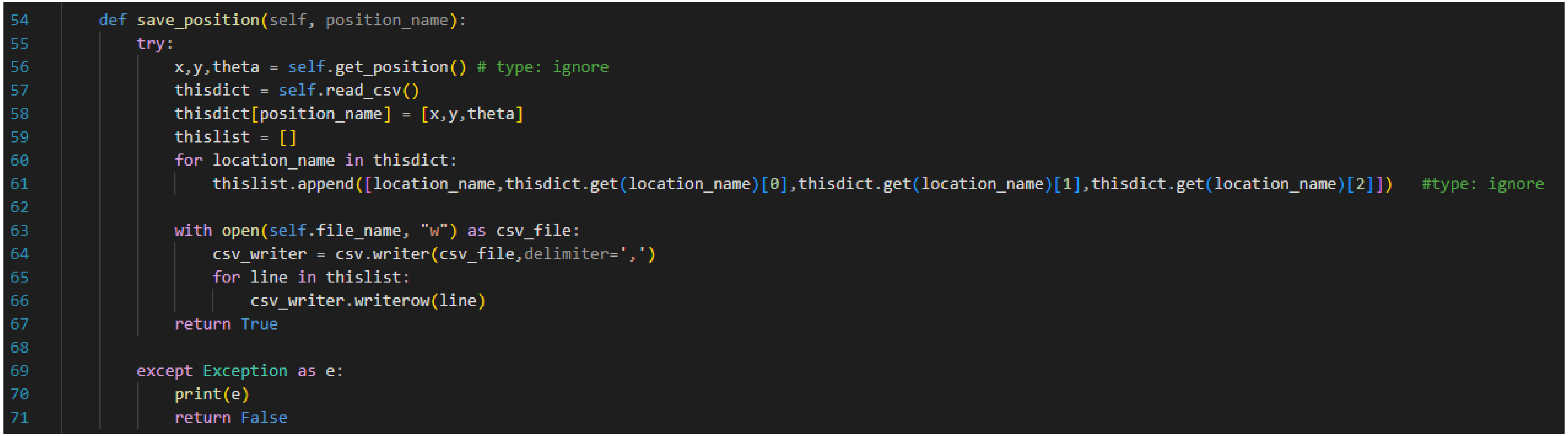
```python
    def save_position(self, position_name):
        try:
            x,y,theta = self.get_position() # type: ignore
            thisdict = self.read_csv()
            thisdict[position_name] = [x,y,theta]
            thislist = []
            for location_name in thisdict:
                thislist.append([location_name,thisdict.get(location_name)[0],thisdict.get(location_name)[1],thisdict.get(location_name)[2]])   #type: ignore

            with open(self.file_name, "w") as csv_file:
                csv_writer = csv.writer(csv_file,delimiter=',')
                for line in thislist:
                    csv_writer.writerow(line)
            return True

        except Exception as e:
            print(e)
            return False
```

Navigation in Python3



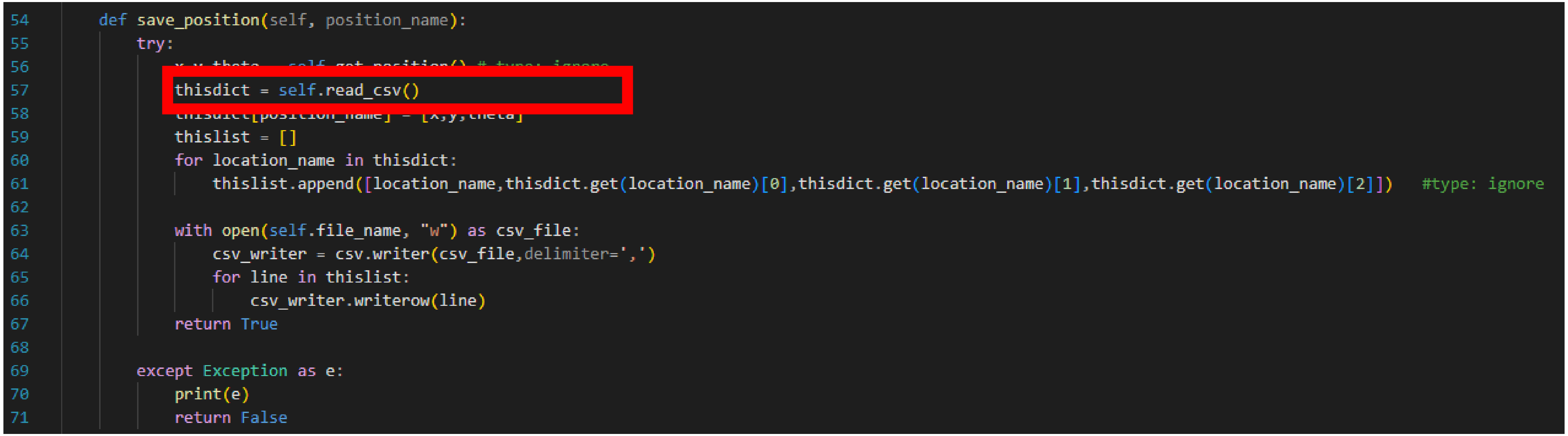
```python
    def get_position(self):
        self.listener = tf.TransformListener()
        self.rate = rospy.Rate(1)
        get_position = False

        while not rospy.is_shutdown() and not get_position:
            try:
                trans = self.listener.lookupTransform("map", "base_footprint", rospy.Time())
                if trans != None:
                    get_position = True
                    print(trans)
                    rot = trans[1]
                    euler = euler_from_quaternion(rot)
                    return  trans[0][0], trans[0][1], euler[2] # type: ignore

            except Exception as e: # type: ignore
                rospy.logdebug(f"Error to get tf: {e}")
                self.rate.sleep()
                continue
```

Navigation in Python3



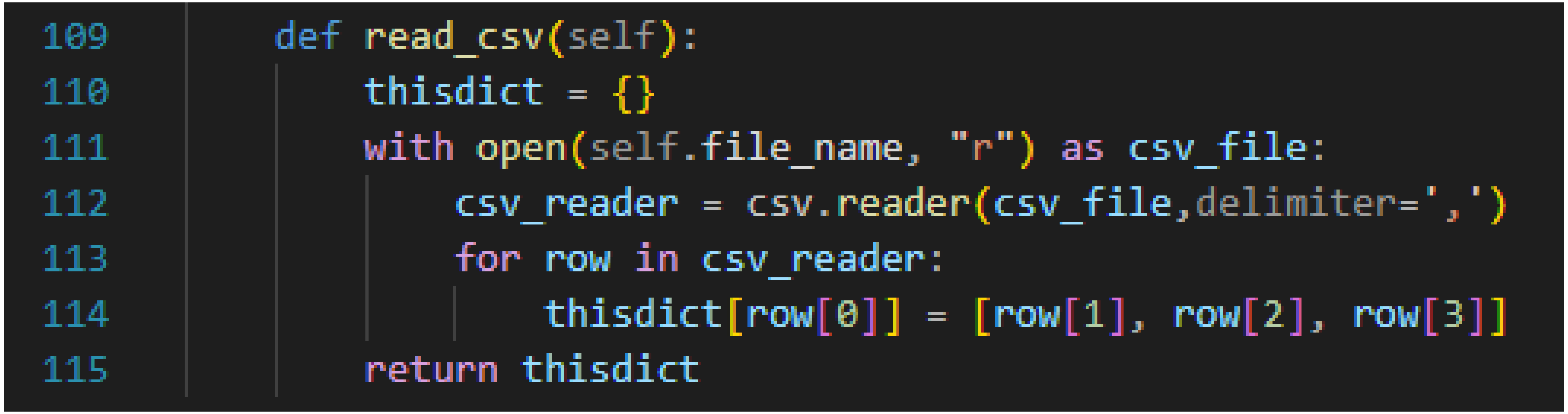
```python
    def save_position(self, position_name):
        try:
            x,y,theta = self.get_position() # type: ignore
            thisdict = self.read_csv()
            thisdict[position_name] = [x,y,theta]
            thislist = []
            for location_name in thisdict:
                thislist.append([location_name,thisdict.get(location_name)[0],thisdict.get(location_name)[1],thisdict.get(location_name)[2]])   #type: ignore

            with open(self.file_name, "w") as csv_file:
                csv_writer = csv.writer(csv_file,delimiter=',')
                for line in thislist:
                    csv_writer.writerow(line)
            return True

        except Exception as e:
            print(e)
            return False
```

Navigation in Python3



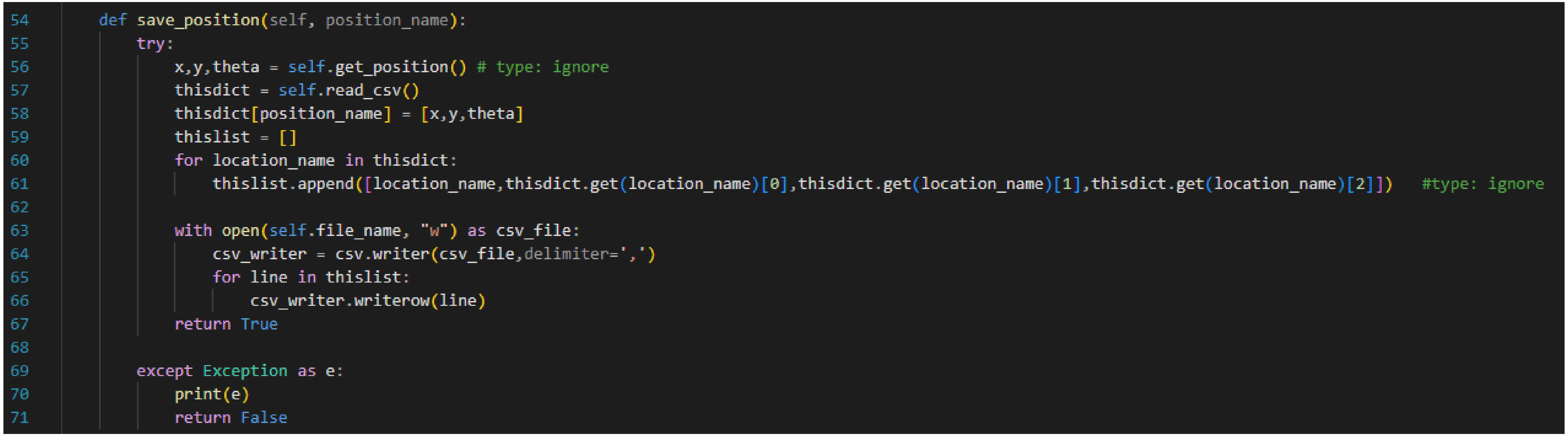
```
    def save_position(self, position_name):
        try:
            x,y,theta = self.get_position() # type: ignore
            thisdict = self.read_csv()
            thisdict[position_name] = [x,y,theta]
            thislist = []
            for location_name in thisdict:
                thislist.append([location_name,thisdict.get(location_name)[0],thisdict.get(location_name)[1],thisdict.get(location_name)[2]])   #type: ignore

            with open(self.file_name, "w") as csv_file:
                csv_writer = csv.writer(csv_file,delimiter=',')
                for line in thislist:
                    csv_writer.writerow(line)
            return True

        except Exception as e:
            print(e)
            return False
```

Navigation in Python3



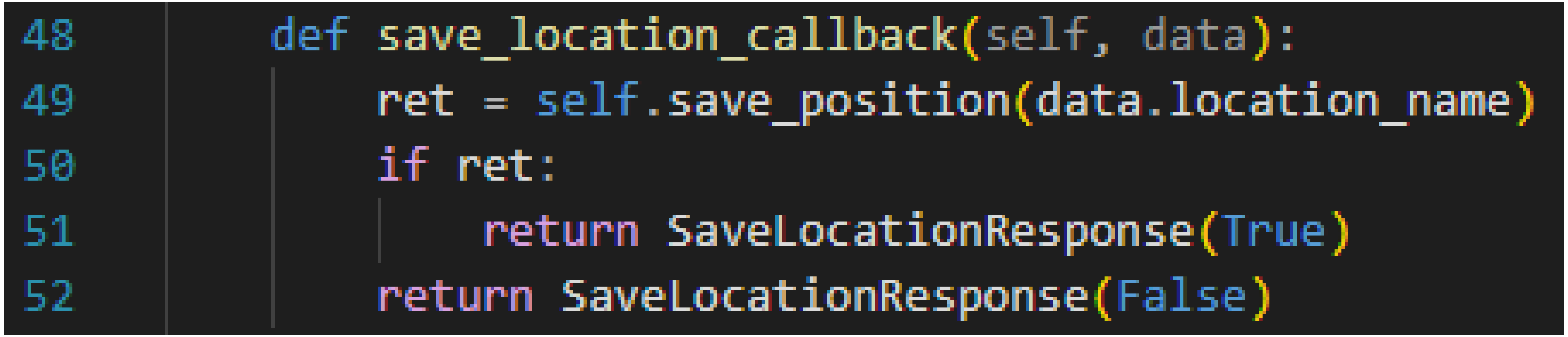
```
    def read_csv(self):
        thisdict = {}
        with open(self.file_name, "r") as csv_file:
            csv_reader = csv.reader(csv_file,delimiter=',')
            for row in csv_reader:
                thisdict[row[0]] = [row[1], row[2], row[3]]
        return thisdict
```

Navigation in Python3



```python
    def save_position(self, position_name):
        try:
            x,y,theta = self.get_position() # type: ignore
            thisdict = self.read_csv()
            thisdict[position_name] = [x,y,theta]
            thislist = []
            for location_name in thisdict:
                thislist.append([location_name,thisdict.get(location_name)[0],thisdict.get(location_name)[1],thisdict.get(location_name)[2]])   #type: ignore

            with open(self.file_name, "w") as csv_file:
                csv_writer = csv.writer(csv_file,delimiter=',')
                for line in thislist:
                    csv_writer.writerow(line)
            return True

        except Exception as e:
            print(e)
            return False
```

Navigation in Python3



```python
    def save_location_callback(self, data):
        ret = self.save_position(data.location_name)
        if ret:
            return SaveLocationResponse(True)
        return SaveLocationResponse(False)
```

Navigation in Python3

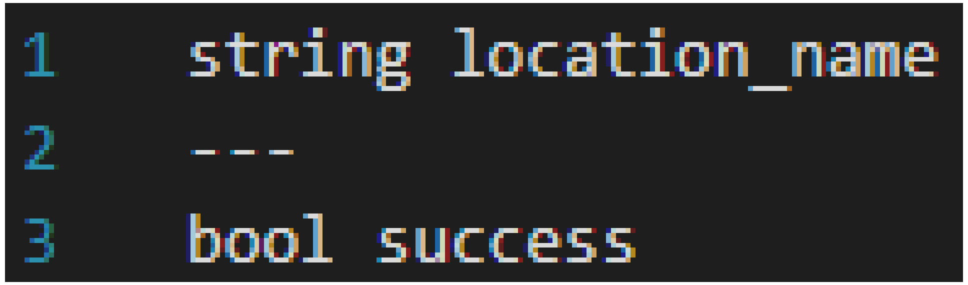
- Create directory srv

```
$ roscd your_package
```

```
$ mkdir srv
```

```
$ cd srv
```

Navigation in Python3



```
string location_name
---
bool success
```

- Add custom service NavToLocation.srv

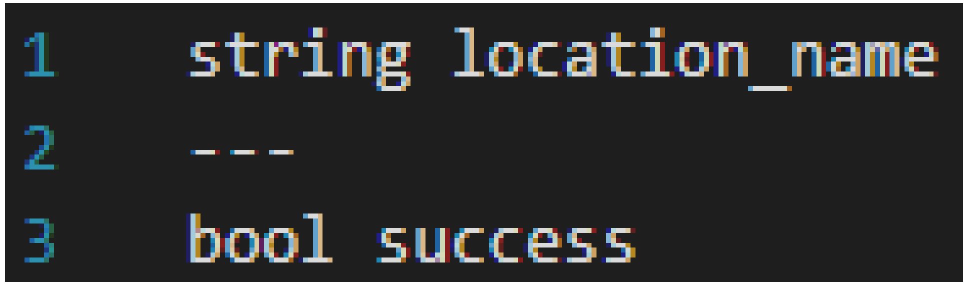
```
$ gedit NavToLocation.srv
```

- Copy code from link to NavToLocation.srv

NavToLocation.srv

- Save and close

Navigation in Python3



```
string location_name
---
bool success
```

- Add custom service SaveLocation.srv

```
$ gedit SaveLocation.srv
```

- Copy code from link to SaveLocation.srv

SaveLocation.srv

- Save and close

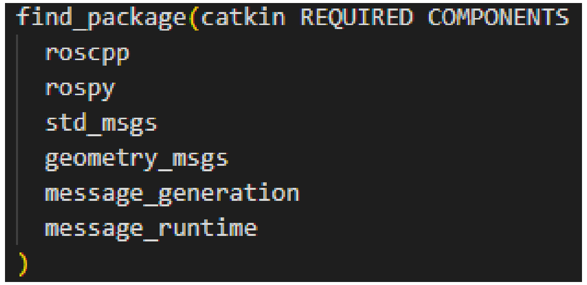
CMakeLists.txt

Navigation in Python3

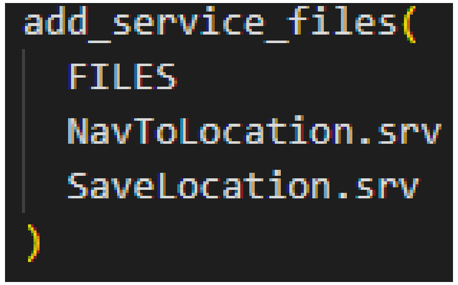
- Go to your_package and edit CMakeLists.txt

```
$ roscd your_package
```

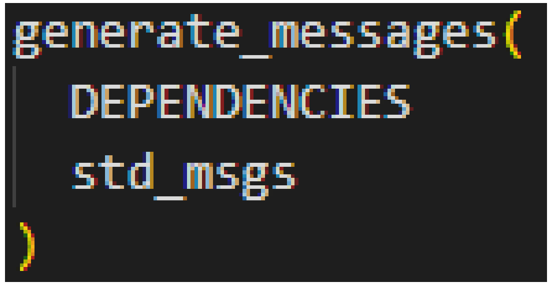
```
$ gedit CMakeLists.txt
```



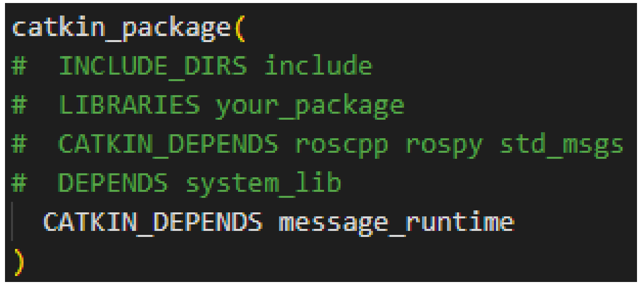
```
find_package(catkin REQUIRED COMPONENTS
  roscpp
  rospy
  std_msgs
  geometry_msgs
  message_generation
  message_runtime
)
```



```
add_service_files(
  FILES
  NavToLocation.srv
  SaveLocation.srv
)
```



```
generate_messages(
  DEPENDENCIES
  std_msgs
)
```



```
catkin_package(
#  INCLUDE_DIRS include
#  LIBRARIES your_package
#  CATKIN_DEPENDS roscpp rospy std_msgs
#  DEPENDS system_lib
  CATKIN_DEPENDS message_runtime
)
```

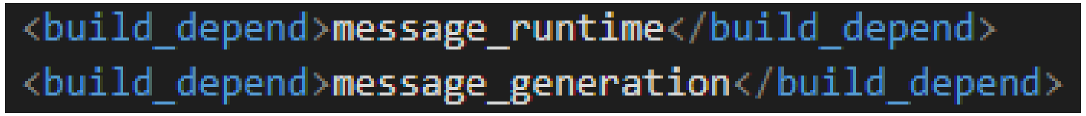
package.xml

Navigation in Python3

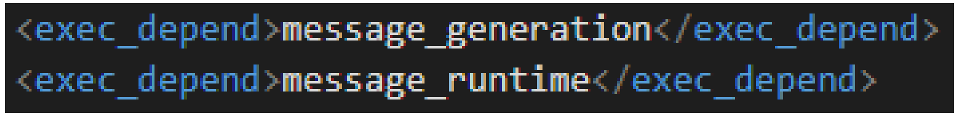
- Go to your_package and edit package.xml

```
$ roscd your_package
```

```
$ gedit package.xml
```



```
<build_depend>message_runtime</build_depend>
<build_depend>message_generation</build_depend>
```



```
<exec_depend>message_generation</exec_depend>
<exec_depend>message_runtime</exec_depend>
```

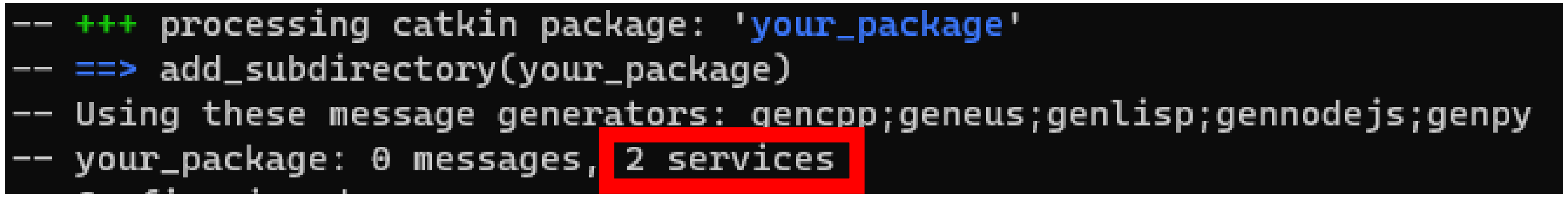
Navigation in Python3

- Build your workspace

```
$ roscd
```

```
$ cd ../
```

```
$ catkin_make
```



```
-- +++ processing catkin package: 'your_package'
-- ==> add_subdirectory(your_package)
-- Using these message generators: gencpp;geneus;genlisp;gennodejs;genpy
-- your_package: 0 messages, 2 services
```

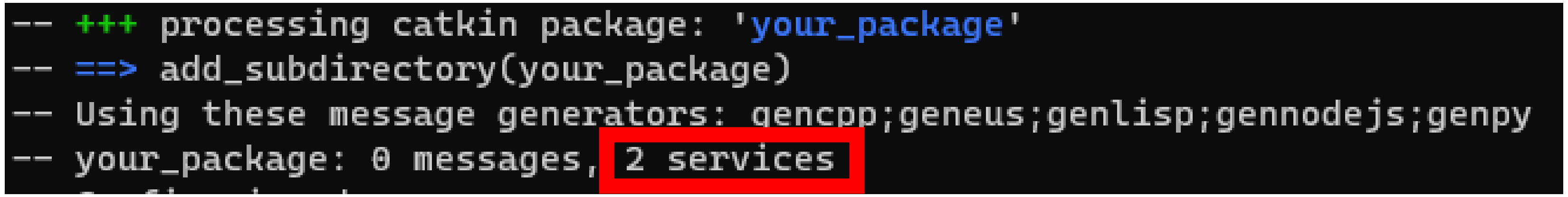
Navigation in Python3

- Build your workspace

```
$ roscd
```

```
$ cd ../
```

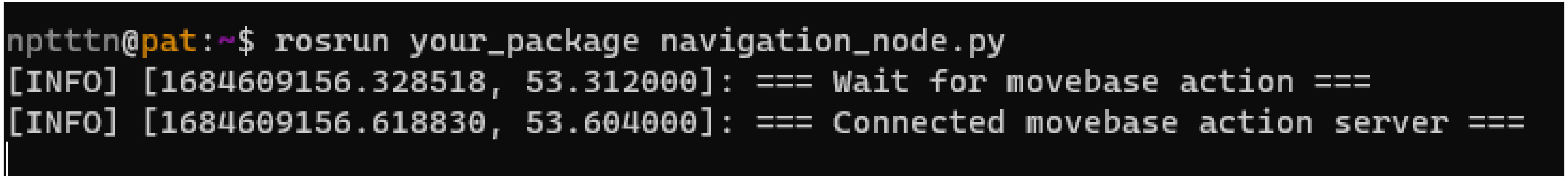
```
$ catkin_make
```



```
-- +++ processing catkin package: 'your_package'
-- ==> add_subdirectory(your_package)
-- Using these message generators: gencpp;geneus;genlisp;gennodejs;genpy
-- your_package: 0 messages, 2 services
```

Navigation in Python3

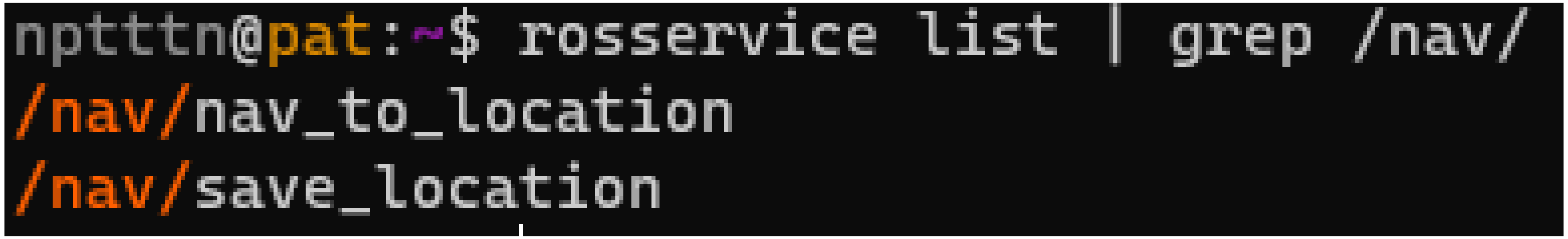
```
$ rosrun your_package navigation_node.py
```



```
nptttn@pat:~$ rosrun your_package navigation_node.py
[INFO] [1684609156.328518, 53.312000]: === Wait for movebase action ===
[INFO] [1684609156.618830, 53.604000]: === Connected movebase action server ===
```

Navigation in Python3

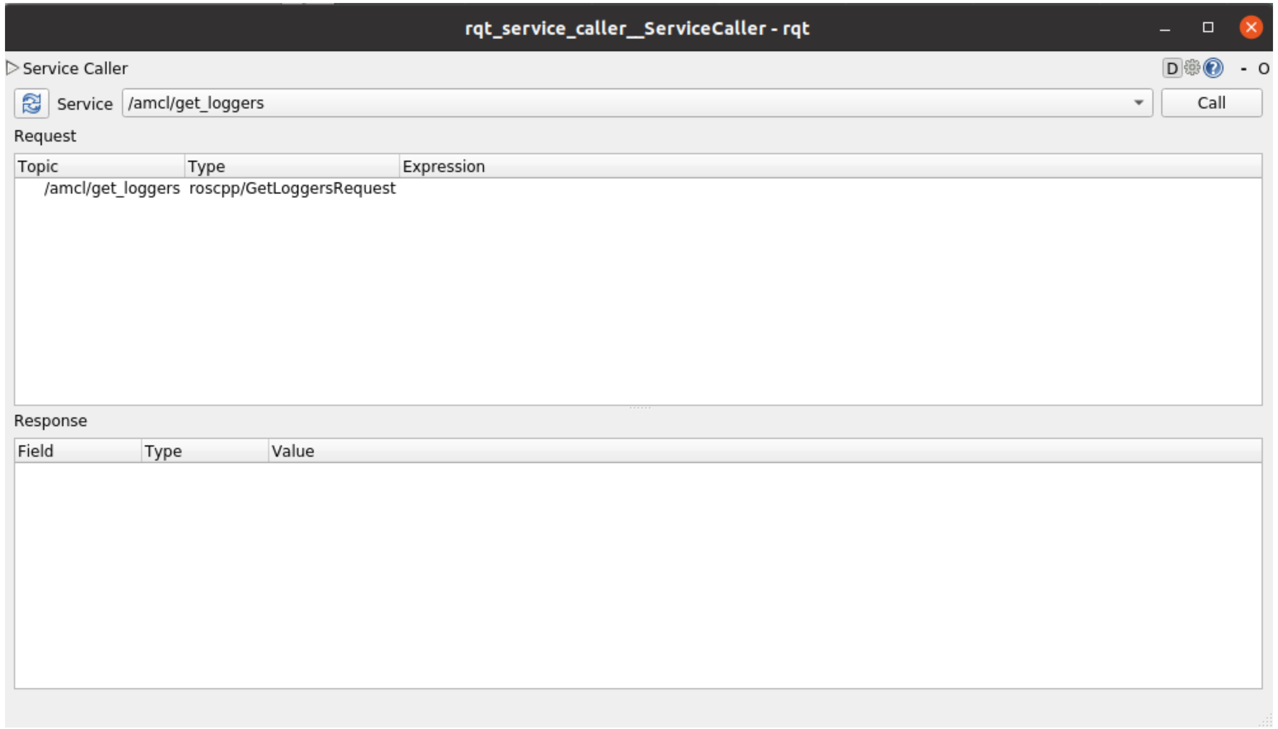
```
$ rosservice list | grep /nav/
```



```
nptttn@pat:~$ rosservice list | grep /nav/
/nav/nav_to_location
/nav/save_location
```

Navigation in Python3

```
$ rosrun rqt_service_caller rqt_service_caller
```



rqt_service_caller__ServiceCaller - rqt

Service Caller

Service /amcl/get_loggers

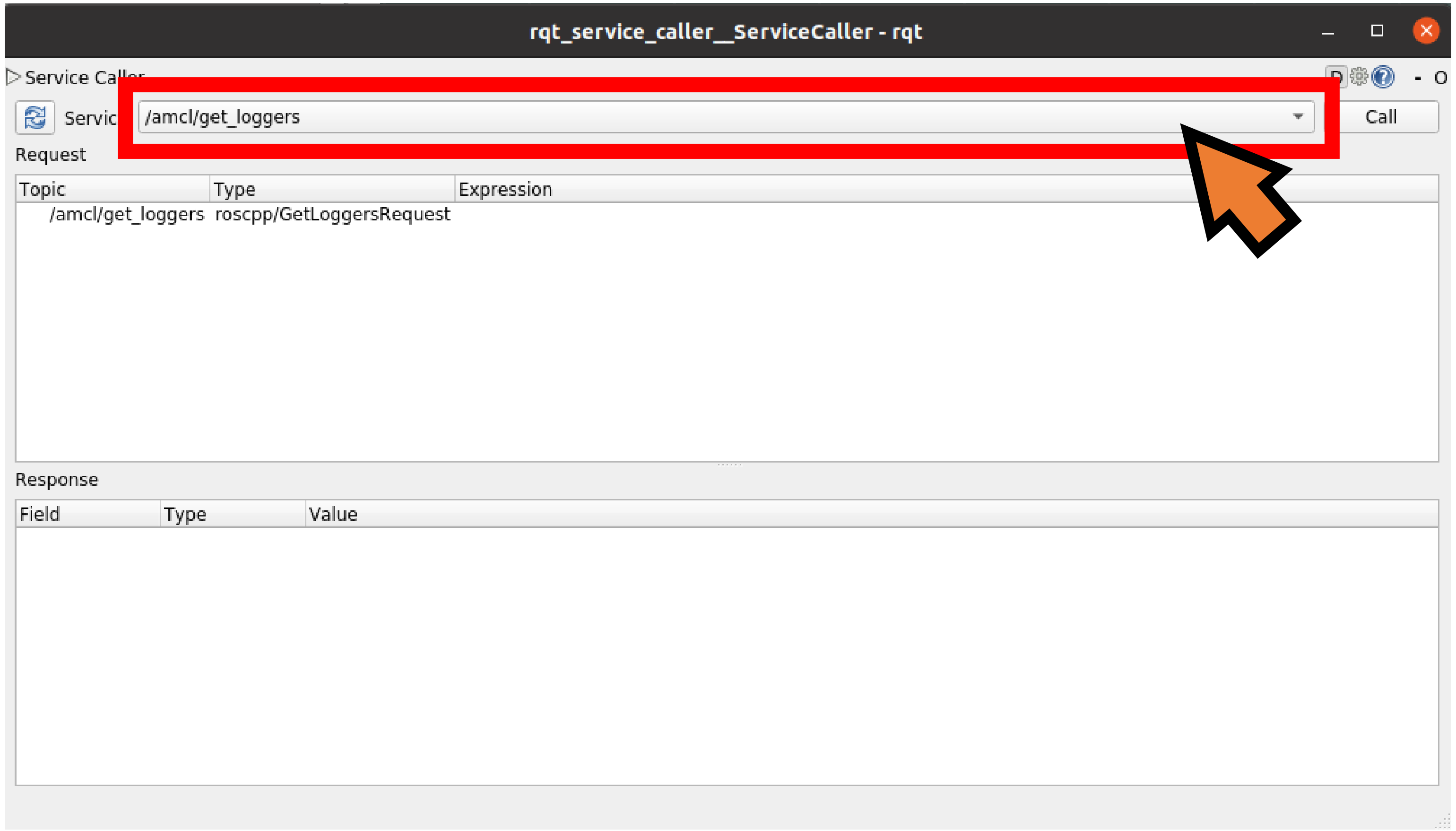
Request

Topic	Type	Expression
/amcl/get_loggers	roscpp/GetLoggersRequest	

Response

Field	Type	Value

Navigation in Python3



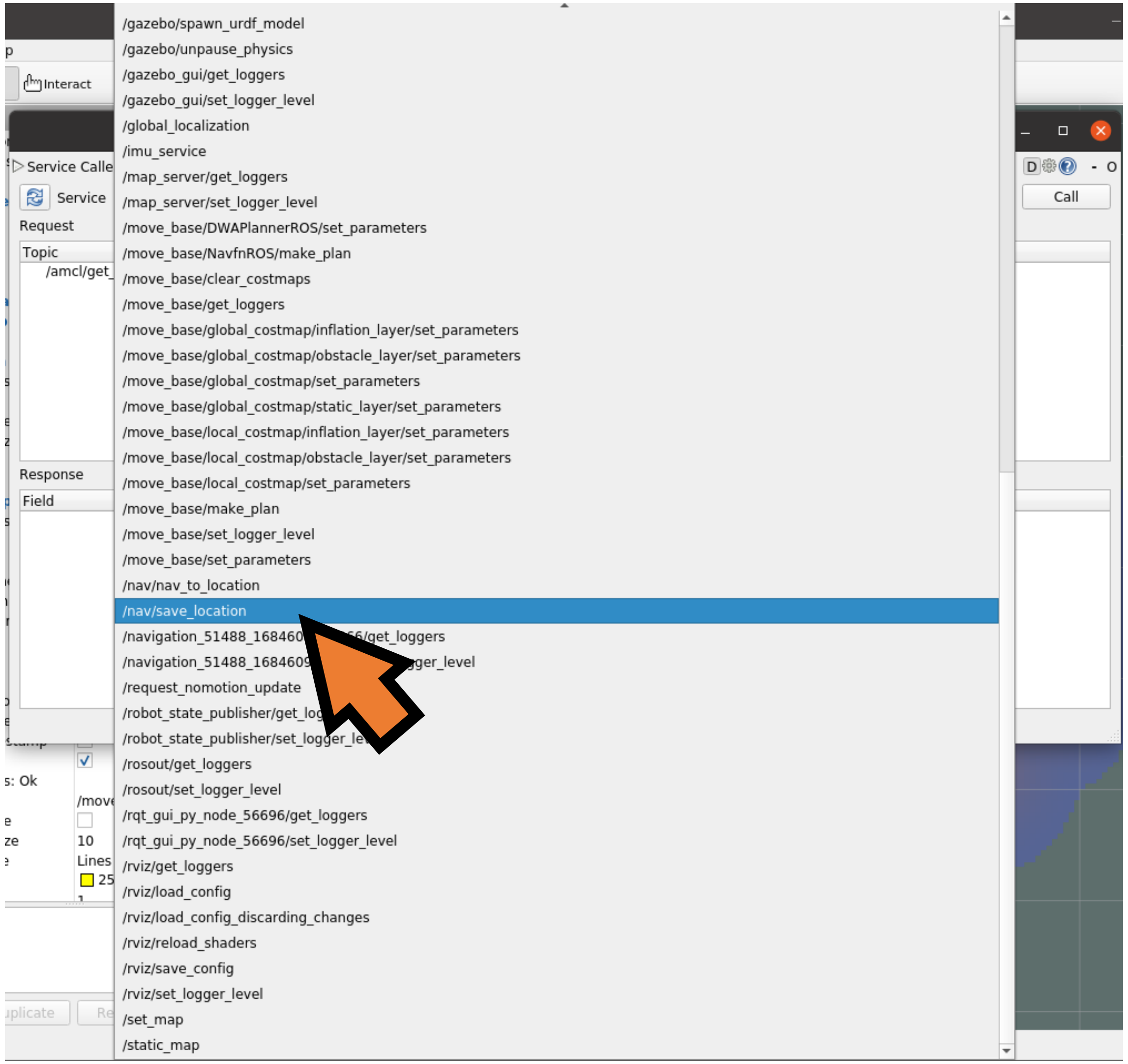
Service: /amcl/get_loggers

Topic	Type	Expression
/amcl/get_loggers	roscpp/GetLoggersRequest	

Field	Type	Value

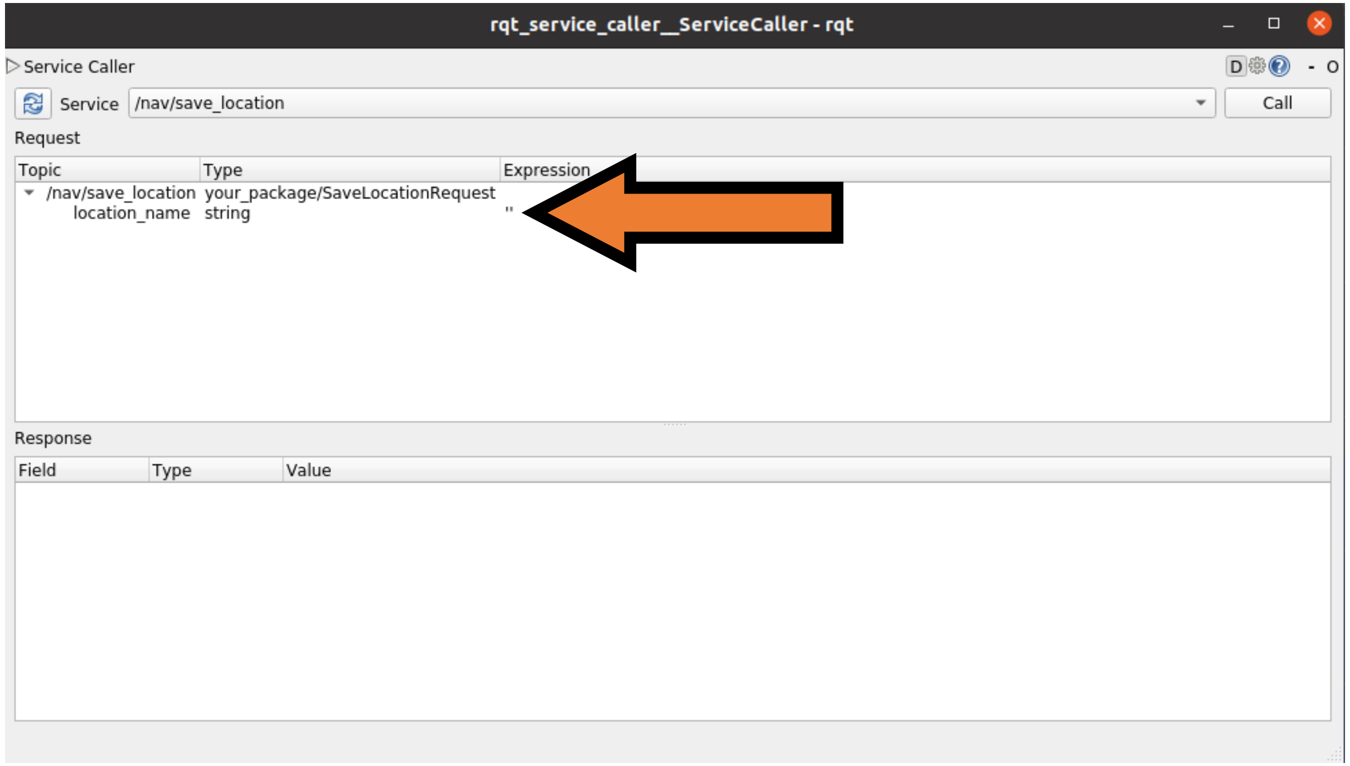
Navigation in Python3

- Find /nav/save_location
- Click the /nav/save_location



```
/gazebo/spawn_urdf_model
/gazebo/unpause_physics
/gazebo_gui/get_loggers
/gazebo_gui/set_logger_level
/global_localization
/imu_service
/map_server/get_loggers
/map_server/set_logger_level
/move_base/DWAPlannerROS/set_parameters
/move_base/NavfnROS/make_plan
/move_base/clear_costmaps
/move_base/get_loggers
/move_base/global_costmap/inflation_layer/set_parameters
/move_base/global_costmap/obstacle_layer/set_parameters
/move_base/global_costmap/set_parameters
/move_base/global_costmap/static_layer/set_parameters
/move_base/local_costmap/inflation_layer/set_parameters
/move_base/local_costmap/obstacle_layer/set_parameters
/move_base/local_costmap/set_parameters
/move_base/make_plan
/move_base/set_logger_level
/move_base/set_parameters
/nav/nav_to_location
/nav/save_location
/navigation_51488_168460   66/get_loggers
/navigation_51488_1684609   gger_level
/request_nomotion_update
/robot_state_publisher/get_log
/robot_state_publisher/set_logger_le
/rosout/get_loggers
/rosout/set_logger_level
/rqt_gui_py_node_56696/get_loggers
/rqt_gui_py_node_56696/set_logger_level
/rviz/get_loggers
/rviz/load_config
/rviz/load_config_discarding_changes
/rviz/reload_shaders
/rviz/save_config
/rviz/set_logger_level
/set_map
/static_map
```

Navigation in Python3



Service /nav/save_location

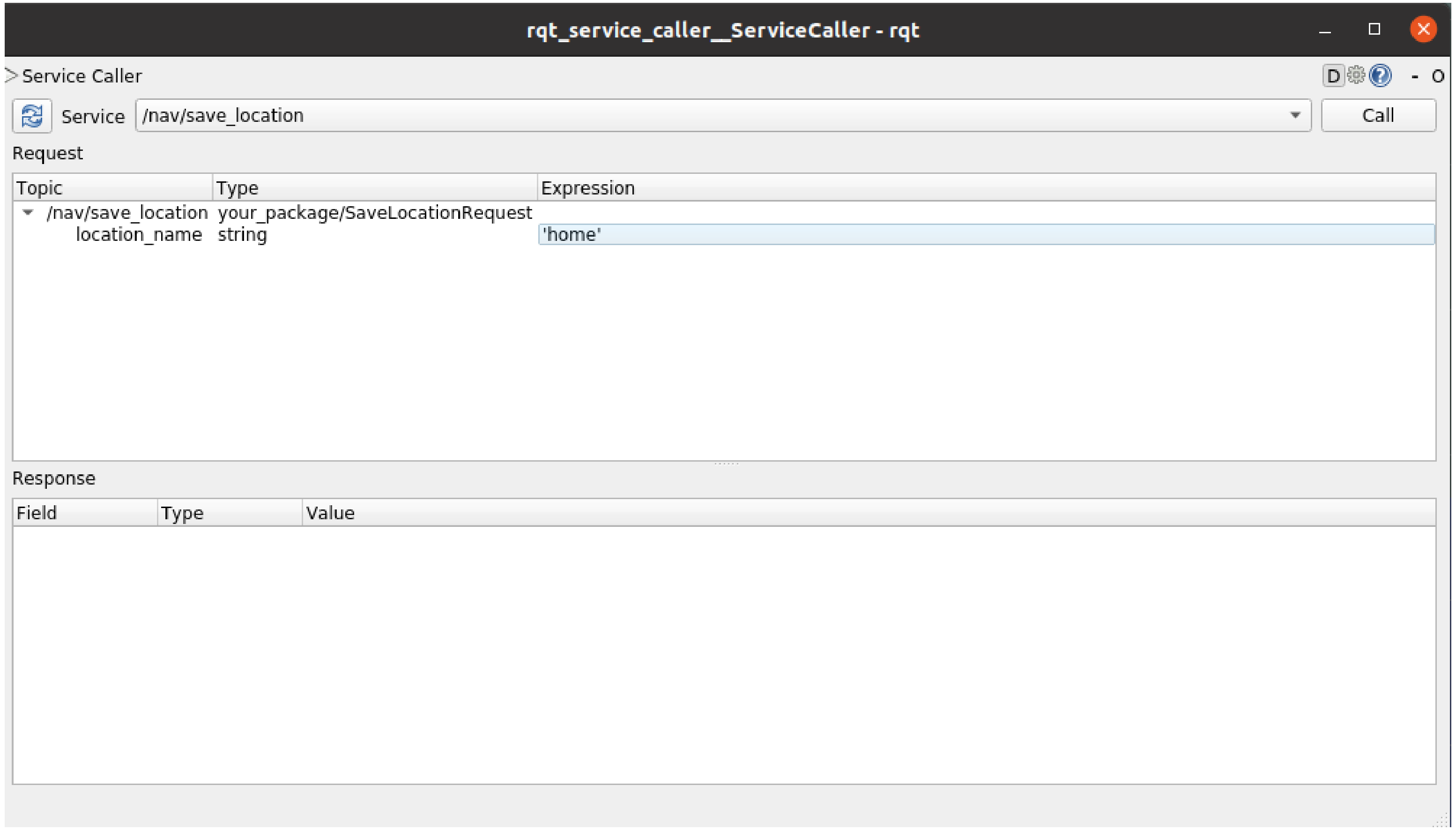
Request

Topic	Type	Expression
/nav/save_location	your_package/SaveLocationRequest	
location_name	string	''

Response

Field	Type	Value

Navigation in Python3

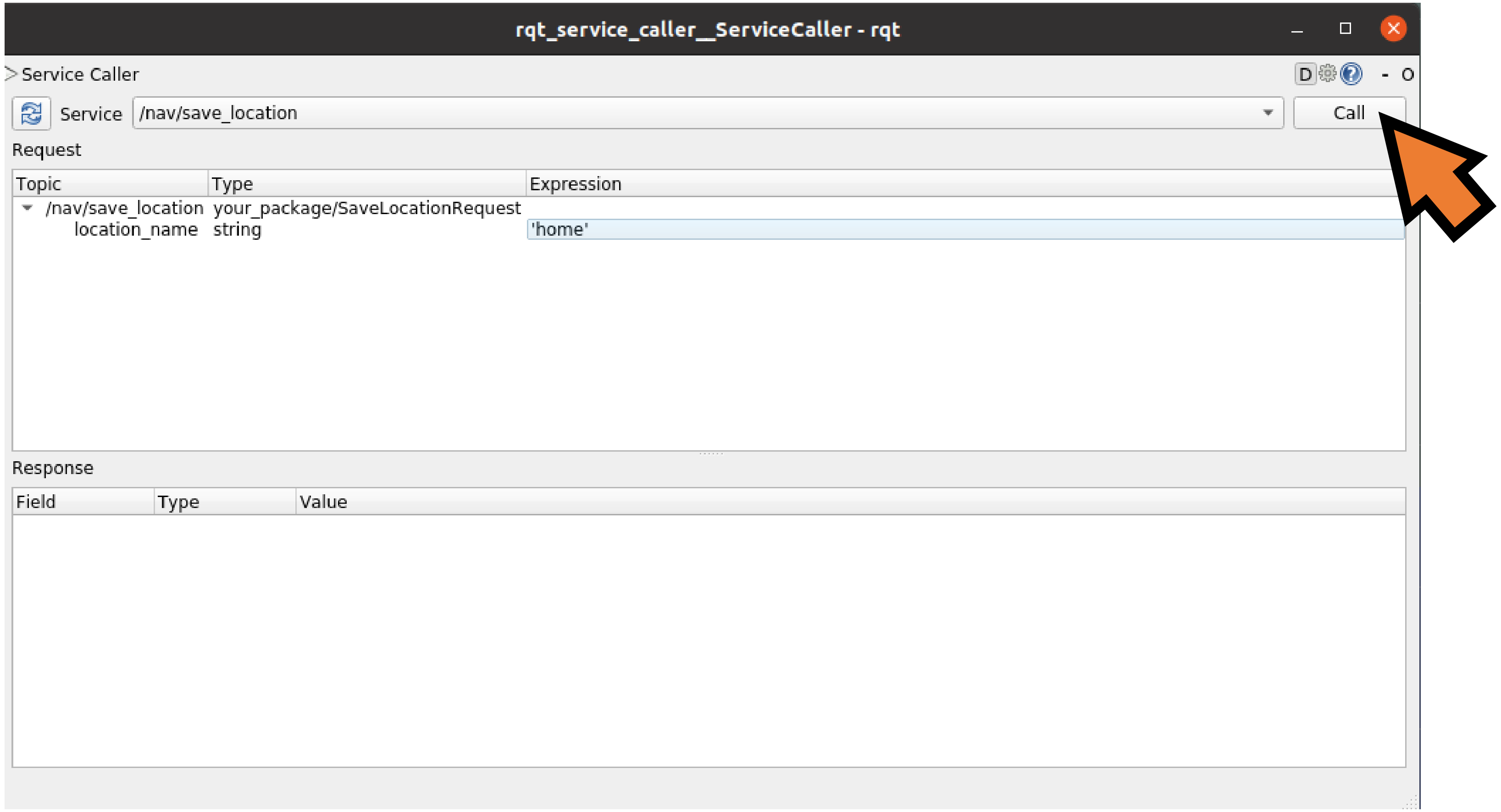


Service: /nav/save_location

Topic	Type	Expression
/nav/save_location	your_package/SaveLocationRequest	
location_name	string	'home'

Field	Type	Value

Navigation in Python3



Service /nav/save_location

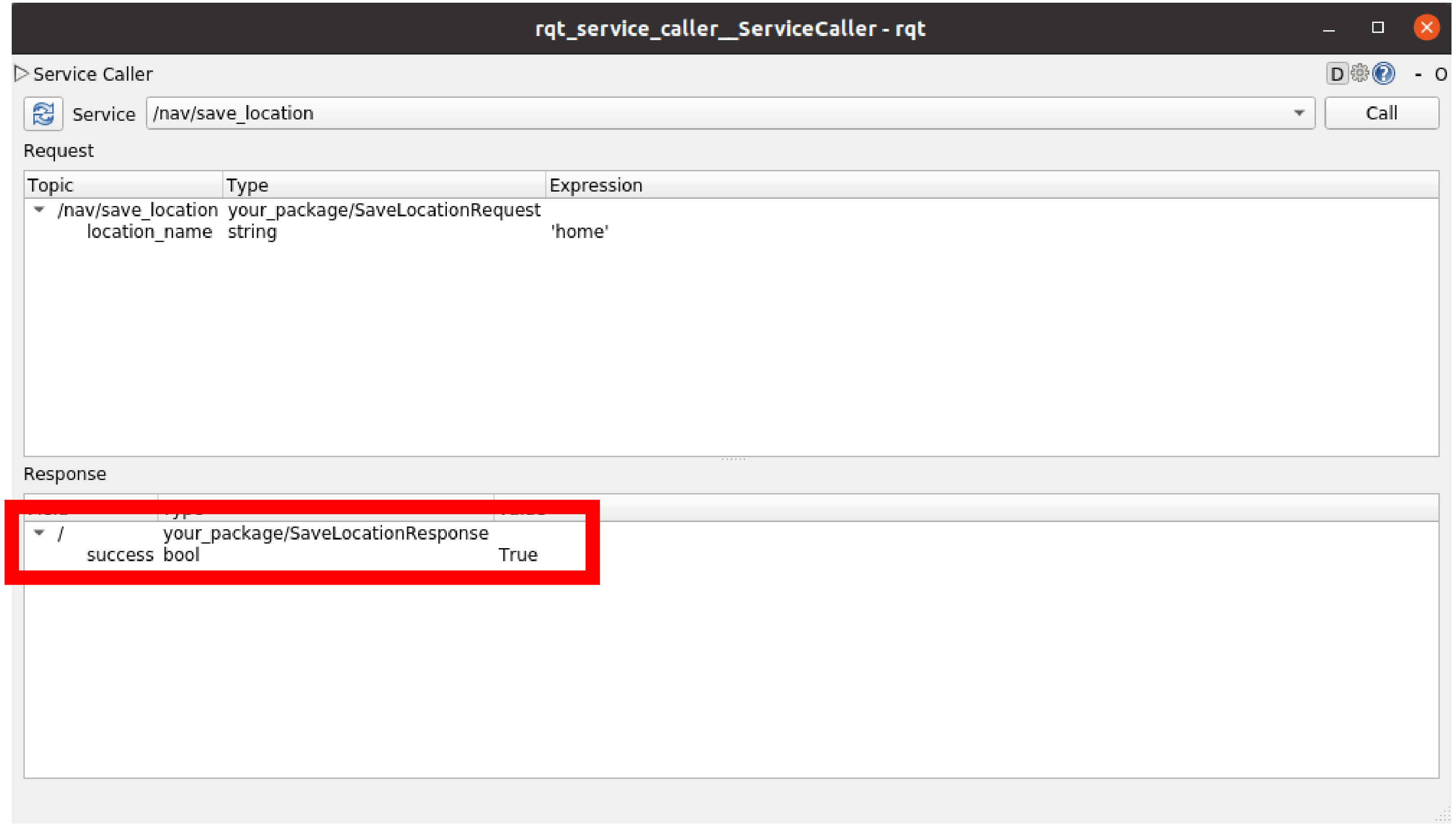
Request

Topic	Type	Expression
/nav/save_location	your_package/SaveLocationRequest	
location_name	string	'home'

Response

Field	Type	Value

Navigation in Python3



rqt_service_caller__ServiceCaller - rqt

Service Caller

Service /nav/save_location

Call

Request

Topic	Type	Expression
/nav/save_location	your_package/SaveLocationRequest	
location_name	string	'home'

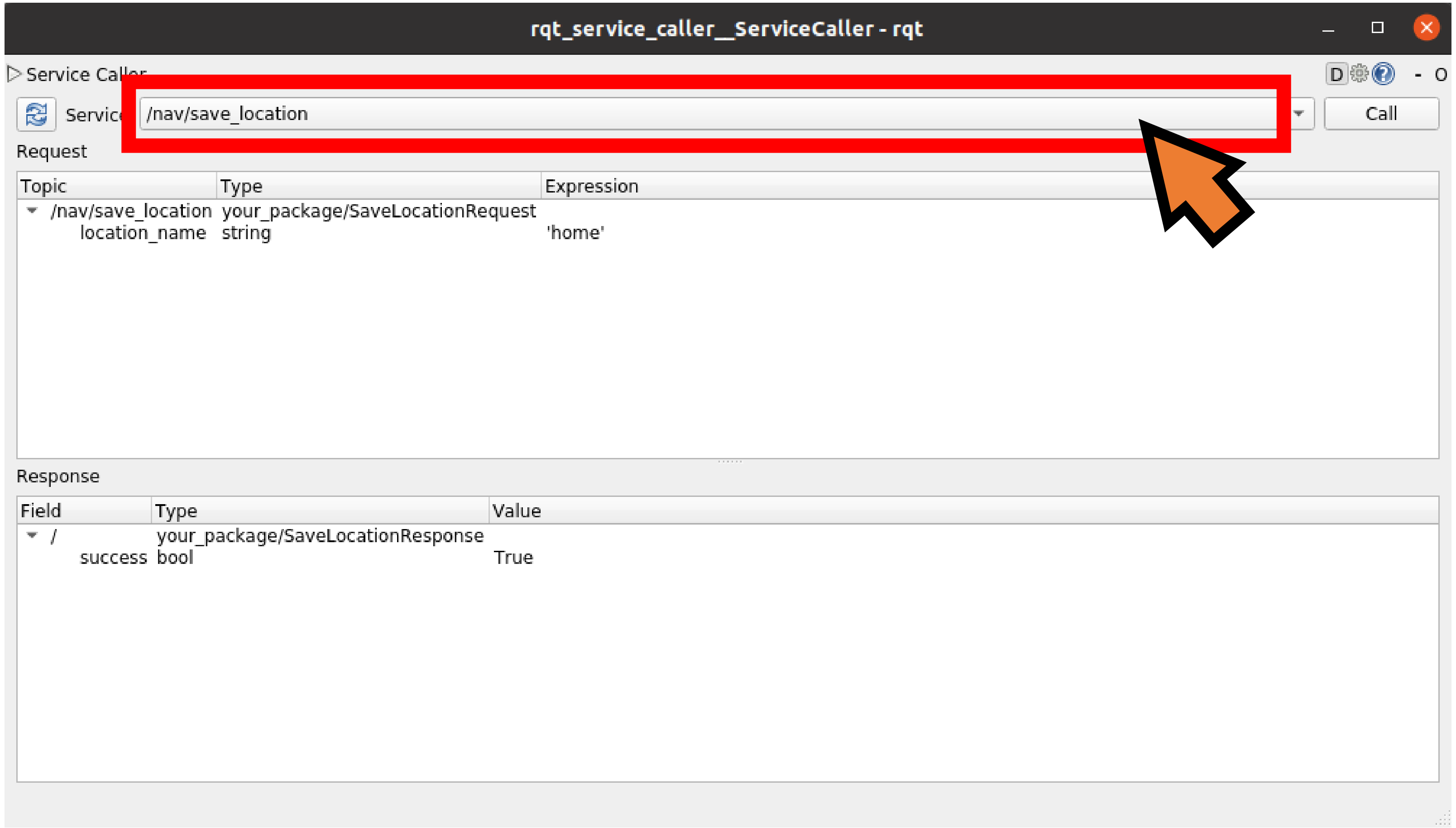
Response

/	your_package/SaveLocationResponse	
success	bool	True

Navigation in Python3

- Try to use keyboard teleop to control the robot to another position and save another position name **living_room**

Navigation in Python3



/nav/save_location

Request

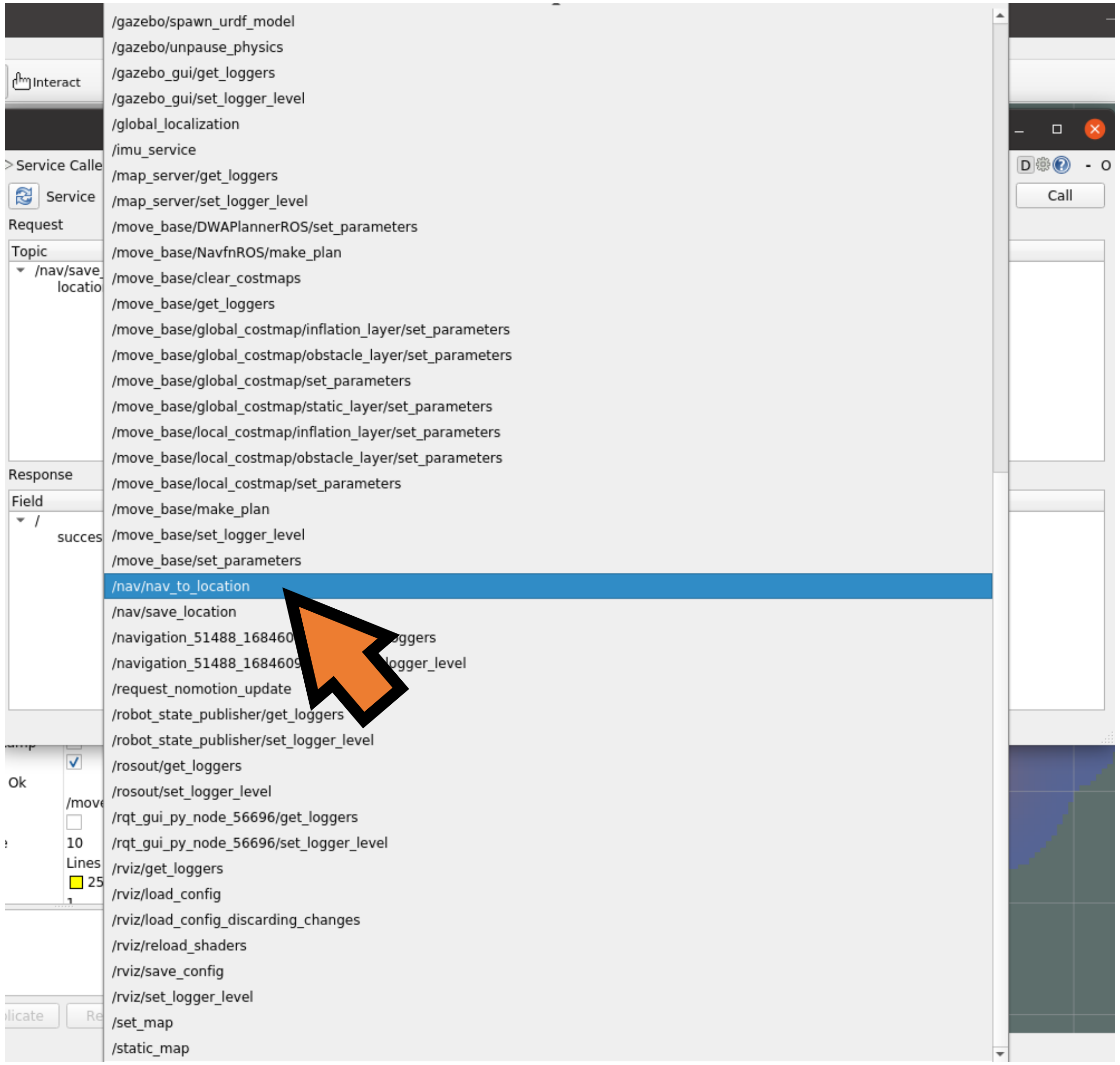
Topic	Type	Expression
/nav/save_location	your_package/SaveLocationRequest	
location_name	string	'home'

Response

Field	Type	Value
/	your_package/SaveLocationResponse	
success	bool	True

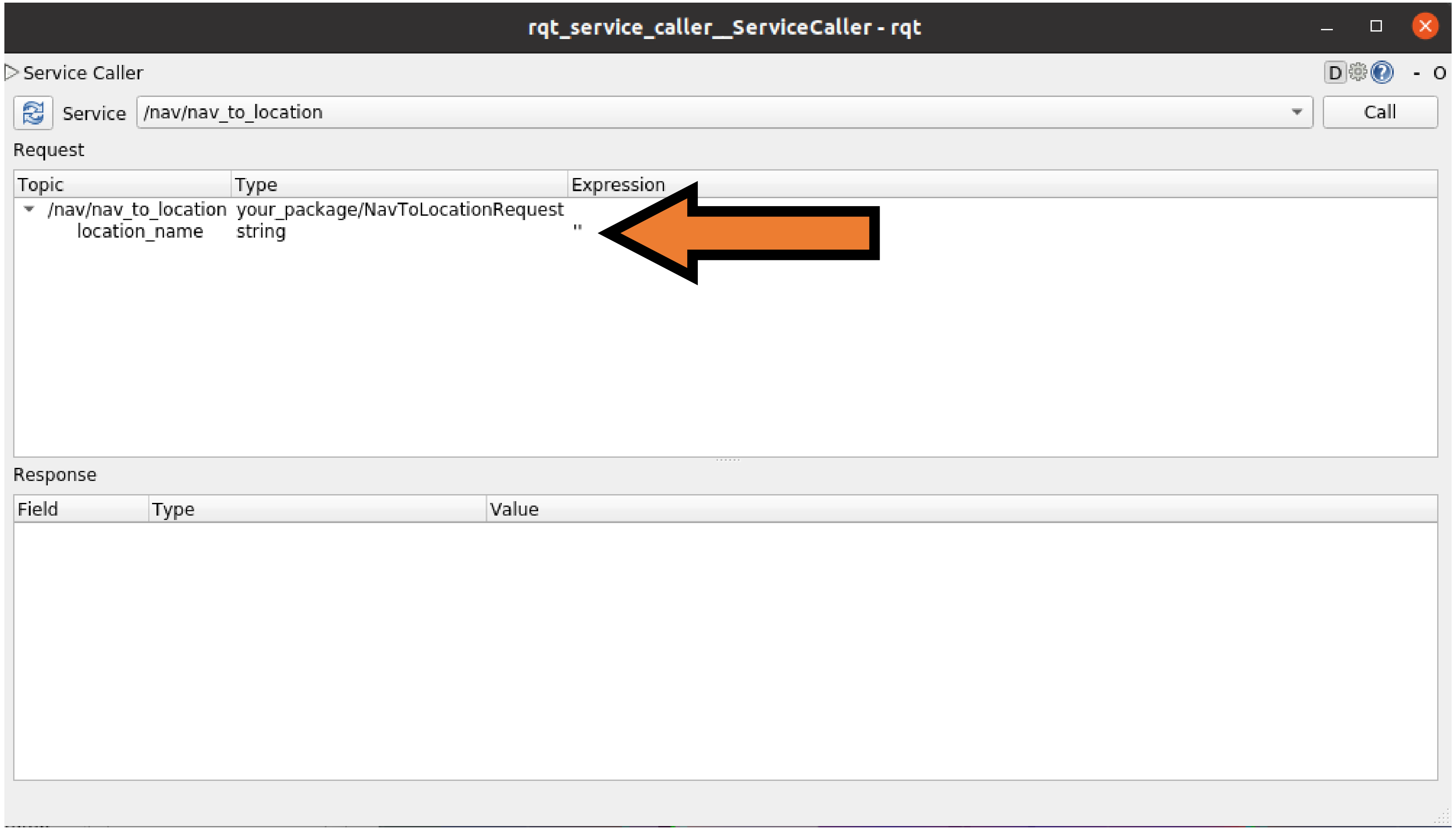
Navigation in Python3

- Find /nav/nav_to_location
- Click the /nav/nav_to_location



/gazebo/spawn_urdf_model
/gazebo/unpause_physics
/gazebo_gui/get_loggers
/gazebo_gui/set_logger_level
/global_localization
/imu_service
/map_server/get_loggers
/map_server/set_logger_level
/move_base/DWAPlannerROS/set_parameters
/move_base/NavfnROS/make_plan
/move_base/clear_costmaps
/move_base/get_loggers
/move_base/global_costmap/inflation_layer/set_parameters
/move_base/global_costmap/obstacle_layer/set_parameters
/move_base/global_costmap/set_parameters
/move_base/global_costmap/static_layer/set_parameters
/move_base/local_costmap/inflation_layer/set_parameters
/move_base/local_costmap/obstacle_layer/set_parameters
/move_base/local_costmap/set_parameters
/move_base/make_plan
/move_base/set_logger_level
/move_base/set_parameters
/nav/nav_to_location
/nav/save_location
/navigation_51488_168460 ggers
/navigation_51488_1684609 ogger_level
/request_nomotion_update
/robot_state_publisher/get_loggers
/robot_state_publisher/set_logger_level
/rosout/get_loggers
/rosout/set_logger_level
/rqt_gui_py_node_56696/get_loggers
/rqt_gui_py_node_56696/set_logger_level
/rviz/get_loggers
/rviz/load_config
/rviz/load_config_discarding_changes
/rviz/reload_shaders
/rviz/save_config
/rviz/set_logger_level
/set_map
/static_map

Navigation in Python3



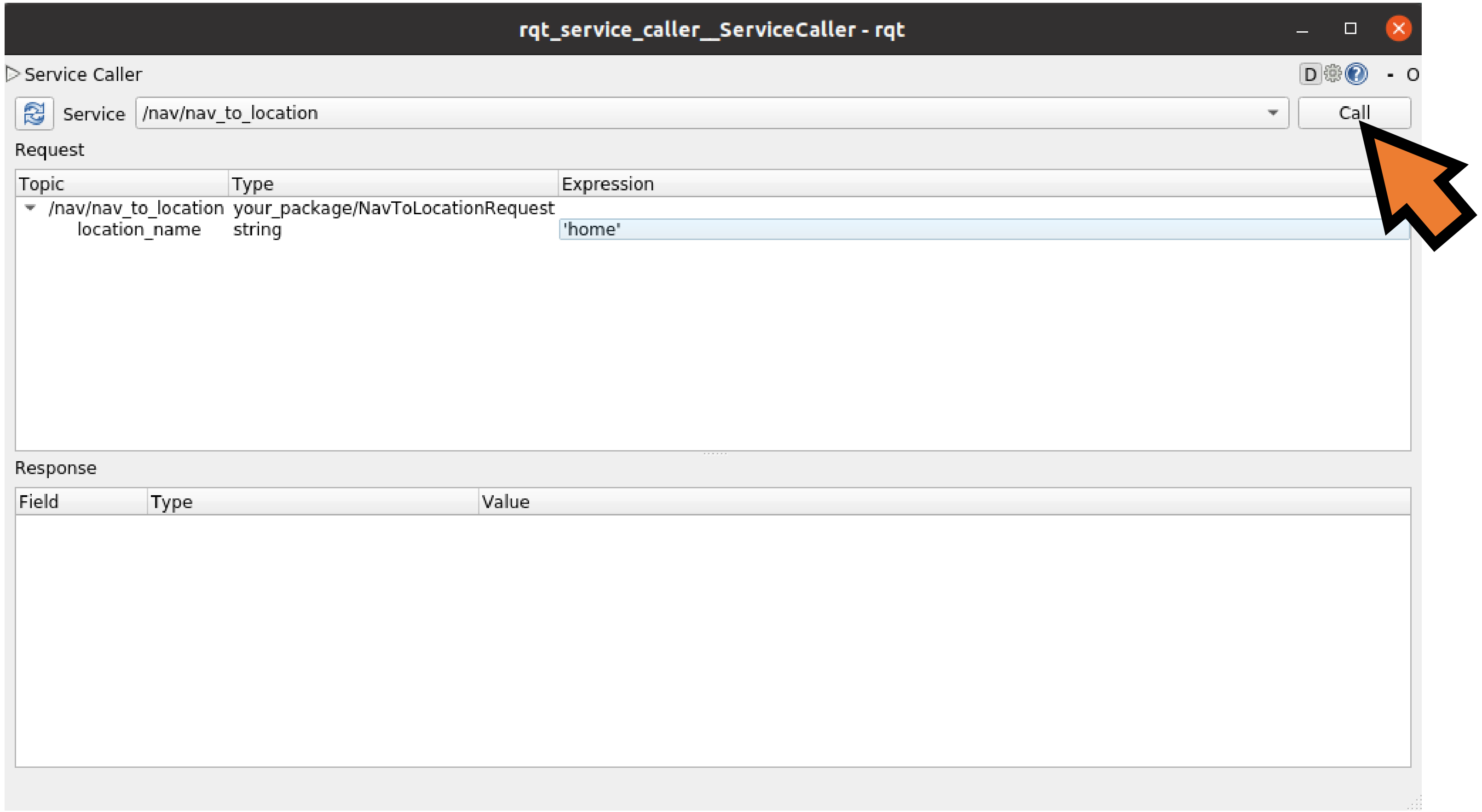
Service: /nav/nav_to_location

Topic	Type	Expression
/nav/nav_to_location	your_package/NavToLocationRequest	
location_name	string	''

Field	Type	Value

robotcitizens.org

Navigation in Python3



rqt_service_caller__ServiceCaller - rqt

Service Caller

Service /nav/nav_to_location

Request

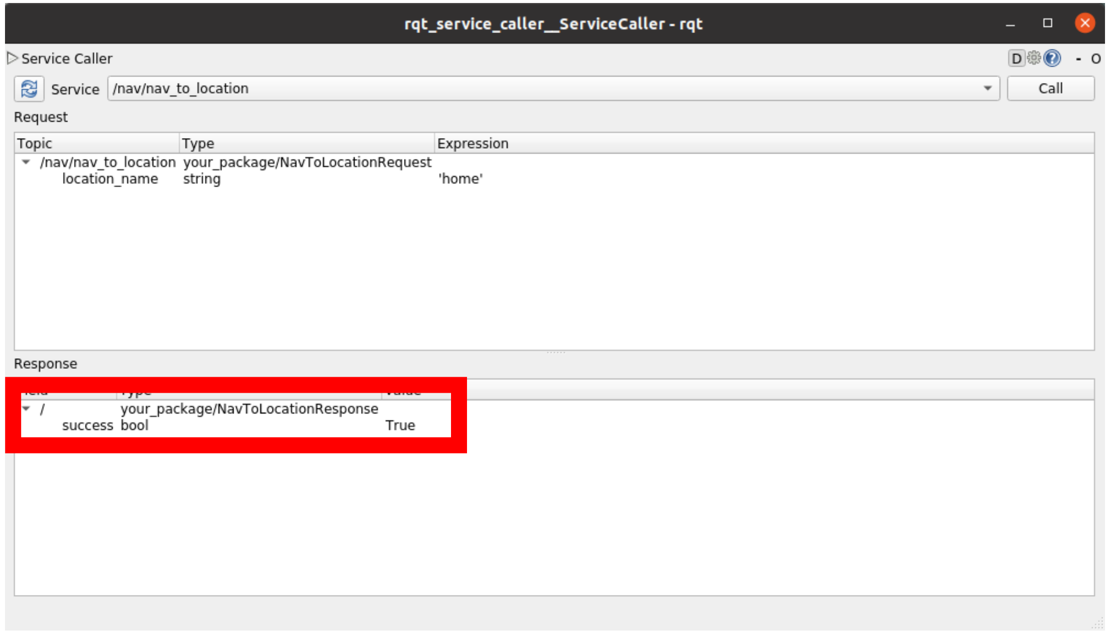
Topic	Type	Expression
/nav/nav_to_location	your_package/NavToLocationRequest	
location_name	string	'home'

Response

Field	Type	Value

Navigation in Python3

- See the robot in Gazebo and Rviz
- After robot reach the goal, the response will show as True



Service /nav/nav_to_location

Request

Topic	Type	Expression
/nav/nav_to_location	your_package/NavToLocationRequest	
location_name	string	'home'

Response

Field	Type	Value
/	your_package/NavToLocationResponse	
success	bool	True

Navigation with Turtlebot2

Using real Turtlebot2 with navigation.

Turtlebot2 – Create map

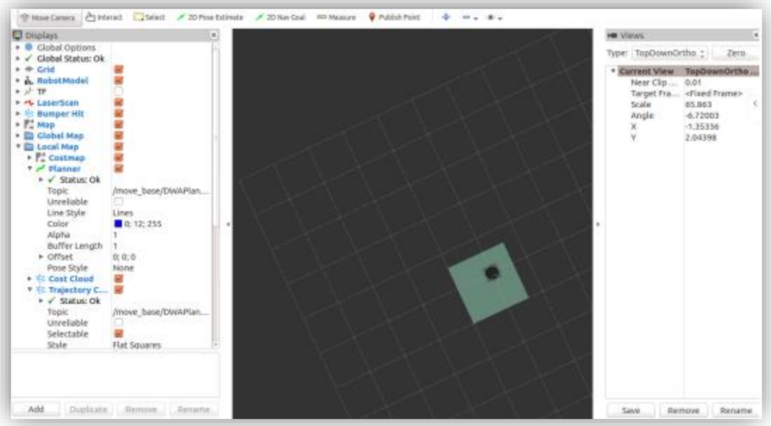
- Bring up your turtlebot2

```
$ roslaunch turtlebot_bringup minimal.launch
```

- Run package for SLAM

```
$ roslaunch turtlebot_navigation gmapping_demo.launch
```

Turtlebot2 - Create map



Global Options
Global Status: Ok
Grid ☑
RobotModel ☑
TF □
LaserScan ☑
Bumper Hit ☑
Map ☑
Global Map ☑
Local Map ☑
Costmap ☑
Planner ☑
Status: Ok
Topic /move_base/DWAPlan...
Unreliable □
Line Style Lines
Color 0; 12; 255
Alpha 1
Buffer Length 1
Offset 0; 0; 0
Pose Style None
Cost Cloud ☑
Trajectory C... ☑
Status: Ok
Topic /move_base/DWAPlan...
Unreliable □
Selectable ☑
Style Flat Squares

Type: TopDownOrtho
Current View TopDownOrtho ...
Near Clip ... 0.01
Target Fra... <Fixed Frame>
Scale 85.863
Angle -6.72003
X -1.35336
Y 2.04398

- Run visualization system

```
$ roslaunch turtlebot_rviz_launchers view_navigation.launch
```

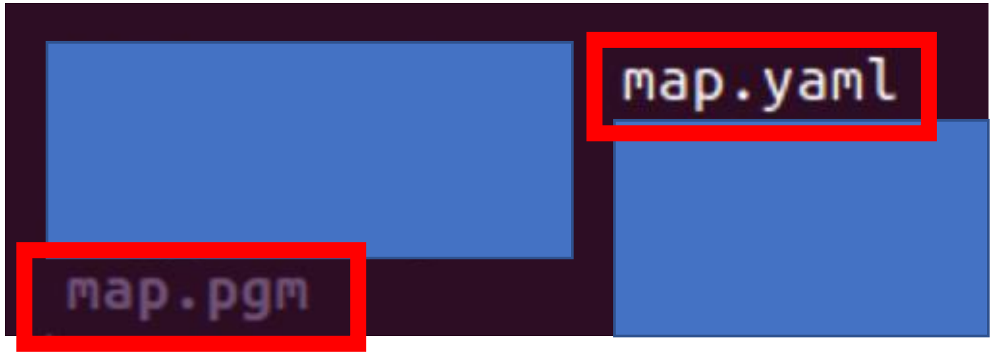
- Run remote controller

```
$ roslaunch turtlebot_teleop logitech.launch
```

- If don’t have joy

```
$ roslaunch turtlebot_teleop keyboard_teleop.launch
```

Turtlebot2 - Create map



```
[redacted]  map.yaml
map.pgm     [redacted]
```

- Save your map

```
$ cd ~/
```

```
$ rosrun map_server map_saver -f map
```

```
$ ls
```

Turtlebot2 - Navigation

- Kill the Gmapping
- Launch navigation with AMCL

```
$ roslaunch turtlebot_navigation amcl_demo.launch map_file:=/home/$USER/map.yaml
```

- Localize your robot
- Try to use 2D Nav Goal
- Try to use navigation_node.py service to
  - Save location
  - Navigate to location