**ASA Delivery System**

**ROS Development Kit**

**for Keenon W3Pro**

[1. Asa\_charge](#_i3uhdupz4nwl)

[- Asa\_charge\_state](#_grtxnfvwokq6)

[- Asa\_charge\_task](#_hab0uyakhl6o)

[2. Asa\_delivery](#_bzeefi9tqqmh)

[- Asa\_cancel](#_hybnbj2wojxp)

[- Asa\_Enter\_lift](#_ggx9tjxa5p50)

[- Asa\_Exit\_lift](#_mtvszfxanbz6)

[- Asa\_move\_base](#_7eizzmax221e)

[- Asa\_pose](#_qs2m6n4qb4hi)

[- Asa\_sys](#_k9xa7dpluznx)

[3. Asa\_lunach](#_h8mbbqhtq89d)

[- Asa\_luanch](#_fdnqkkcsbd6o)

[4. Asa\_status](#_7vopr8ew4nux)

[- Asa\_status](#_13688ttzvteu)

[5. Asa\_switch\_floor](#_n4a44fdudpwe)

[- Asa\_switch](#_1oujthj2fqir)

## Asa\_charge

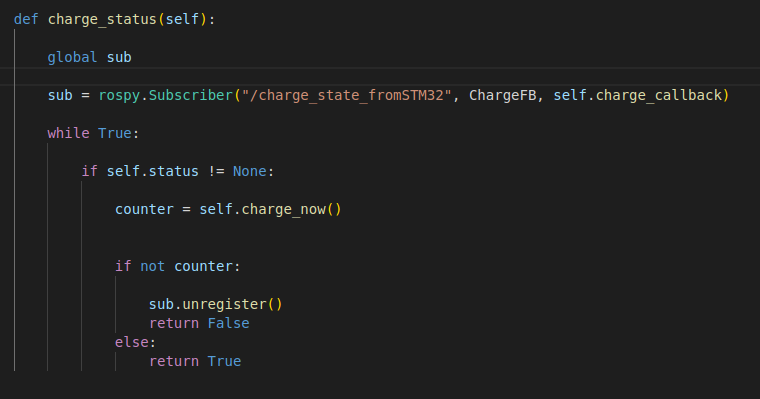
#### **1.1: Asa\_charge\_state**

A programme created to check the charging status of Keenon W3 Pro.

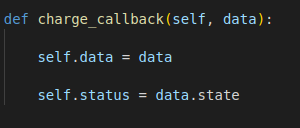
When called, the programme will subscribe to rostopic “/charge\_state\_fromSTM32” developed by Keenon Robot.

Return True or False to the server for further action

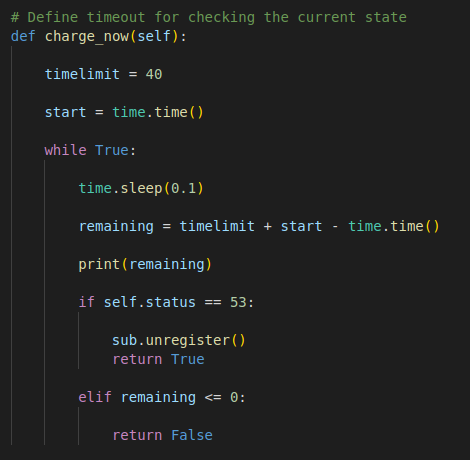
**Code Explain:**

****

Create a function for subscribing rostopic “/charge\_state\_fromSTM32” and get the data through the callback



from the callback function, the programme still store the state status from charge\_state



the programme will loop until timeout (default setting: 40s) and return to boolean

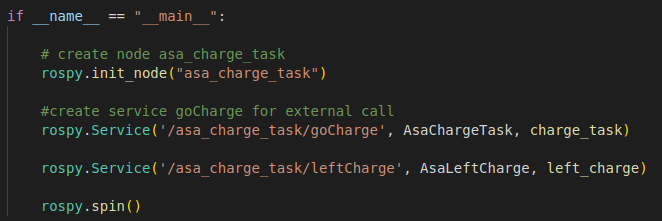
#### **1.2: Asa\_charge\_task**

A programme created for executing charging instructions while called by the server.

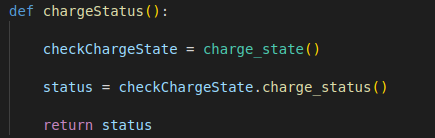
When called, the programme will create an actionlib client and send the requested data to the action server “charge\_task” for executing instructions “go\_to\_charge\_pile” and “leave\_chrage\_pile”.

Return True / False boolean to the server according to the result of the function.

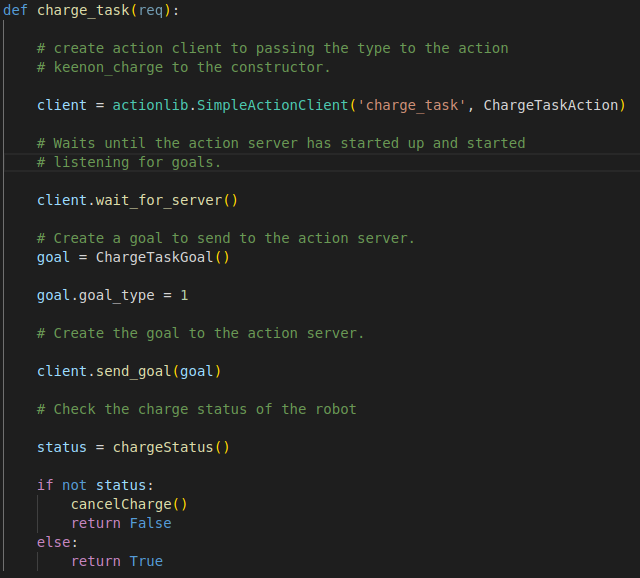
**Code Explain:**

****

Create a node and create service “goCharge” and “leftCharge” for external call

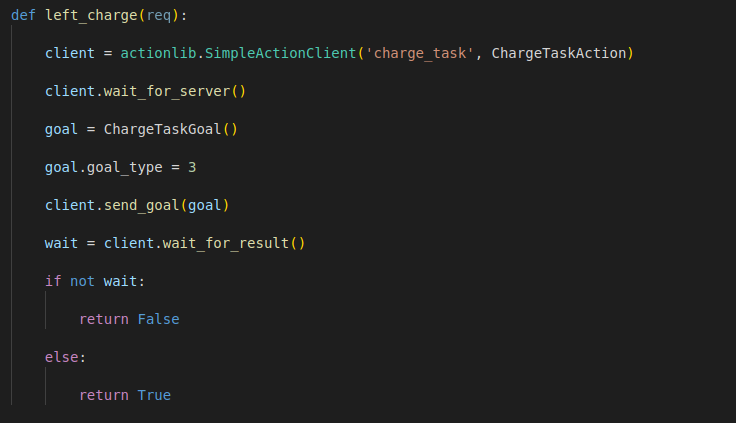


Create a function for calling asa\_charge\_state to get the data from topic “/charge\_state\_fromSTM32”



Create function “charge\_task” as the callback of service “goCharge” with a actionlib client for sending instruction to the robot

Inorder to cancel the charge task, the programme import GoalID from actionlib\_msg and publish it to rostopic “charge\_task”



Create function “left\_charge” as the callback of service “leftCharge” which has the same structure of function “charge\_task”

## Asa\_delivery

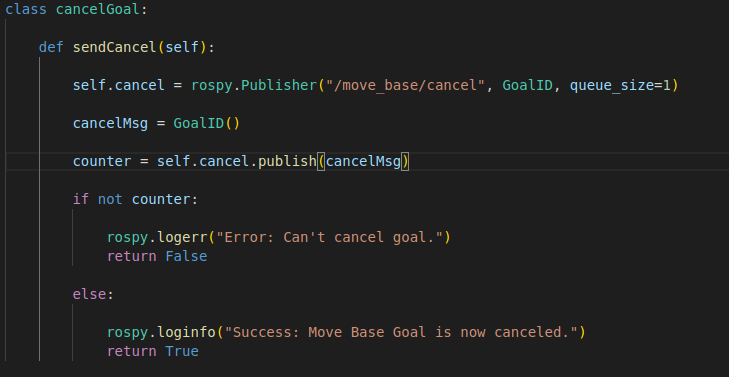
#### **2.1: Asa\_cancel**

A programme created for cancel the action of actionlib server “move\_base\_action”

This function will be called when the move\_base’s mission is failed in following situation:

1. The Goal Location is blocked
2. Run out of mission timeout

**Code Explain:**

****

Create function “sendCancel” for canceling the goal while the robot cannot arrive the goal position

In order to cancel the move base goal, the programme import GoadID from actionlib\_msg, and publish it to the topic “/move\_base/cancel” to clear the move\_base\_action instruction.

#### 

#### **2.2: Asa\_Enter\_lift**

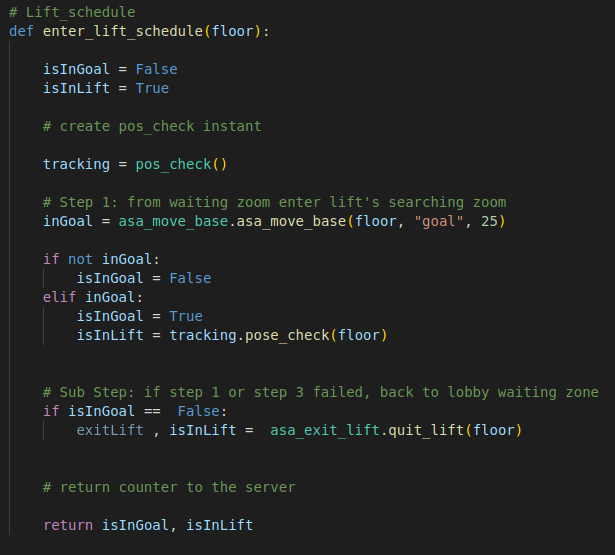
A programme created for instructing the robot to go into the lift’s waiting zone.

The programme will check the pose of the robot while the robot successfully arrives at the target zone or fails.

If the robot fails to enter the target waiting zone, the robot will call out the function “asa\_exit\_lift” automatically and wait for return.

This Function have two return - ifInLift and isInGoal

**Code Explain:**



Create function “enter\_lift\_schedule” and send the goal’s data to “asa\_move\_base”

While the execution of entering goal position fail or over time limit, call “asa\_exit\_lift”

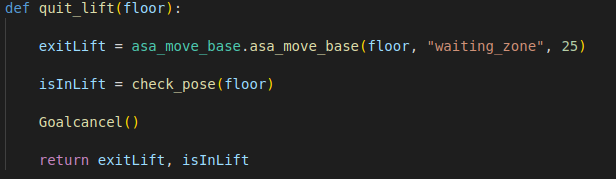
#### **2.3: Asa\_Exit\_lift**

A programme created for instructing the robot to exit the lift area.

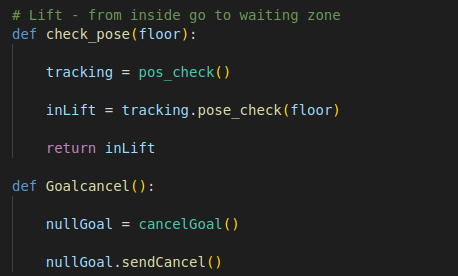
The programme subscribes to “asa\_pose” and will check the pose of the robot while the robot successfully exits the lift or fails.

This Function have two return - ifInLift and isInGoal

**Code Explain:**

****

Create function “quit\_lift” and send the goal’s data to “asa\_move\_base”

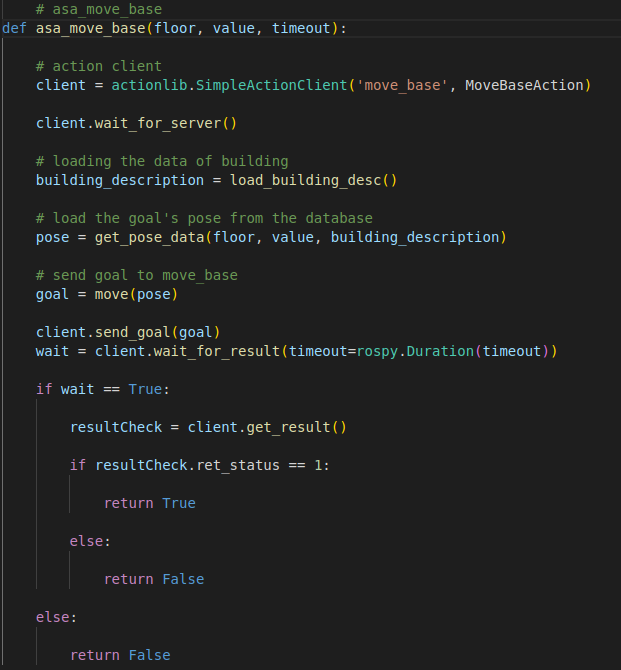


Subscribe to “Asa\_cancel”, cancel move\_base\_goal while over time limit

#### **2.4: Asa\_move\_base**

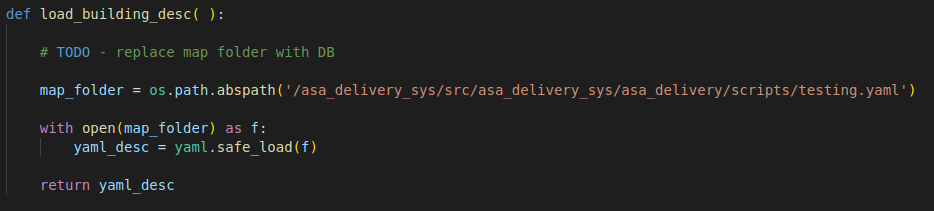
A programme created for executing the moving instruction from the server. Providing a point to point moving service.

**Code Explain:**

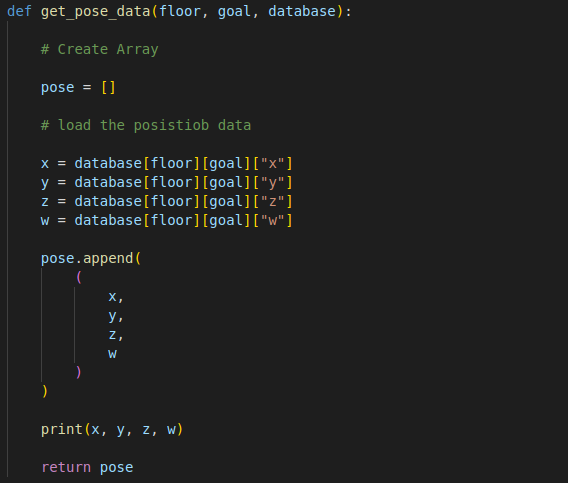
****

Create function “asa\_move\_base” as the server and create a actionlib client subscribe to ‘move\_base’

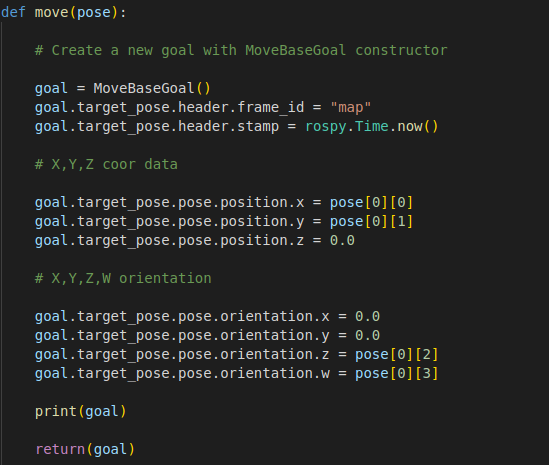
Send the goal to move\_base and wait\_for the result



Create function “load\_building\_desc” to load the data from the database by the target pose data from the server input.



Create function “get\_pose\_data” to record the data needed passing to the move\_base and store to a list.



Create a function “move” to make the data from recorded link to the ‘MoveBaseGoal’ type message and return

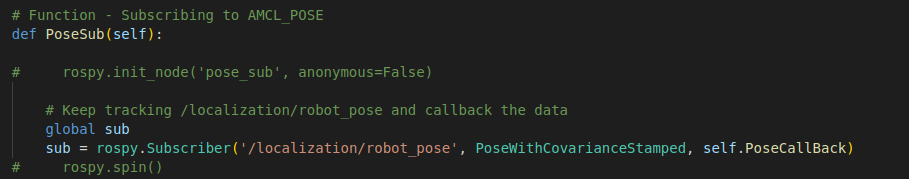
#### 

#### **2.5: Asa\_pose**

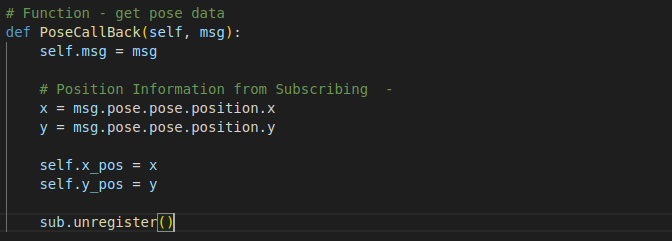
A programme created for checking the current position of the robot. Helps the server determine whether the robot successfully executes the move\_base action or not.

If the robot is in the specified position. return True. Else, return False.

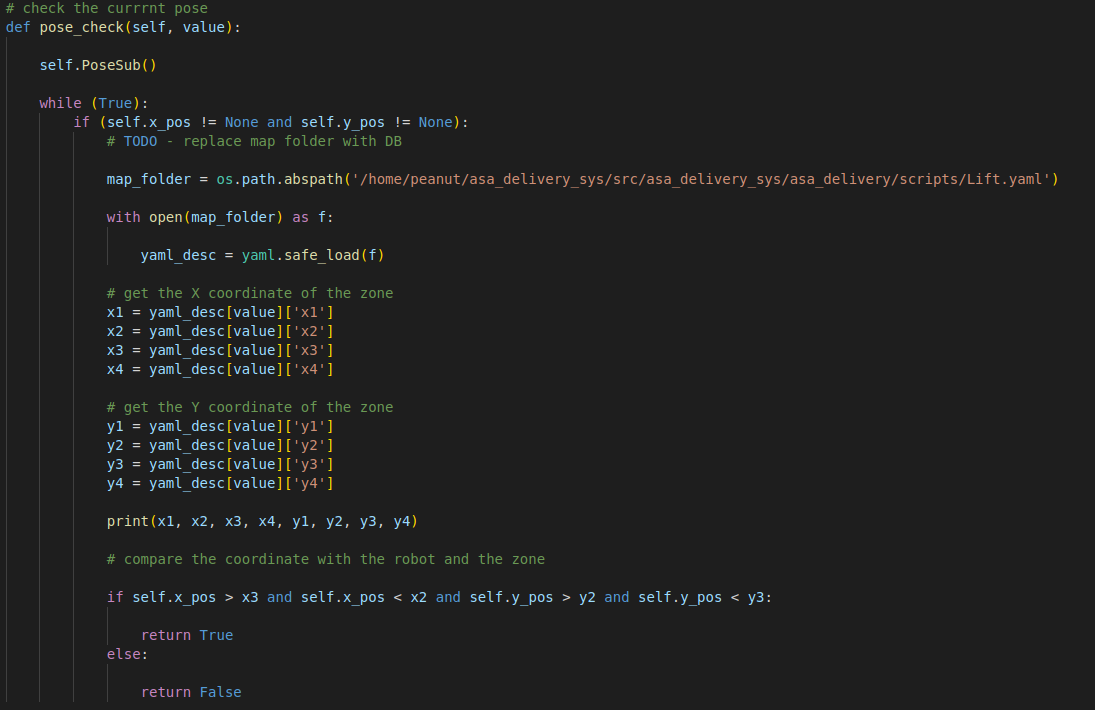
**Code Explain:**

****

Create function ‘posesub’ that subscribe to the rostopic '/localization/robot\_pose'



The callback of the subscription, helps update the current X,Y position on the map frame.



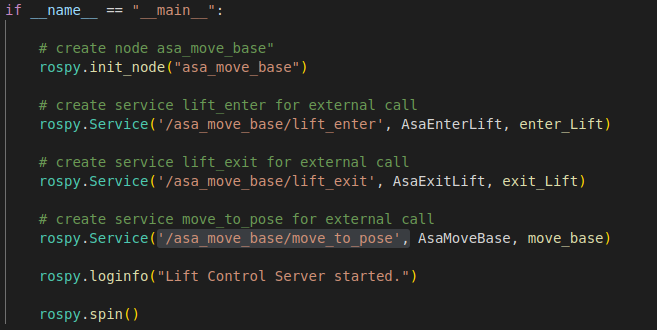
Create a function ‘pose\_check’ for checking if the robot enters the target zone or not.

#### 

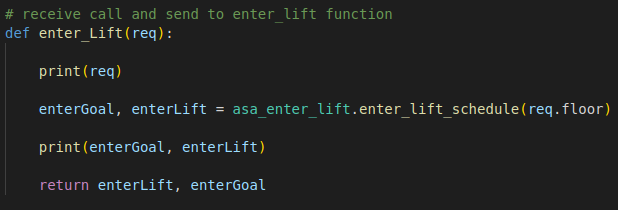
#### **2.6: Asa\_sys**

A programme as the server creates ros service of each function and called by the external end user.

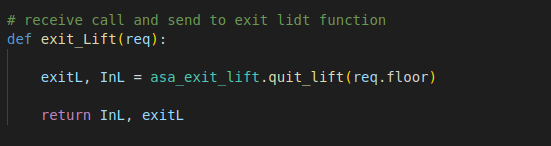
**Code Explain:**

****

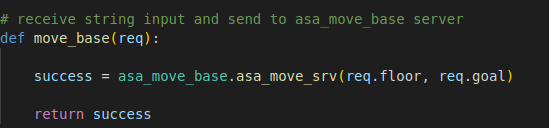
Create node “asa\_move\_base” and service '/asa\_move\_base/lift\_enter','/asa\_move\_base/lift\_exit','/asa\_move\_base/move\_to\_pose'



With the user’s input, call programme ‘asa\_enter\_lift’



With the user’s input, call programme ‘asa\_exit\_lift’



With the user’s input, call programme ‘asa\_move\_base’

## Asa\_lunach

#### **Asa\_luanch**

A lunch directory to run the python code inside the package among the workspace “Asa\_delivery\_sys”

Called by common\_robot.launch developed by keenon while the robot booted.

## Asa\_status

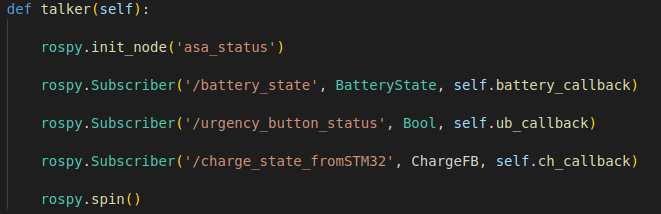
#### **Asa\_status**

This package was created to keep posting the feedback from the subscribed topic.

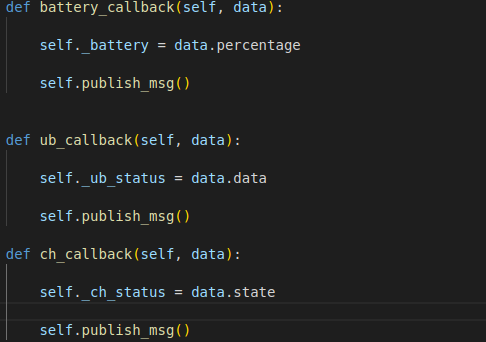
**Code Explain:**

****

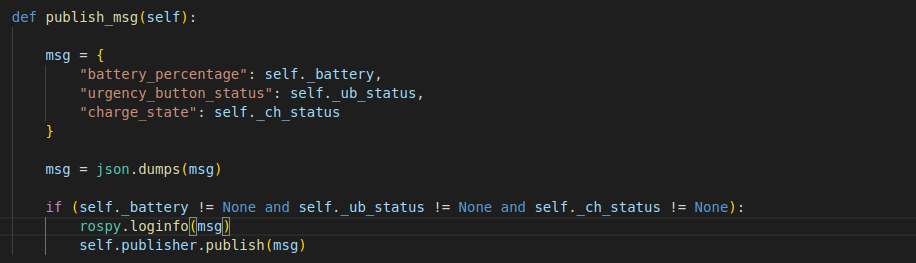
Create a topic for the server listening.



Create a function to subscribe to the topic that will publish the robot status data.



Call function “pulish\_msg” after ros topic subscribed pulish data to the callback

****

publish receive data from the callback to the topic created

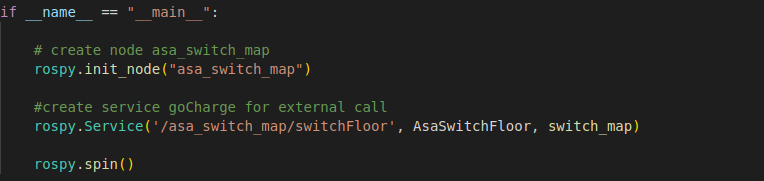
## 

## Asa\_switch\_floor

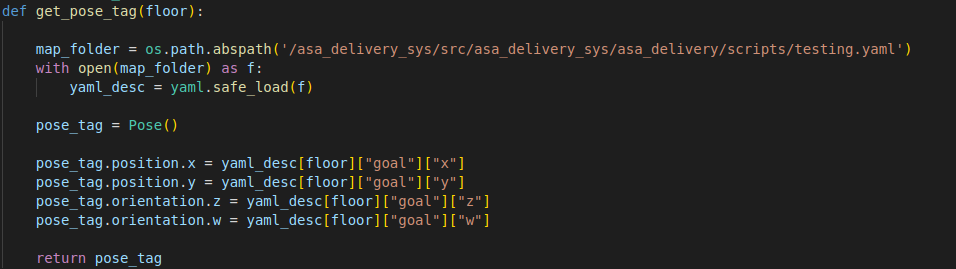
#### **Asa\_switch**

A package created for switching the map of the robot for multiple floor mission.

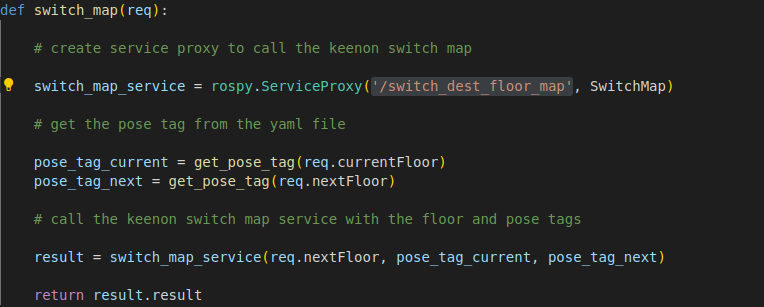
**Code Explain:**

****

Created function “switch\_map” which subscribes to service '/switch\_dest\_floor\_map' developed by keenon.



Created function “get\_pose\_tag” that stored the location data of target floor from database.



Execute switching floor and localize the robot on the new floor map.