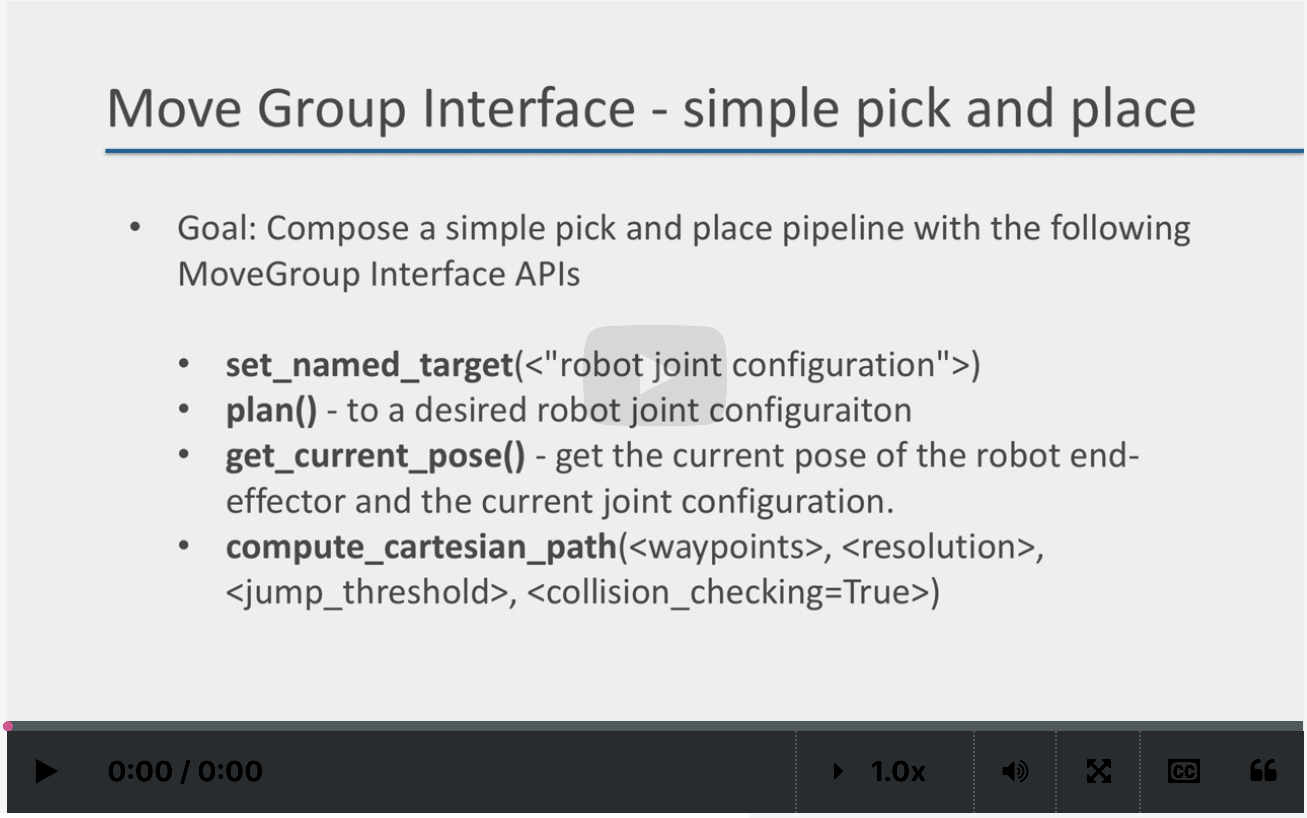
In this lecture, we will compose a simple pick and place pipeline with different MoveGroup APIs. Furthermore, the purpose and function of different APIs will be explained.

Important Note:

In the new version of ROS, the plan() function returns a tuple. Therefore, the old code:
`plan = robot1_group.plan(),` could no longer be used.
It has been replaced by the following code: `_, plan, _, _ = robot1_group.plan()`

Simple pick & place: Part 1



Move Group Interface - simple pick and place

- Goal: Compose a simple pick and place pipeline with the following MoveGroup Interface APIs
  - **set_named_target**(<"robot joint configuration">)
  - **plan()** - to a desired robot joint configuraiton
  - **get_current_pose()** - get the current pose of the robot end-effector and the current joint configuration.
  - **compute_cartesian_path**(<waypoints>, <resolution>, <jump_threshold>, <collision_checking=True>)

Video ▼

Subtitles ▼

Other ▼

MoveGroup Interface has a couple of different functions to interface with its API:

- `set_named_target` (<"robot joint configuration">) - set a certain robot configuration as target
- `plan()` - plans a motion to the goal
- `get_current_pose()` - get pose of the end effector and joint configuration
- `compute_cartesian_path` (<waypoints>, <resolution>, <jump_threshold>, <collision_checking=True>

Review the the *simple_pick_place.py* script.

Navigate to the week 4 ROS package

```
$ source $HOME/hrwros_ws/devel/setup.bash
$ roscd hrwros_week4/scripts
```

Open the `simple_pick_place.py` script, using your favorite editor (outside the CCS). Let's go through it:

The required modules related to MoveIt are:

- `moveit_commander` - Tells python we work with MoveIt
- `moveit_msgs.msg` - Loads the MoveIt specific ROS messages
- `actionlib` - For the movement with `actionlib.SimpleActionClient()`
- `geometry_msgs` - Loads the required messages for planning linear or cartesian spaced motions

Further in the script:

- The initialization of `moveit_commander,` and ROS node named '`simple_pick_place'`

- Create move groups for each one of the robots.
- Instantiate the two action clients, one for each robot, so they can use the execute_trajectory action server.
- Use of the APIs from MoveIt
  - `set_named_target(<"robot joint configuration">)` - set a goal configuration (If it does not exist, you will need to create them on the assignments)
  - `plan()` - plans a motion to the goal
  - `send_goal(<robot goal>)` - sends the goal to the action server

Since this API works via actions, we can start computing the next trajectory while the robot is still executing the current one.

Question 1

1 point possible (ungraded)

Which of the following import statements ensure that MoveIt specific functionalities are available to the simple_pick_place.py script?

There are two correct answers.

□ import rospy

□ import moveit_msgs.msg

□ import actionlib

□ import moveit_commander

Submit

Question 2

1 point possible (ungraded)

The set_named_target API accepts robot poses specifed using the geometry_msgs/Pose message type as argument.

○ True

○ False

Submit