pycram.designators.motion_designator

Contents

Classes

MoveMotion	Moves the robot to a designated location
MoveTCPMotion	Moves the Tool center point (TCP) of the robot
LookingMotion	Lets the robot look at a point
MoveGripperMotion	Opens or closes the gripper
DetectingMotion	Tries to detect an object in the FOV of the robot
MoveArmJointsMotion	Moves the joints of each arm into the given position
WorldStateDetectingMotion	Detects an object based on the world state.
MoveJointsMotion	Moves any joint on the robot
OpeningMotion	Designator for opening container
ClosingMotion	Designator for closing a container
TalkingMotion	Talking Motion, lets the robot say a sentence.

Module Contents

class pycram.designators.motion_designator.**MoveMotion**

Bases: pycram.designator.BaseMotion

Moves the robot to a designated location

target: *pycram.datastructures.pose.Pose*

Location to which the robot should be moved

perform()

Passes this designator to the process module for execution. Will be overwritten by each motion.

to_sql() → pycram.orm.motion_designator.MoveMotion

Create an ORM object that corresponds to this description. Will be overwritten by each motion.

Returns:

The created ORM object.

insert(*session*, **args*, ***kwargs*) → pycram.orm.motion_designator.MoveMotion

Add and commit this and all related objects to the session. Auto-Incrementing primary keys and foreign keys have to be filled by this method.

Parameters:

- **session** – Session with a database that is used to add and commit the objects
- **args** – Possible extra arguments
- **kwargs** – Possible extra keyword arguments

Returns:

The completely instanced ORM motion.

class pycram.designators.motion_designator.MoveTCPMotion

Bases: pycram.designator.BaseMotion

Moves the Tool center point (TCP) of the robot

target: *pycram.datastructures.pose.Pose*

Target pose to which the TCP should be moved

arm: *pycram.datastructures.enums.Arms*

Arm with the TCP that should be moved to the target

allow_gripper_collision: *typing_extensions.Optional[bool] = None*

If the gripper can collide with something

perform()

Passes this designator to the process module for execution. Will be overwritten by each motion.

to_sql() → pycram.orm.motion_designator.MoveTCPMotion

Create an ORM object that corresponds to this description. Will be overwritten by each motion.

Returns:

The created ORM object.

insert(*session: sqlalchemy.orm.Session*, **args*, ***kwargs*) → pycram.orm.motion_designator.MoveTCPMotion

Add and commit this and all related objects to the session. Auto-Incrementing primary keys and foreign keys have to be filled by this method.

Parameters:

- **session** – Session with a database that is used to add and commit the objects
- **args** – Possible extra arguments
- **kwargs** – Possible extra keyword arguments

Returns:

The completely instanced ORM motion.

class pycram.designators.motion_designator.LookingMotion

Bases: pycram.designator.BaseMotion

Lets the robot look at a point

target: *pycram.datastructures.pose.Pose*

perform()

Passes this designator to the process module for execution. Will be overwritten by each motion.

to_sql() → pycram.orm.motion_designator.LookingMotion

Create an ORM object that corresponds to this description. Will be overwritten by each motion.

Returns:

The created ORM object.

insert(*session: sqlalchemy.orm.Session*, **args*, ***kwargs*) → pycram.orm.motion_designator.LookingMotion

Add and commit this and all related objects to the session. Auto-Incrementing primary keys and foreign keys have to be filled by this method.

Parameters:

- **session** – Session with a database that is used to add and commit the objects
- **args** – Possible extra arguments
- **kwargs** – Possible extra keyword arguments

Returns:

The completely instanced ORM motion.

class pycram.designators.motion_designator.**MoveGripperMotion**

Bases: pycram.designator.BaseMotion

Opens or closes the gripper

motion: *pycram.datastructures.enums.GripperState*

Motion that should be performed, either ‘open’ or ‘close’

gripper: *pycram.datastructures.enums.Arms*

Name of the gripper that should be moved

allow_gripper_collision: *typing_extensions.Optional[bool] = None*

If the gripper is allowed to collide with something

perform()

Passes this designator to the process module for execution. Will be overwritten by each motion.

to_sql() → pycram.orm.motion_designator.MoveGripperMotion

Create an ORM object that corresponds to this description. Will be overwritten by each motion.

Returns:

The created ORM object.

latest

insert(*session: sqlalchemy.orm.Session*, **args*, ***kwargs*) → pycram.orm.motion_designator.MoveGripperMotion

Add and commit this and all related objects to the session. Auto-Incrementing primary keys and foreign keys have to be filled by this method.

Parameters:

- **session** – Session with a database that is used to add and commit the objects
- **args** – Possible extra arguments
- **kwargs** – Possible extra keyword arguments

Returns:

The completely instanced ORM motion.

class pycram.designators.motion_designator.**DetectingMotion**

Bases: pycram.designator.BaseMotion

Tries to detect an object in the FOV of the robot

object_type: *pycram.datastructures.enums.ObjectType*

Type of the object that should be detected

perform()

Passes this designator to the process module for execution. Will be overwritten by each motion.

to_sql() → pycram.orm.motion_designator.DetectingMotion

Create an ORM object that corresponds to this description. Will be overwritten by each motion.

Returns:

The created ORM object.

insert(*session: sqlalchemy.orm.Session*, **args*, ***kwargs*) → pycram.orm.motion_designator.DetectingMotion

Add and commit this and all related objects to the session. Auto-Incrementing primary keys and foreign keys have to be filled by this method.

Parameters:

- **session** – Session with a database that is used to add and commit the objects
- **args** – Possible extra arguments
- **kwargs** – Possible extra keyword arguments

Returns:

The completely instanced ORM motion.

class pycram.designators.motion_designator.MoveArmJointsMotion

Bases: pycram.designator.BaseMotion

Moves the joints of each arm into the given position

left_arm_poses: *typing_extensions.Optional[typing_extensions.Dict[str, float]] = None*

Target positions for the left arm joints

right_arm_poses: *typing_extensions.Optional[typing_extensions.Dict[str, float]] = None*

Target positions for the right arm joints

perform()

Passes this designator to the process module for execution. Will be overwritten by each motion.

to_sql() → pycram.orm.motion_designator.Motion

Create an ORM object that corresponds to this description. Will be overwritten by each motion.

Returns:

The created ORM object.

insert(*session: sqlalchemy.orm.Session*, **args*, ***kwargs*) → pycram.orm.motion_designator.Motion

Add and commit this and all related objects to the session. Auto-Incrementing primary keys and foreign keys have to be filled by this method.

Parameters:

- **session** – Session with a database that is used to add and commit the objects
- **args** – Possible extra arguments
- **kwargs** – Possible extra keyword arguments

Returns:

The completely instanced ORM motion.

class pycram.designators.motion_designator.**WorldStateDetectingMotion**

Bases: pycram.designator.BaseMotion

Detects an object based on the world state.

object_type: *pycram.datastructures.enums.ObjectType*

Object type that should be detected

perform()

Passes this designator to the process module for execution. Will be overwritten by each motion.

to_sql() → pycram.orm.motion_designator.Motion

Create an ORM object that corresponds to this description. Will be overwritten by each motion.

Returns:

The created ORM object.

insert(*session: sqlalchemy.orm.Session*, **args*, ***kwargs*) → pycram.orm.motion_designator.Motion

Add and commit this and all related objects to the session. Auto-Incrementing primary keys and foreign keys have to be filled by this method.

Parameters:

- **session** – Session with a database that is used to add and commit the objects
- **args** – Possible extra arguments
- **kwargs** – Possible extra keyword arguments

Returns:

The completely instanced ORM motion.

class pycram.designators.motion_designator.**MoveJointsMotion**

Bases: pycram.designator.BaseMotion

Moves any joint on the robot

names: *list*

List of joint names that should be moved

latest

positions: *list*

Target positions of joints, should correspond to the list of names

perform()

Passes this designator to the process module for execution. Will be overwritten by each motion.

to_sql() → pycram.orm.motion_designator.Motion

Create an ORM object that corresponds to this description. Will be overwritten by each motion.

Returns:

The created ORM object.

insert(*session: sqlalchemy.orm.Session*, **args*, ***kwargs*) → pycram.orm.motion_designator.Motion

Add and commit this and all related objects to the session. Auto-Incrementing primary keys and foreign keys have to be filled by this method.

Parameters:

- **session** – Session with a database that is used to add and commit the objects
- **args** – Possible extra arguments
- **kwargs** – Possible extra keyword arguments

Returns:

The completely instanced ORM motion.

class pycram.designators.motion_designator.OpeningMotion

Bases: pycram.designator.BaseMotion

Designator for opening container

object_part: *pycram.designators.object_designator.ObjectPart.Object*

Object designator for the drawer handle

arm: *pycram.datastructures.enums.Arms*

Arm that should be used

perform()

Passes this designator to the process module for execution. Will be overwritten by each motion.

to_sql() → pycram.orm.motion_designator.OpeningMotion

Create an ORM object that corresponds to this description. Will be overwritten by each motion.

Returns:

The created ORM object.

insert(*session: sqlalchemy.orm.Session*, **args*, ***kwargs*) → pycram.orm.motion_designator.OpeningMotion

Add and commit this and all related objects to the session. Auto-Incrementing primary keys and foreign keys have to be filled by this method.

Parameters:

- **session** – Session with a database that is used to add and commit the objects
- **args** – Possible extra arguments
- **kwargs** – Possible extra keyword arguments

Returns:

The completely instanced ORM motion.

class pycram.designators.motion_designator.**ClosingMotion**

Bases: pycram.designator.BaseMotion

Designator for closing a container

object_part: *pycram.designators.object_designator.ObjectPart.Object*

Object designator for the drawer handle

arm: *pycram.datastructures.enums.Arms*

Arm that should be used

perform()

Passes this designator to the process module for execution. Will be overwritten by each motion.

to_sql() → pycram.orm.motion_designator.ClosingMotion

Create an ORM object that corresponds to this description. Will be overwritten by each motion.

latest

Returns:

The created ORM object.

insert(*session: sqlalchemy.orm.Session*, **args*, ***kwargs*) → pycram.orm.motion_designator.ClosingMotion

Add and commit this and all related objects to the session. Auto-Incrementing primary keys and foreign keys have to be filled by this method.

Parameters:

- **session** – Session with a database that is used to add and commit the objects
- **args** – Possible extra arguments
- **kwargs** – Possible extra keyword arguments

Returns:

The completely instanced ORM motion.

class pycram.designators.motion_designator.**TalkingMotion**

Bases: pycram.designator.BaseMotion

Talking Motion, lets the robot say a sentence.

cmd: *str*

Talking Motion, let the robot say a sentence.

perform()

Passes this designator to the process module for execution. Will be overwritten by each motion.

to_sql() → pycram.orm.motion_designator.Motion

Create an ORM object that corresponds to this description. Will be overwritten by each motion.

Returns:

The created ORM object.

insert(*session: sqlalchemy.orm.Session*, **args*, ***kwargs*) → pycram.orm.motion_designator.Motion

Add and commit this and all related objects to the session. Auto-Incrementing primary keys and foreign keys have to be filled by this method.

Parameters:

- **session** – Session with a database that is used to add and commit the objects
- **args** – Possible extra arguments
- **kwargs** – Possible extra keyword arguments

Returns:

The completely instanced ORM motion.