Intelligent Robotics

Motion Planning in Practice:
Motion Planning in ROS using Python

Philipp Ennen, M.Sc.

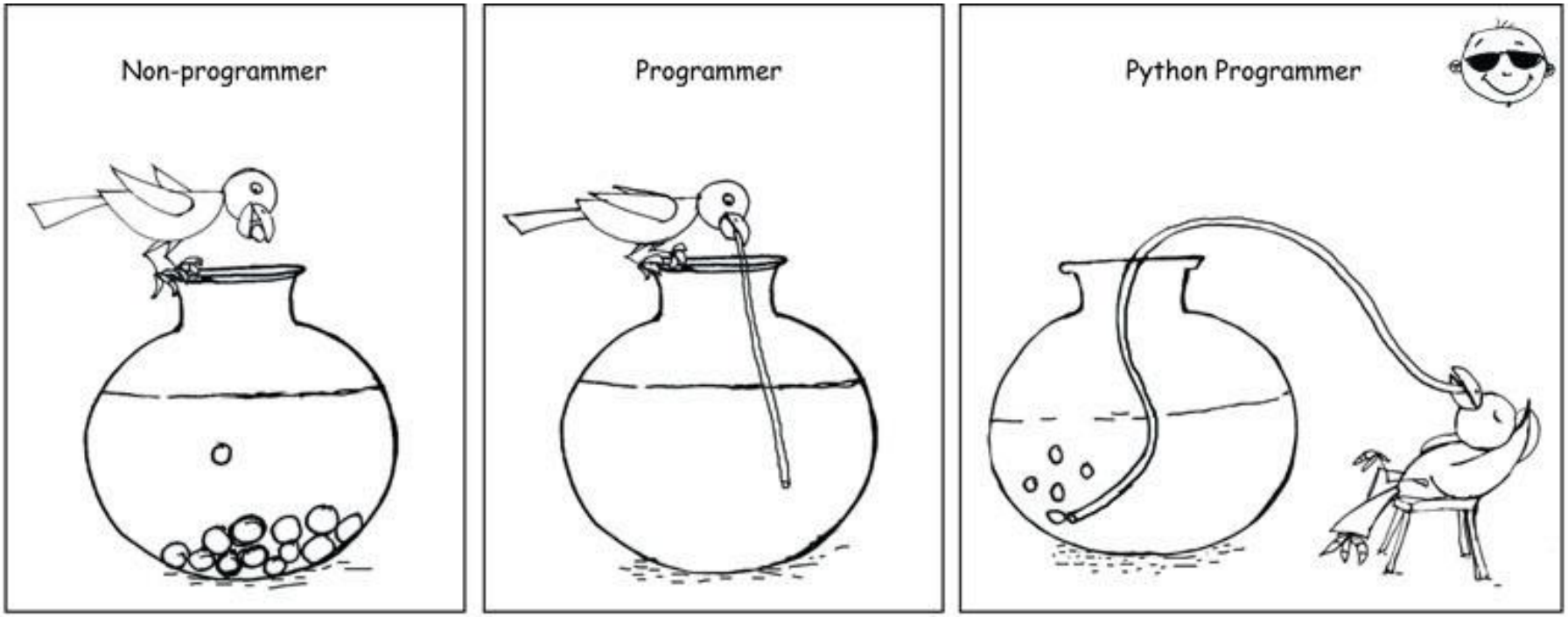
IfU IMA ZLW RWTH AACHEN UNIVERSITY

Outline

Outline

The Python Programming Language

How does a Crow get Water



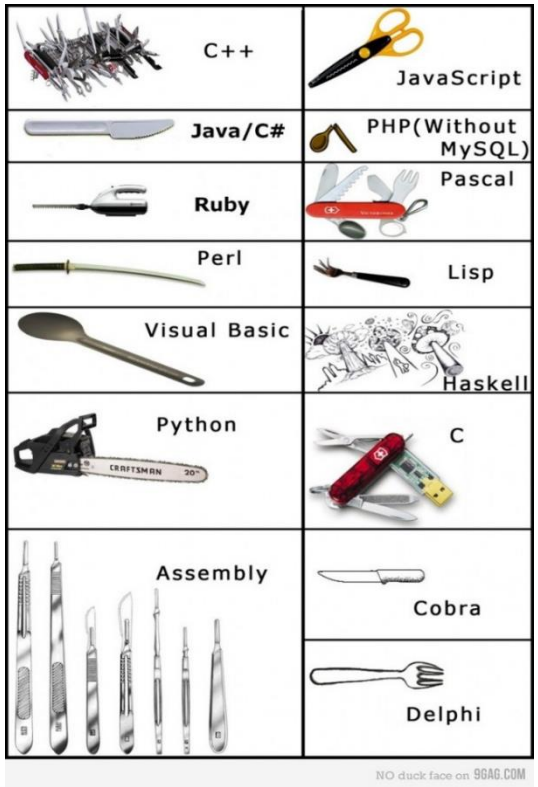
[Source: http://lpycot.appspot.com/]

The Python Programming Language

Introduction

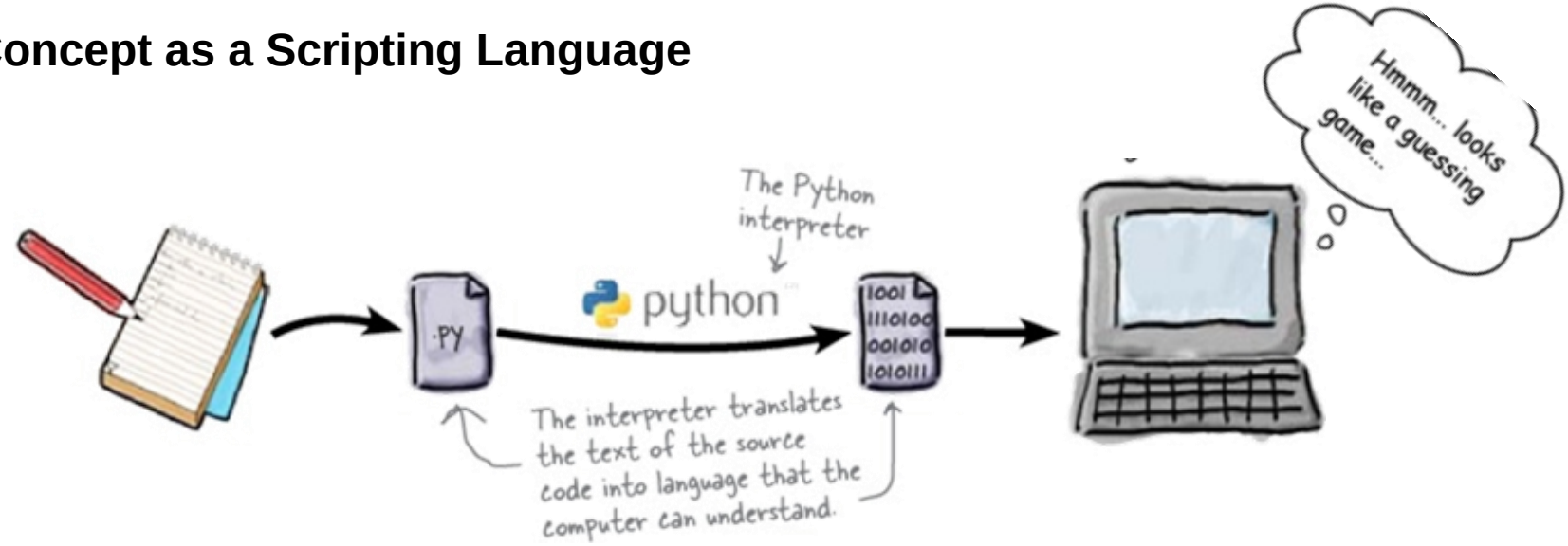
- Created by **Guido van Rossum** in the early 1990s
- Dervied from many other languages: ABC, Modula-3, C, C++, Algol-68, SmallTalk, Unix shell and other **scripting** languages.
- Available unter GNU General Public License.
- Features:
  - Easy-to-learn
  - Easy-to-read
  - Easy-to-maintain
  - A broad standard library
  - Interactive Mode
  - Portable
  - Extendable
  - Databases
  - Graphic User Interface (GUI) Programming
  - Scalable
- Usage:
  - Data Analysis
  - Machine Learning
  - Computer Vision
  - IoT
  - Game Development
  - Web Development
  - GUI Development
  - Rapid Prototyping

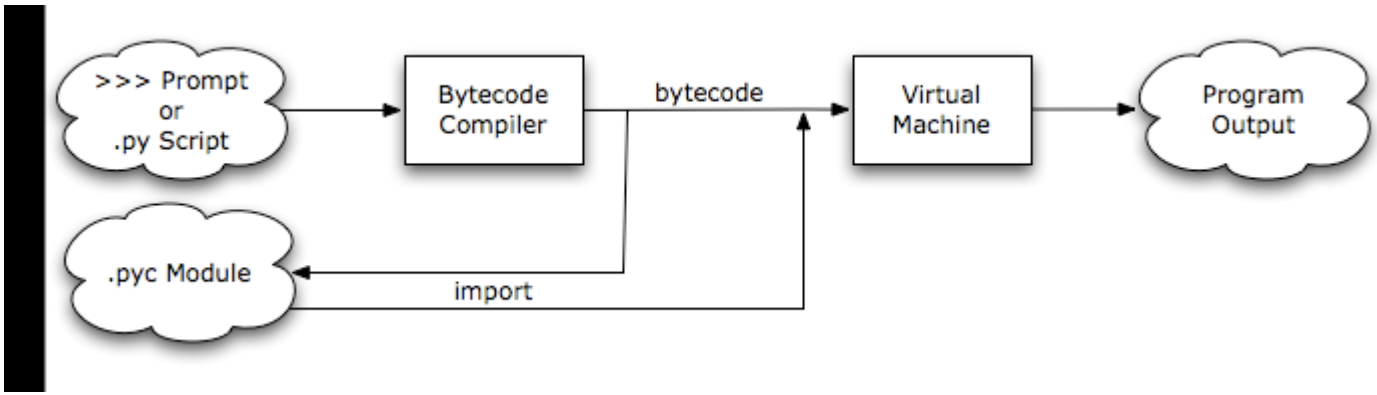

Guido van Rossum



Outline

Concept as a Scripting Language





Outline

Python Basic

Variable and Data Structures:

- Numbers and String:
  - `int, long, float, complex`
  - `string`
  - `Multiple Assignment`
- Dynamically Typed:

  Variable Typ can be changed dynamically.
- Strong Typed:

  Enforce the Variables after it figures them out.
- Naming Rules:
  - Case sensitive
  - Contains letters, numbers, underscores
  - But can‘t start with numbers
  - Can‘t contain reserved Words

int	long	float	complex
10	51924361L	0.0	3.14j
100	-0x19323L	15.20	45.j
-786	0122L	-21.9	9.322e-36j
080	0xDEFABCECBDAECBFBAEl	32.3+e18	.876j
-0490	535633629843L	-90.	-.6545+0J
-0x260	-052318172735L	-32.54e100	3e+26J
0x69	-4721885298529L	70.2-E12	4.53e-7j


```
str = 'Hello World!'

print str          # Prints complete string
print str[0]       # Prints first character of the string
print str[2:5]     # Prints characters starting from 3rd to 5th
print str[2:]      # Prints string starting from 3rd character
print str * 2      # Prints string two times
print str + "TEST" # Prints concatenated string
```

False	class	finally	is	return
None	continue	for	lambda	try
True	def	from	nonlocal	while
and	del	global	not	with
as	elif	if	or	yield
assert	else	import	pass	
break	except	in	raise	

Reserved Words

Python Basic

Variable and Data Structures:

- Data Structures:
  - **Lists**: Most versatile data types, keeps order `[1, ‘2‘, [3], {4}, (5), {6:‘6‘}, …]`
  - Tuples: similar to list, but a „**read-only**“ list, `(1, ‘2‘, [3], {4}, (5), {6:‘6‘}, …)`
  - **Dictionary**: `{‘int‘: 1, ‘str‘: ‘2‘, ‘list‘: [3], ‘set‘: {4}, ‘tup‘: (5), ‘dict‘: {6:‘6‘}, …}`
    - Associative arrays or hashes
    - Key-Value Pairs
    - Key is unique
  - Sets and fronzenset: no order, unique `{1, ‘2‘, [3], {4}, (5), {6:‘6‘}, …}`

- Others: Date, Time, hex, oct, ……

- Data Type Conversion: `new_type(x[, base])`

Outline

Logic, Condition and Loops

- Logic

Operator	Description	Example
==	If the values of two operands are equal, then the condition becomes true.	$a == b$ is not true.
!=	If values of two operands are not equal, then condition becomes true.	
<>	If values of two operands are not equal, then condition becomes true.	$a <> b$ is true. This is similar to != operator.
>	If the value of left operand is greater than the value of right operand, then condition becomes true.	$a > b$ is not true.
<	If the value of left operand is less than the value of right operand, then condition becomes true.	$a < b$ is true.
>=	If the value of left operand is greater than or equal to the value of right operand, then condition becomes true.	$a >= b$ is not true.
<=	If the value of left operand is less than or equal to the value of right operand, then condition becomes true.	$a <= b$ is true.

Operator	Description	Example
in	Evaluates to true if it finds a variable in the specified sequence and false otherwise.	x in y, here in results in a 1 if x is a member of sequence y.
not in	Evaluates to true if it does not finds a variable in the specified sequence and false otherwise.	x not in y, here not in results in a 1 if x is not a member of sequence y.
is	Evaluates to true if the variables on either side of the operator point to the same object and false otherwise.	x is y, here **is** results in 1 if id$x$ equals id y.
is not	Evaluates to false if the variables on either side of the operator point to the same object and true otherwise.	x is not y, here **is not** results in 1 if id$x$ is not equal to id$y$.

Logic, Condition and Loops

- Condition and Loops: `if, for, while`

```
if expression1:
   statement 1.1 …
elif expression2:
   statement 2.1 …
…
else:
   statement n.1 …
```

```
while condition:
   if exp1:
      continue
   elif exp2:
      pass
   else:
      break
```

```
for expression:
   if exp1:
      continue
   elif exp2:
      pass
   else:
      break
```

- Condition and Loops in Data Structures:
  - `a_list = [word for word in [‘RWTH‘, ‘Summer‘, ‘School‘, 2017] if type(word) is str]`
  - `a_set = {word for word in [‘RWTH‘, ‘Summer‘, ‘School‘, 2017, ‘RWTH‘] if type(word) is str}`
  - `A_turple = (word for word in [‘RWTH‘, ‘Summer‘, ‘School‘, 2017]) Why???`

Outline

1 Programming using Python

1.1 The Python Programming Language

1.2 Python Basic

a Variable and Data Structures

b Logic, Condition and Loops

c Functions and Object oriented Programming (OOP)

d A whole Python Script

2 Robot Operating System

3 Programming your Motion Planning for Kinova

Functions and OOP

- Function
  - Regular Functions

```
def function_name(para_a, para_b=None):
    statements……
    return res1, res2, res3
```

  - Others: lambda, yield
- OOP:
  - Object oriented and not object oriented
  - Everything in Python is a child object from `class object`


```
class class_name(faher_class=object):
      a_counter = 0

      def __init__(self, name):
         self.name = name

      def all_can_use(self, para):
         return res

      def _not_all_can_use(self, para):
         return res

      @staticmethod
      def method_for_everyone(para):
         pass

      @classmethod
      def method_within_class_instance(cls, para):
         pass
```

Outline

Python Basic

A whole Python Script

- `import` Modules

```
import a_module
import a_module as a_new_name_for_this_module
from a_module import *
from a_module import something_in_this_module
```

- Do as a Python Programmer: PEP8

PEP 8
Coding style in Python

- Be smart and always ready for error handling

```
try:
   res = trying_to_run_a_function(para)

except a_General_Error:
   print “Are you kidding me?“

Except a_user_defined_error:
   pinrt “Are you kidding youself?“

Finally:
   print “anyway you will get here.“
```

- Many advanced Features
  - Regular Expressions
  - Networking
  - Multithreading
  - GUI

IfU IMA ZLW RWTH AACHEN UNIVERSITY

Outline

Robot Operating System (ROS)

What is ROS?

Robot Software

The Robot Operating System (ROS) is a set of software libraries and tools that help you build robot applications. From drivers to state-of-the-art algorithms, and with powerful developer tools, ROS has what you need for your next robotics project. And it's all open source.

Robot Operating System (ROS)

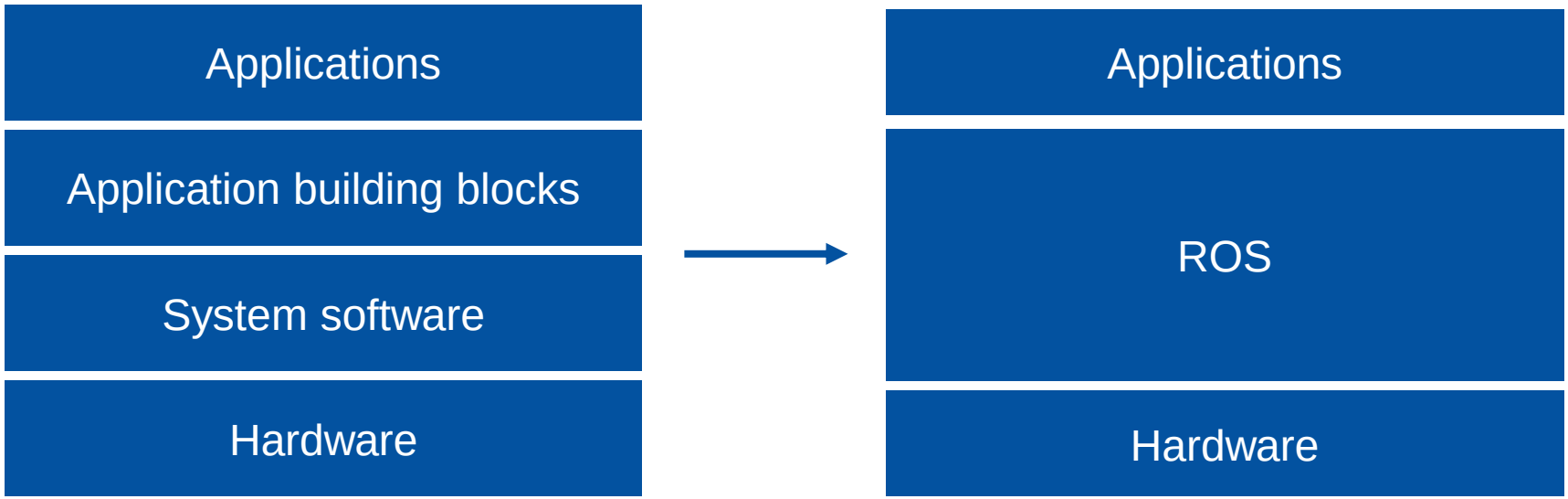
What is ROS?

- A “Meta” Operating System for robots
  - Open source
  - Runs in Linux (esp. Ubuntu)
  - Ongoing Windows implementation
- Agent based (nodes)
- Message passing
  - Publish
  - Subscribe
  - Services via remote invocation
- Supports numerous programming languages (C++. Python, Lisp, Java)

Robot Operating System (ROS)

What is ROS?

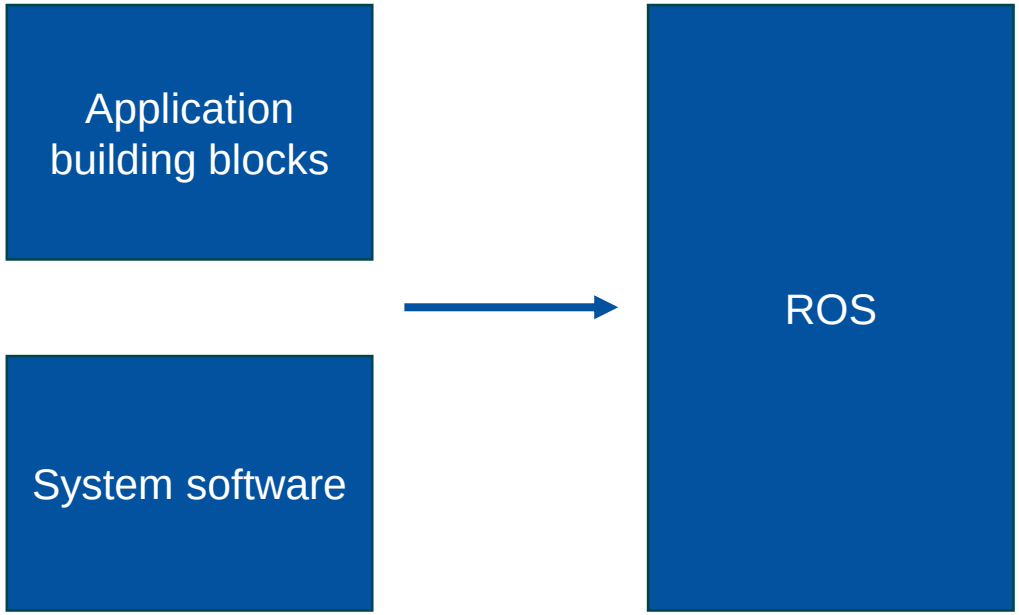
- Standardized layers
- System software abstracts
- Applications leverage other applications
- Widely existent sets of libraries



Robot Operating System (ROS)

What is ROS?

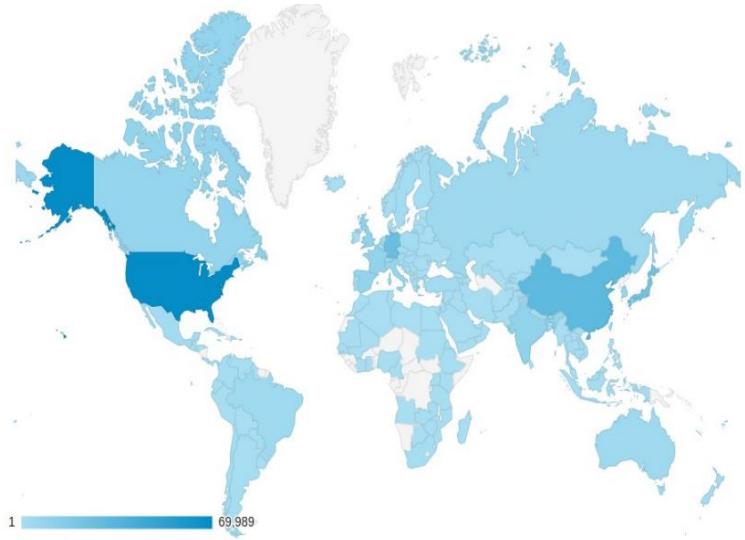
- Low level device abstraction
  - Joystick
  - GPS
  - Camera
  - Controllers
  - Laser Scanners
  - …
- Application building blocks
  - Coordinate system transforms
  - Visualization tools
  - Debugging tools
  - Robust navigation stack (SLAM with loop closure)
  - Arm path planning
  - Object recognition
  - …

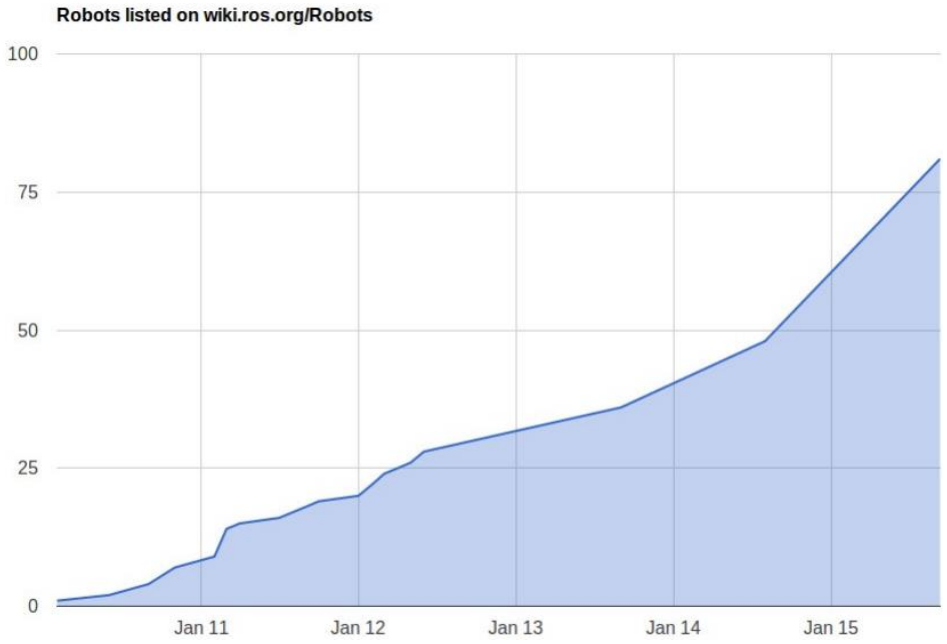


Robot Operating System (ROS)

Where is it used?

- More than 100 robots use ROS
  - http://wiki.ros.org/Robots
- Number of papers citing “ROS: an open-source Robot Operating System” (Quigley et al., 2009)
  - 4149





ROS Basics: Key Concepts

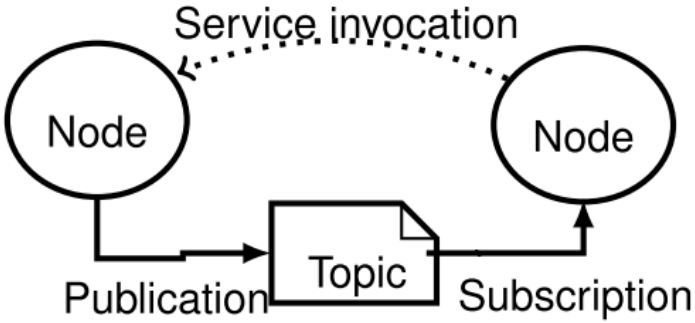
Key Concepts

- **roscore**:
  - **ROS Master**: Negotiates communication connections between nodes
  - **Parameter server**: stores persistent configuration parameters
  - **rosout**: a network-based stdout for human-readable messages

- **Package**: a virtual directory holding one or more executables (nodes)

- **Node**: An agent communicating with ROS and other nodes via
  - **Topics** (public/subscribe) using typed messages
  - **Services**: Request / Response paradigm (think of method or operation) via typed messages

- References:
  - ROS wiki: http://www.ros.org/wiki
  - ROS tutorials: http://www.ros.org/wiki/ROS/Tutorials/



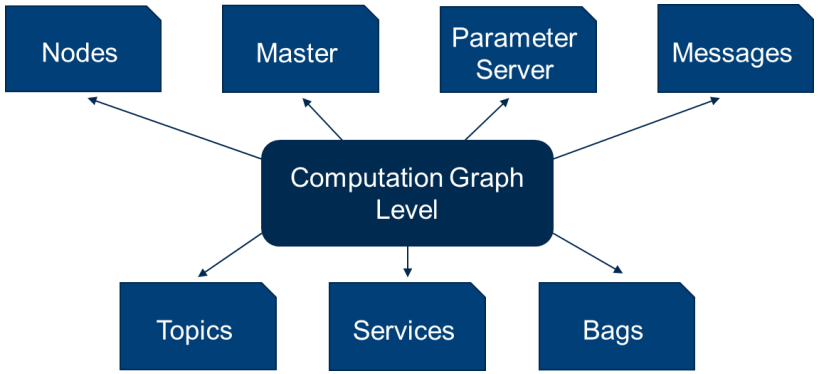
ROS Basics: Computation Graph Level

- The computation graph is the peer-to-peer network of ROS processes
- All provide data to the Graph in different ways
- Contained in the *ros_comm* repository
  - Several packages
  - ROS middleware/communications
- Core client libraries
  - roscpp, rospy, roslisp
- Graph introspection tools
  - rostopic, rosnode, rosservice, rosparam
  - e.g.

```
$ rostopic list
```

```
$ rostopic echo /cmd_vel
```

```
$ rosnode info /teleop_turtle
```

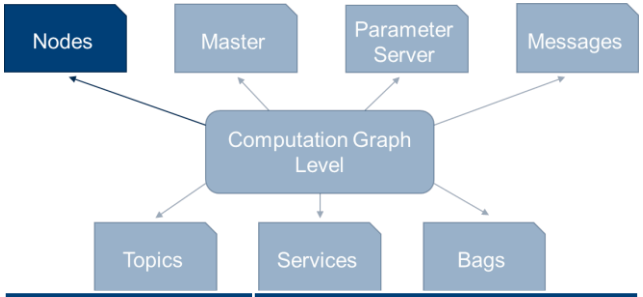


http://wiki.ros.org/ROS/Concepts : http://wiki.ros.org/ros_comm

ROS Basics: Computation Graph Level

- A *node* is a process that performs computation
- Combined together into a graph
- Communication
  - Topics
  - Services
  - Parameter Server
- E.g.:
  - One Node controls a laser range-finder
  - One Node controls the robot‘s wheel motors
  - One Node performs localization
  - …
- Fault tolerance
  - Crashes are isolated to individual nodes
- All running nodes have a *graph resource name*
  - Uniquely identification
  - E.g.: */hokuyo_node , /laser_node , /amcl*

http://wiki.ros.org/ROS/Concepts : http://wiki.ros.org/Nodes

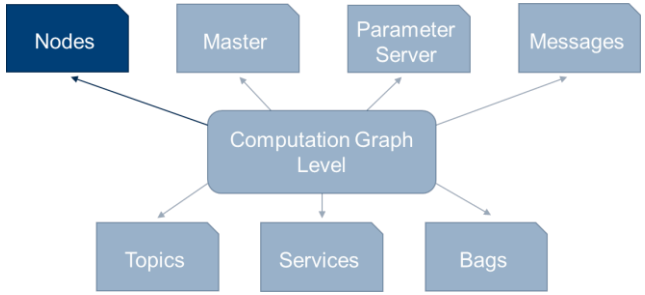


Commands	Description
$ rosnode ping	test connectivity to node
$ rosnode list	list active nodes
$ rosnode info	print information about node
$ rosnode machine	list node running on a particular machine or list machines
$ rosnode kill	kill a running node
$ rosnode cleanup	purge registration information of unreachable nodes

```
laptop@mobile-98:~$ rosnode info /teleop_turtle
--------------------------------------------------
Node [/teleop_turtle]
Publications:
 * /rosout [rosgraph_msgs/Log]
 * /turtle1/cmd_vel [geometry_msgs/Twist]

Subscriptions: None

Services:
 * /teleop_turtle/get_loggers
 * /teleop_turtle/set_logger_level
```

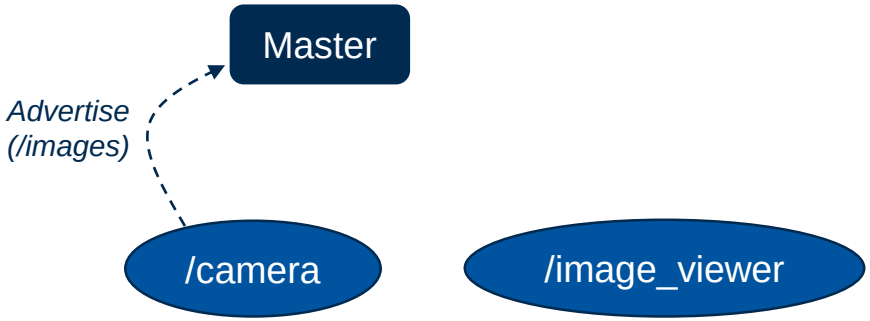
ROS Basics: Computation Graph Level

- The *ROS master* provides naming and registration services to the rest of the *nodes*
  - Tracks *publishers, subscribers, services*
- Enable individual ROS *nodes* to locate each other
  - Subsequent, they communicate peer-to-peer
- Provides the *Parameter Server*
- *Example: Two nodes: /camera and /image_viewer*



$ **roscore**

```
laptop@mobile-98:~$ roscore
... logging to /home/laptop/.ros/log/b3a4f352-1a27-11e7-8a27-
Checking log directory for disk usage. This may take awhile.
Press Ctrl-C to interrupt
Done checking log file disk usage. Usage is <1GB.

started roslaunch server http://mobile-98:44845/
ros_comm version 1.12.7


SUMMARY
========

PARAMETERS
 * /rosdistro: kinetic
 * /rosversion: 1.12.7

NODES

auto-starting new master
process[master]: started with pid [8919]
ROS_MASTER_URI=http://mobile-98:11311/

setting /run_id to b3a4f352-1a27-11e7-8a27-6067201943ac
process[rosout-1]: started with pid [8932]
started core service [/rosout]
```

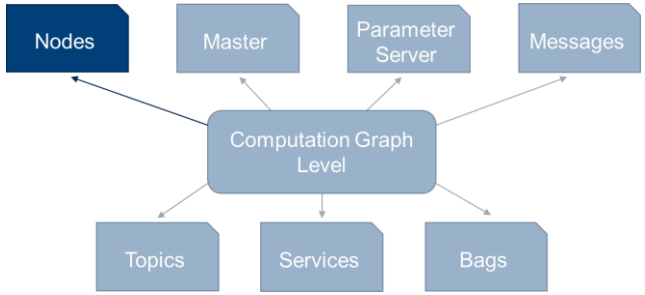


http://wiki.ros.org/ROS/Concepts : http://wiki.ros.org/Master

IfU IMA ZLW RWTH AACHEN UNIVERSITY

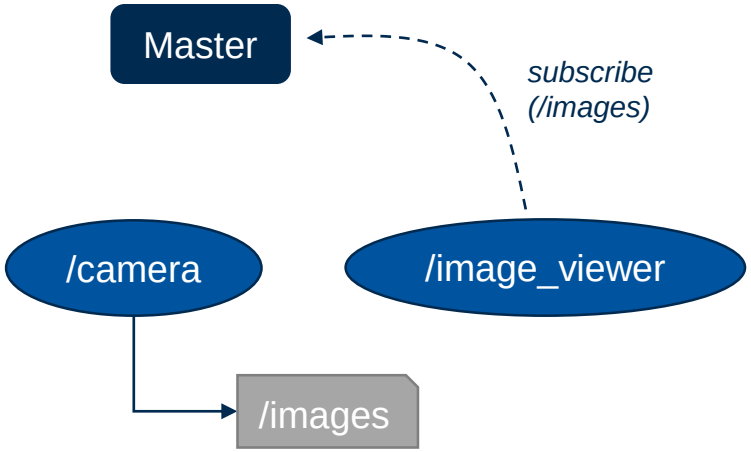
ROS Basics: Computation Graph Level

- The *ROS master* provides naming and registration services to the rest of the *nodes*
  - Tracks *publishers, subscribers, services*
- Enable individual ROS *nodes* to locate each other
  - Subsequent, they communicate peer-to-peer
- Provides the *Parameter Server*
- *Example: Two nodes: /camera and /image_viewer*




```
$ roscore
```

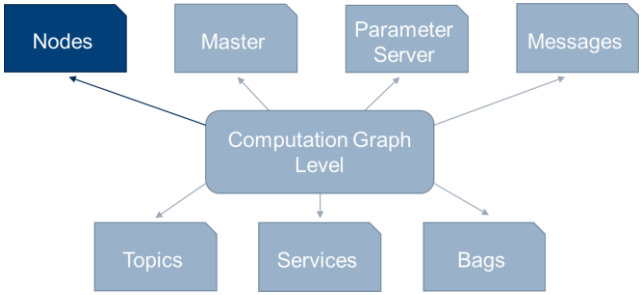
```
laptop@mobile-98:~$ roscore
... logging to /home/laptop/.ros/log/b3a4f352-1a27-11e7-8a27-
Checking log directory for disk usage. This may take awhile.
Press Ctrl-C to interrupt
Done checking log file disk usage. Usage is <1GB.

started roslaunch server http://mobile-98:44845/
ros_comm version 1.12.7


SUMMARY
========

PARAMETERS
 * /rosdistro: kinetic
 * /rosversion: 1.12.7

NODES

auto-starting new master
process[master]: started with pid [8919]
ROS_MASTER_URI=http://mobile-98:11311/

setting /run_id to b3a4f352-1a27-11e7-8a27-6067201943ac
process[rosout-1]: started with pid [8932]
started core service [/rosout]
```



http://wiki.ros.org/ROS/Concepts : http://wiki.ros.org/Master

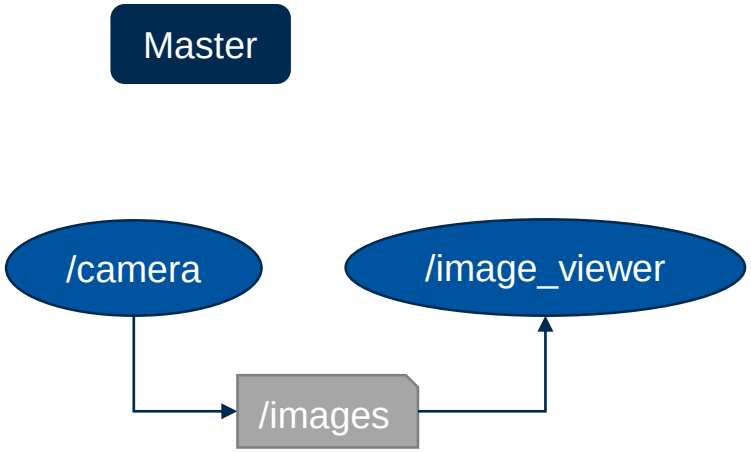
ROS Basics: Computation Graph Level

- The *ROS master* provides naming and registration services to the rest of the *nodes*
  - Tracks *publishers, subscribers, services*
- Enable individual ROS *nodes* to locate each other
  - Subsequent, they communicate peer-to-peer
- Provides the *Parameter Server*
- *Example: Two nodes: /camera and /image_viewer*




```
$ roscore
```

```
laptop@mobile-98:~$ roscore
... logging to /home/laptop/.ros/log/b3a4f352-1a27-11e7-8a27-
Checking log directory for disk usage. This may take awhile.
Press Ctrl-C to interrupt
Done checking log file disk usage. Usage is <1GB.

started roslaunch server http://mobile-98:44845/
ros_comm version 1.12.7


SUMMARY
========

PARAMETERS
 * /rosdistro: kinetic
 * /rosversion: 1.12.7

NODES

auto-starting new master
process[master]: started with pid [8919]
ROS_MASTER_URI=http://mobile-98:11311/

setting /run_id to b3a4f352-1a27-11e7-8a27-6067201943ac
process[rosout-1]: started with pid [8932]
started core service [/rosout]
```



http://wiki.ros.org/ROS/Concepts : http://wiki.ros.org/Master

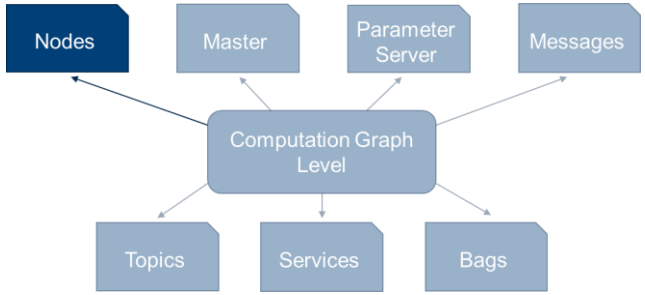
IfU IMA ZLW RWTH AACHEN UNIVERSITY

ROS Basics: Computation Graph Level

- *Nodes* communicating with each other by publishing **messages** to topics
- A **message** is a simple data structure
  - Comprising typed fields
  - Primitive types: integer, floating point, boolean, …
  - Arrays of primitive types
  - Arbitrarily nested structures and arrays
- *Nodes* can also exchange a request and response
  - Part of *ROS service call*
  - Defined in *srv files*
- Naming convention
  - [package name] + / + [name of .msg file]
  - E.g. *std_msgs/msg/String.msg*
- *MD5 checksum*
  - Versionization by MD5 sum of .msg file
  - Message type and MD5 sum have to match

```
5f3f794301c7af61b3beab5b9997bb64  PoseArray.msg
```

md5sum



Commands	Description
$ rosmsg show	show message description
$ rosmsg info	alias for rosmsg show
$ rosmsg list	list all messages
$ rosmsg md5	display message md5sum
$ rosmsg package	list messages in a package
$ rosmsg packages	list packages that contain messages

```
laptop@mobile-98:~$ rosmsg list
actionlib/TestAction
actionlib/TestActionFeedback
actionlib/TestActionGoal
actionlib/TestActionResult
actionlib/TestFeedback
geometry_msgs/Pose2D
geometry_msgs/PoseArray
geometry_msgs/PoseStamped
geometry_msgs/PoseWithCovariance
geometry_msgs/PoseWithCovarianceStamped
geometry_msgs/Quaternion
```

http://wiki.ros.org/ROS/Concepts : http://wiki.ros.org/Messages

ROS Basics: Computation Graph Level

- *Example message composition (geometry_msgs):*

Type: ***PoseArray.msg*** *contains Array of Pose*

```
laptop@mobile-98:/opt/ros/kinetic/share/geometry_msgs/msg$ cat PoseArray.msg
# An array of poses with a header for global reference.

Header header

Pose[] poses
```

Type: ***Pose.msg*** *contains Point & Quaternion*

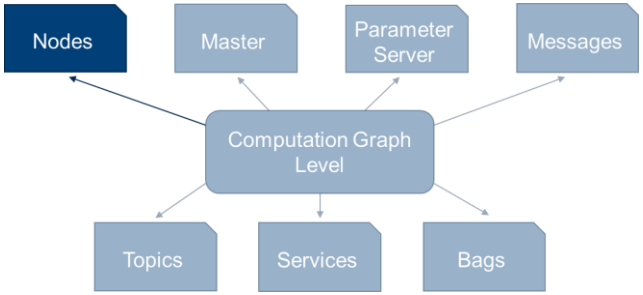
```
laptop@mobile-98:/opt/ros/kinetic/share/geometry_msgs/msg$ cat Pose.msg
# A representation of pose in free space, composed of position and orientation.
Point position
Quaternion orientation
```

Type: ***Point.msg*** *contains 3x float64*

```
laptop@mobile-98:/opt/ros/kinetic/share/geometry_msgs/msg$ cat Point.msg
# This contains the position of a point in free space
float64 x
float64 y
float64 z
```

Type: ***Quaternion.msg*** *contains 4x float64*

```
laptop@mobile-98:/opt/ros/kinetic/share/geometry_msgs/msg$ cat Quaternion.msg
# This represents an orientation in free space in quaternion form.

float64 x
float64 y
float64 z
float64 w
```



Commands	Description
$ rosmsg show	show message description
$ rosmsg info	alias for rosmsg show
$ rosmsg list	list all messages
$ rosmsg md5	display message md5sum
$ rosmsg package	list messages in a package
$ rosmsg packages	list packages that contain messages

```
laptop@mobile-98:~$ rosmsg list
actionlib/TestAction
actionlib/TestActionFeedback
actionlib/TestActionGoal
actionlib/TestActionResult
actionlib/TestFeedback
geometry_msgs/Pose2D
geometry_msgs/PoseArray
geometry_msgs/PoseStamped
geometry_msgs/PoseWithCovariance
geometry_msgs/PoseWithCovarianceStamped
geometry_msgs/Quaternion
geometry_msgs/QuaternionStamped
```

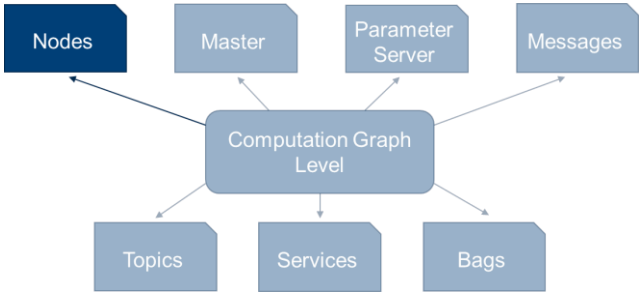
http://wiki.ros.org/ROS/Concepts : http://wiki.ros.org/Messages

IfU IMA ZLW RWTH AACHEN UNIVERSITY

ROS Basics: Computation Graph Level

- *Messages* are routed via transport system
  - *Publish / subscribe semantics*
  - *Decouples production of information from its consumption*
  - *Nodes are not awar of who they are communicating with*
- Publishing/Subscribing to *topics*
  - Nodes that are interested in data *subscribe*
  - Nodes that generate data *publish*
- Topic are intended for unidirectional, streaming communication
  - If you need request/response use ***services***
- Topic Types
  - *Each topic is strongly types by the ROS message*
  - *Master **does not** enforce type consistency*
- *Topic Transports*
  - ***TCPROS** (default): streams message data over persistent TCP/IP connections*
  - ***UDPROS**: separates messages into UDP packets*

http://wiki.ros.org/ROS/Concepts : http://wiki.ros.org/Topics

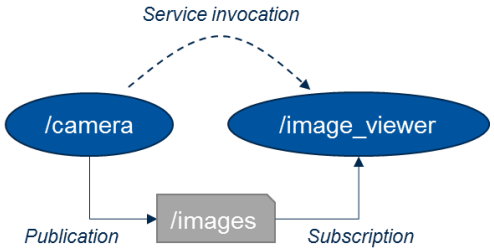


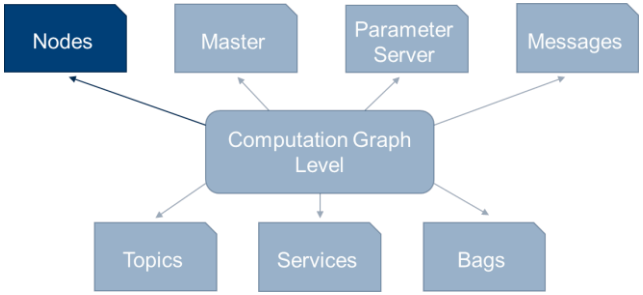
Commands	Description
$ rostopic bw	display bandwidth used by topic
$ rostopic delay	display delay of topic from timestamp in header
$ rostopic echo	print messages to screen
$ rostopic find	find topics by type
$ rostopic hz	display publishing rate of topic
$ rostopic info	print information about active topic
$ rostopic list	list active topics
$ rostopic pub	publish data to topic
$ rostopic type	print topic or field type

```
Type: turtlesim/Pose

Publishers:
 * /turtlesim (http://mobile-98:39731/)

Subscribers:
 * /rostopic_9360_1491414592736 (http://mobile-98:33807/)
```

ROS Basics: Computation Graph Level

- Publish/Subscribe is not appropriate for *RPC request/reply* interactions
  - Often required in distributed systems
- Request/reply
  - Is done via a service
  - Defined by **a pair** of messages
    - one for the request
    - one for the reply
- Services
  - Defined using **srv** files
  - Compiled into source code by a ROS client library
- Services have an associated service type
  - like topics
  - package resource name of the .srv file
- Versioned by a MD5 sum of the .srv file





Commands	Description
$ rosservice args	print service arguments
$ rosservice call	call the service with provided args
$ rosservice find	find services by service type
$ rosservice info	print information about service
$ rosservice list	list active services
$ rosservice type	print service type
$ rosservice uri	print service ROSRPC uri

```
laptop@mobile-98:~$ rosservice list
/clear
/kill
/reset
/rosout/get_loggers
/rosout/set_logger_level
/rostopic_9360_1491414592736/get_loggers
/rostopic_9360_1491414592736/set_logger_level
/spawn
/teleop_turtle/get_loggers
```

http://wiki.ros.org/ROS/Concepts : http://wiki.ros.org/Services

ROS Basics: Computation Graph Level

- Some examples of .srv files:

  Get a map by request:

```
laptop@mobile-98:/opt/ros/kinetic/share$ cat ./map_msgs/srv/GetPointMap.srv
# Get the map as a sensor_msgs/PointCloud2
---
sensor_msgs/PointCloud2 map
```

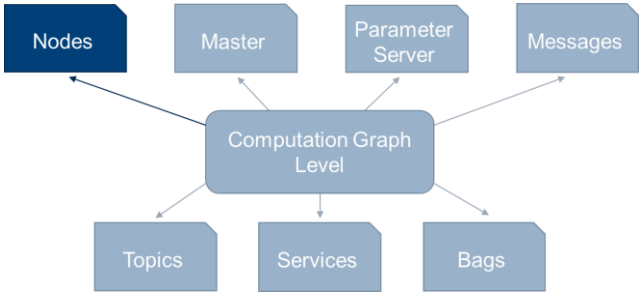
  Bilateral: setting and getting link states

  - get link state:

```
laptop@mobile-98:/opt/ros/kinetic/share$ cat ./gazebo_msgs/srv/GetLinkState.srv
string link_name          # name of link
                          # link names are prefixed by model name, e.g. pr2::base
string reference_frame    # reference frame of returned information, must be a va
                          # if empty, use inertial (gazebo world) frame
                          # reference_frame names are prefixed by model name, e.g
---
gazebo_msgs/LinkState link_state
bool success              # return true if get info is successful
string status_message     # comments if available
```

  - and set link state:

```
laptop@mobile-98:/opt/ros/kinetic/share$ cat ./gazebo_msgs/srv/SetLinkState.srv
gazebo_msgs/LinkState link_state
---
bool success                # return true if set wrench successful
string status_message       # comments if available
```



Commands	Description
$ rosservice args	print service arguments
$ rosservice call	call the service with provided args
$ rosservice find	find services by service type
$ rosservice info	print information about service
$ rosservice list	list active services
$ rosservice type	print service type
$ rosservice uri	print service ROSRPC uri

```
laptop@mobile-98:~$ rosservice list
/clear
/kill
/reset
/rosout/get_loggers
/rosout/set_logger_level
/rostopic_9360_1491414592736/get_loggers
/rostopic_9360_1491414592736/set_logger_level
/spawn
/teleop_turtle/get_loggers
```

http://wiki.ros.org/ROS/Concepts : http://wiki.ros.org/Services

ROS Basics: Computation Graph Level - Overview

- Quick overview of Graph Concepts

Concept	Description
Nodes	A node is an executable that uses ROS to communicate with other nodes.
Messages	ROS data type used when subscribing or publishing to a topic.
Topics	Nodes can *publish* messages to a topic as well as *subscribe* to a topic to receive messages.
Master	Name service for ROS (i.e. helps nodes find each other)
rosout	ROS equivalent of stdout/stderr
roscore	Master + rosout + parameter server (parameter server will be introduced later)

http://wiki.ros.org/ROS/Tutorials/UnderstandingNodes

There are much more to be done, if you are interested in.

http://wiki.ros.org/

Outline

Outline

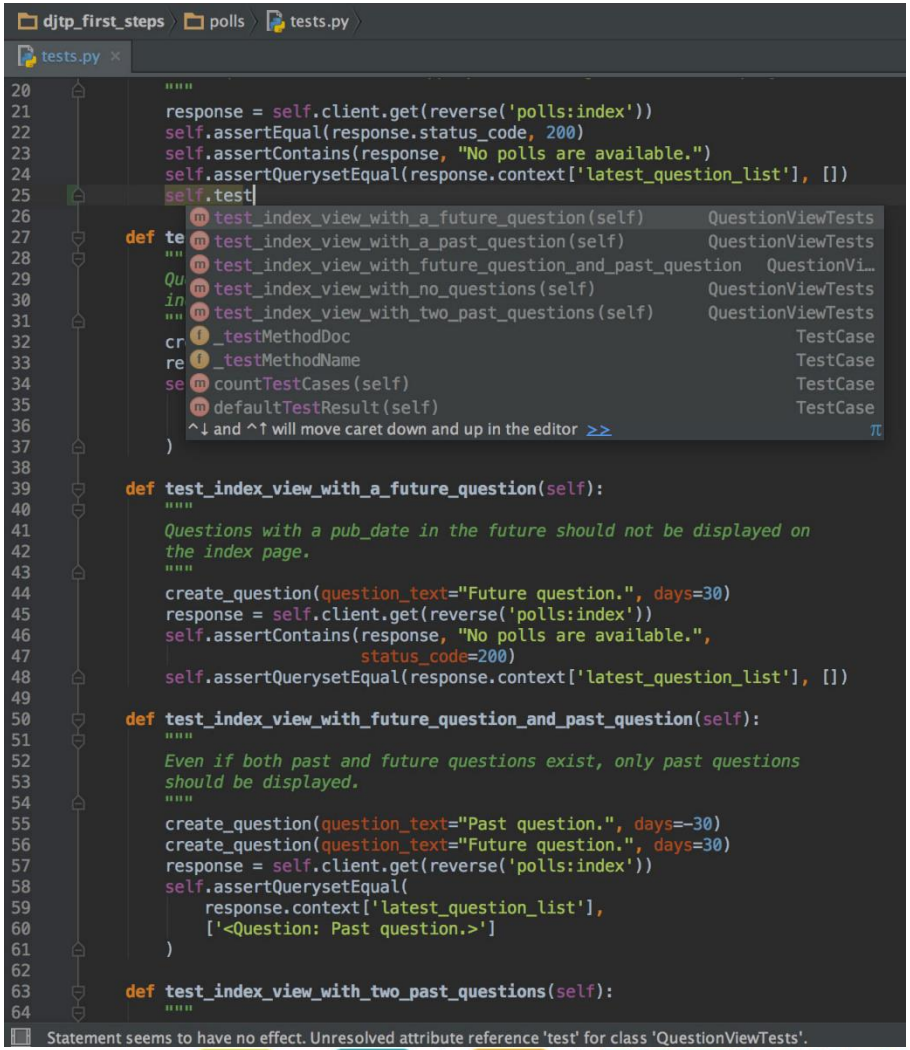
Outline

Practicle Unit: Your ROS Code in Python

Integrated Development Environment (IDE)

- An interactive Programming environment:
  - A programming language specified text editor
    - Smart tipping
    - Error checking
    - Keep your code cool
  - Builds automation tools
  - Debugger
- Who needs an IDE?

  People who are not programming **FREAKs**.



Practicle Unit: Your ROS Code in Python

Integrated Development Environment (IDE) : PyCharm

Outline

Practicle Unit: Your ROS Code in Python

Configure Your ROS Environment: Workspace

- Create a Workspace in ROS

- Further information: http://wiki.ros.org/catkin/Tutorials/create_a_workspace

```
mkdir -p ~/catkin_ws/src
cd ~/catkin_ws/
catkin_make
```

Outline

Outline

Programming your Motion Planning for Kinova

Motion Planning Interface for Python

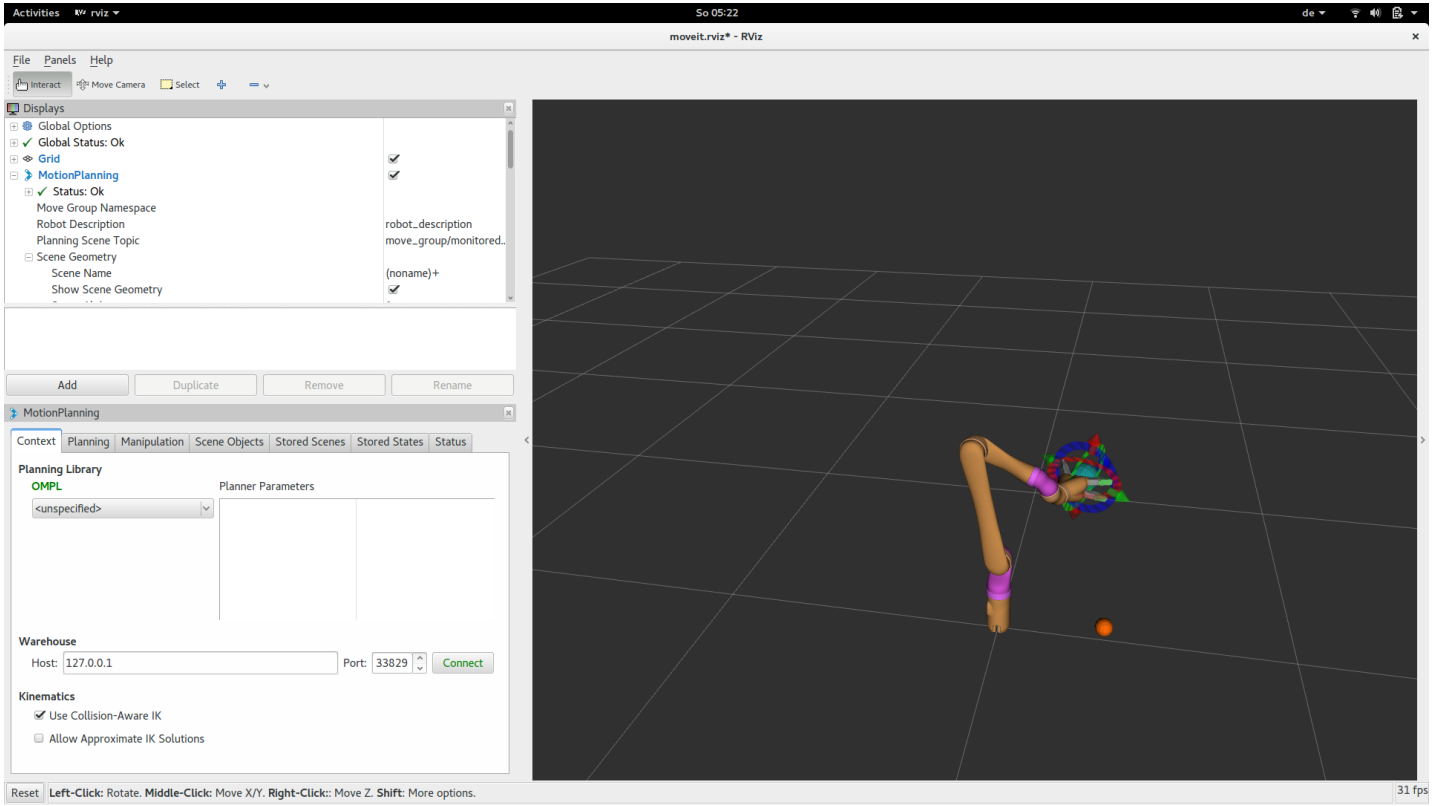
- Visit: http://docs.ros.org/kinetic/api/moveit_tutorials/html/doc/pr2_tutorials/planning/scripts/doc/move_group_python_interface_tutorial.html

Outline

Programming your Motion for Kinova

Mission Statement

- Let the robot move to the orange sphere
  - Use Motion Planer from previous section
  - `Program a Python Script for the Task`



Robot Operating System (ROS)

How to generate motion? => Use Moveit!

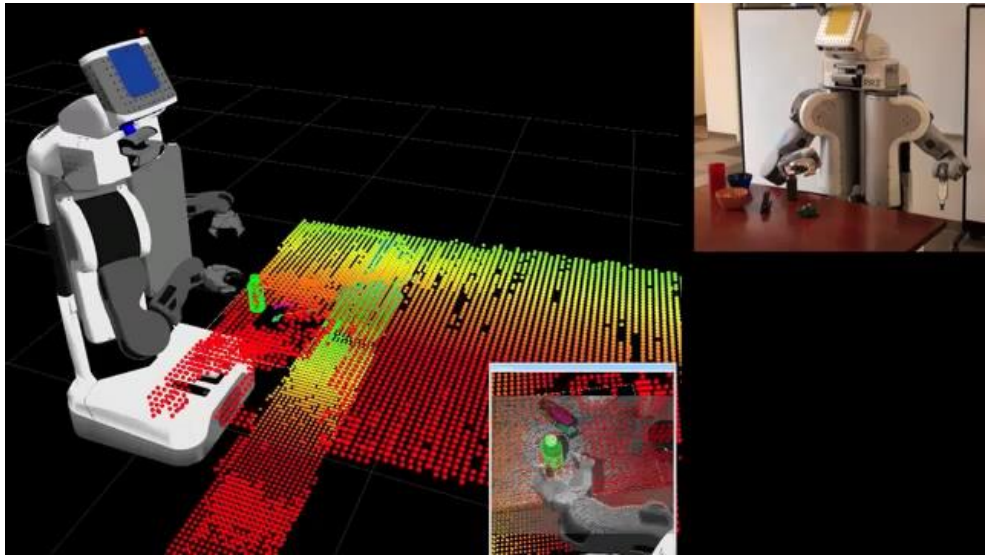
- Moveit is a user-friendly platform for building **flexible** industrial, research and commercial applications
  - Easy configuration, easy programming, quick switch-over
  - High performance

- Features
  - Motion planning
    - Navigation
    - Manipulation
  - Environment integration
    - 3D perception
    - Kinematics
    - control

Robot Operating System (ROS)

What does Moveit offer?

- Technical Capabilities
  - Collision checking: fast and flexible
  - Integrated kinematics
  - Motion Planning
    - Fast, good quality paths
    - Kinematic constraints
  - Standardized interfaces to controllers
  - Execution and monitoring



Programming your Motion for Kinova

Prerequirements

- Download and build our kinova-ros package:

1. Open a terminal (ctrl+T)
2. Enter line-by-line:

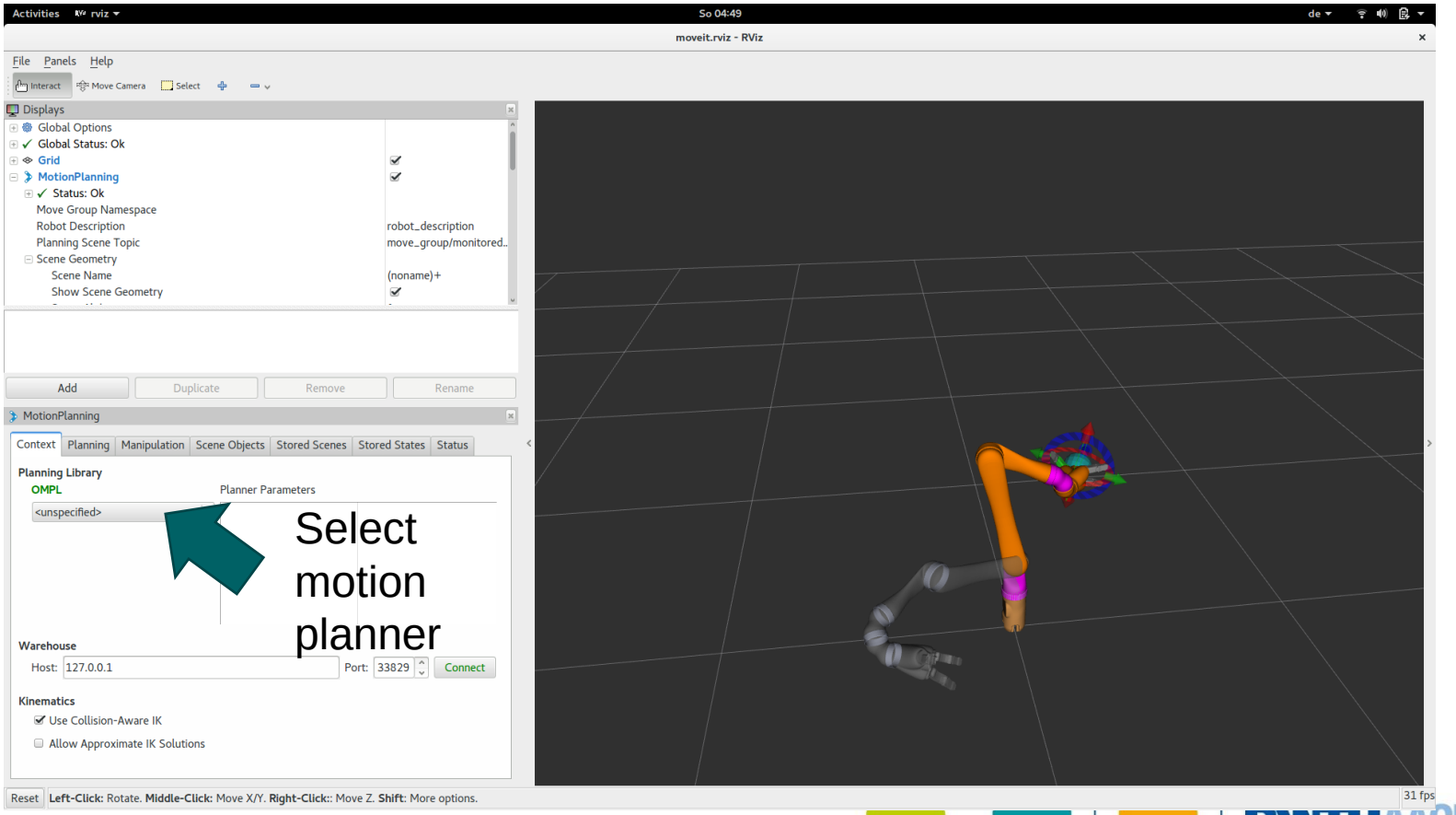
```
cd catkin_ws/src
git clone https://github.com/philippente/kinova-ros.git
cd ..
catkin_make
```

Programming your Motion for Kinova

Task: Open Moveit and let the robot move

- Open a terminal and enter

```
roslaunch j2n6s300_moveit_config demo.launch
```

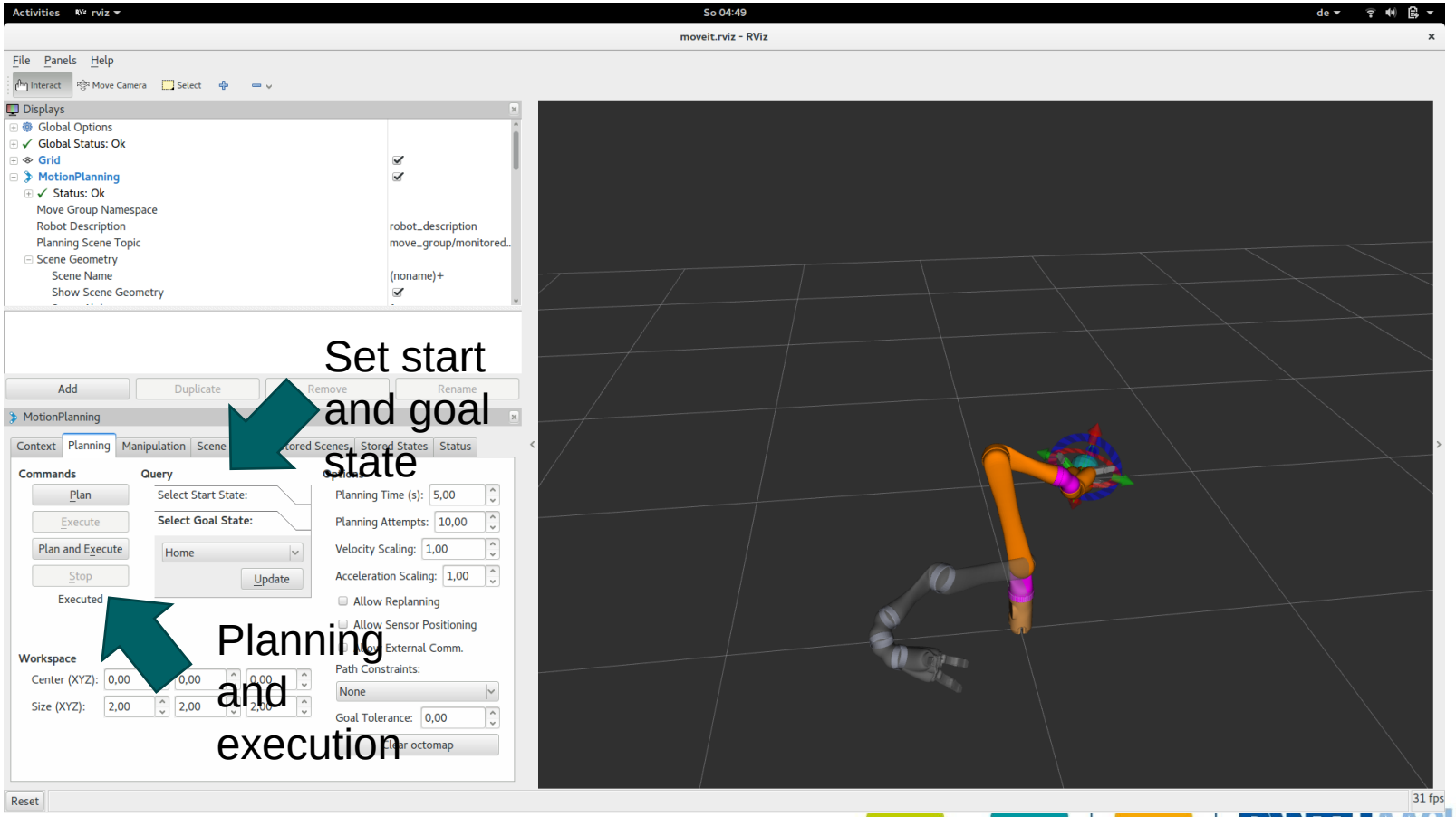


Programming your Motion for Kinova

Task: Open Moveit and let the robot move

- Open a terminal and enter

```
roslaunch j2n6s300_moveit_config demo.launch
```



Robotic framework : ROS : Moveit

Task: Adjust the motion

- Open `catkin_ws/src/kinova-ros/my_test_move/src/my_move.py` in PyCharm
  - Two kinds of motions are available: CartesianPath and PoseTarget
  - CartesianPath creates a linear motion
  - PoseTarget creates a point-to-point motion

- Scroll down to `if plan_type == "pose":`
  - Here a goal state is defined for PoseTarget
- Scroll down to `elif plan_type is "cartesian":`
  - Here a goal state is defined for CartesianPath

- **Task1:** Adjust the positions (`wpose` and `pose_target`) and see what will happen
  - 1. Load ROS libraries: `roslaunch j2n6s300_moveit_config demo.launch`
  - 2. Run motion prograyourm in different terminal: `rosrun my_test_move my_move.py`
- **Task2**: Adjust and add positions to reach the orange ball with the Kinova Jaco 2!

Outline

Programming your Motion Planning for Kinova

Practicle Unit: Get into the task

Backup: New Agenda