ROS Intro/Tutorial Contents

- Motivation
- ROS
  - Basic Structure, Commands etc.
- Installation
- Basic Workflow (incl. simple example project)
- Simulator: Stage
- Visualization: rViz
- Demos (Wavefront, Tangent bug, Potential)
- Debugging rospy in PyCharm

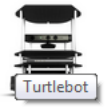
- ROS (Robot Operating System) provides libraries and tools to help software developers create robot applications. It provides hardware abstraction, device drivers, libraries, visualizers, message-passing, package management, and more.

Video: https://www.youtube.com/watch?v=PGaXiLZD2KQ

Supported Robots


Fraunhofer IPA Care-O-bot


Videre Erratic



TurtleBot


Aldebaran Nao


Lego NXT


Shadow Hand


Willow Garage PR2


iRobot Roomba

Robotnik Guardian


Merlin miabotPro


AscTec Quadrotor


CoroWare Corobot

Clearpath Robotics Husky

Clearpath Robotics Kingfisher


Festo Didactic Robotino


Gostai Jazz


Neobotix mp-500


Neobotix mpo-500


Neobotix mpo-700


ROS-Industrial

Robotnik Modular Arm

Robotnik Summit


Cyton-Gamma


Robonaut 2


Adept MobileRobots Pioneer family (P3DX, P3AT, PeopleBot, PowerBot, AmigoBot, PatrolBot, GuiaBot)


Adept MobileRobots Seekur family (Seekur, Seekur Jr.)


Adept MobileRobots Pioneer LX


Eddiebot

Allegro Hand SimLab


REEM


Kobuki


Komodo

Dr. Robot Jaguar


BipedRobin


WheeledRobin

Kawada Nextage / Hiro

Denso VS060


PAL Robotics REEM-C


jaco

Robotnik SummitXL

AMIGO

TUlip

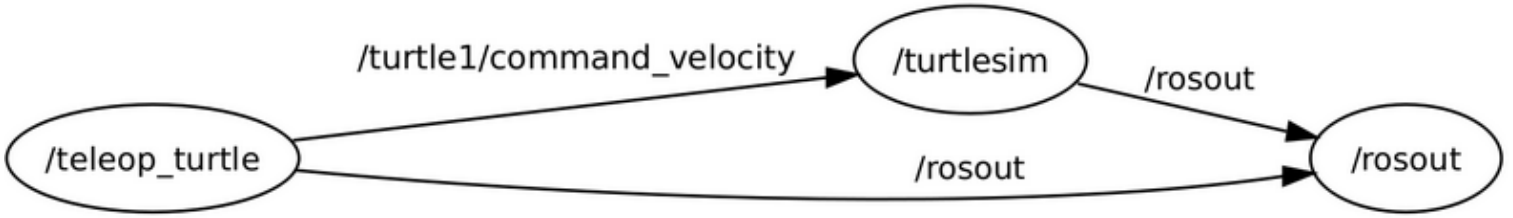
Supported Sensors

- Sharp IR range
- Hokuyo laser scanners
- Sick lasers
- Microsoft Kinect
- Asus Xtion
- Cameras
  - monocular and stereo
  - USB (uvc) and rewire
  - video streaming (gstreamer)

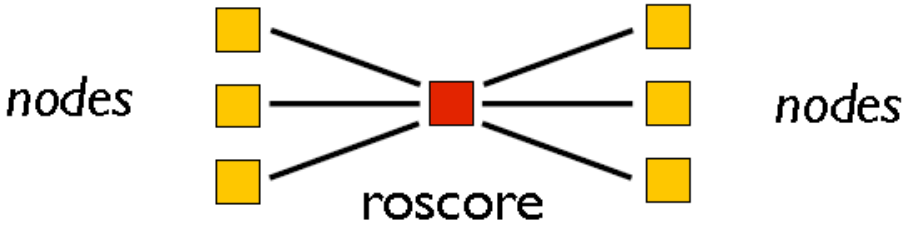
Structure of the ROS Framework

- Related to Filesystem (resources that you encounter on disk)
  - Packages
    - Main unit for organizing software in ROS
    - A folder (e.g. in `~/catkin_ws/src`) that contains your code, build files, launch files, etc.
    - Can contain any number of nodes
    - Should only contain code that is related
    - You will create a package to hand in the assignment
  - Package Manifests
    - Manifests (package.xml) provide metadata about a package, including its name, version, description, license information, dependencies.
    - You have to adjust this file eventually
  - See http://wiki.ros.org/ROS/Concepts for more details

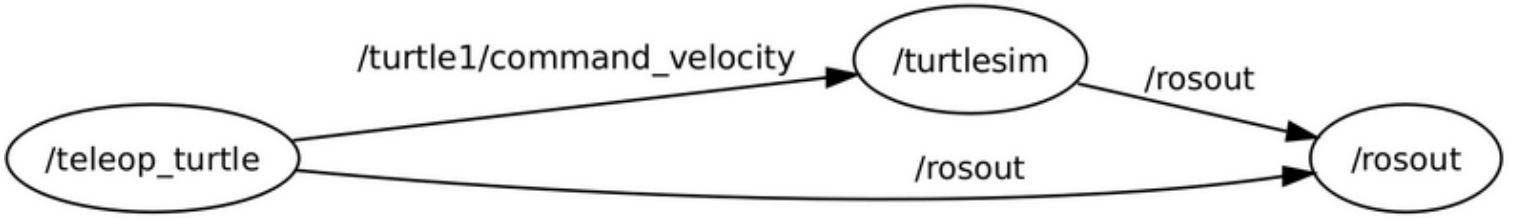
Structure of the ROS Framework



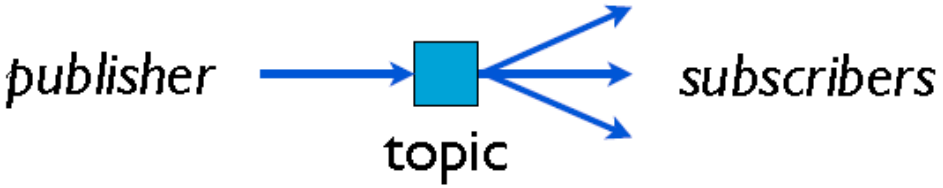
- Computation Graph Level
  - Nodes
    - Nodes are processes that perform computation
    - robot control system usually comprises many nodes
    - You will program nodes (in c++ or python)
  - Master
    - To start: `$ roscore`
    - Central unit
    - You always start working by starting the ROS Master

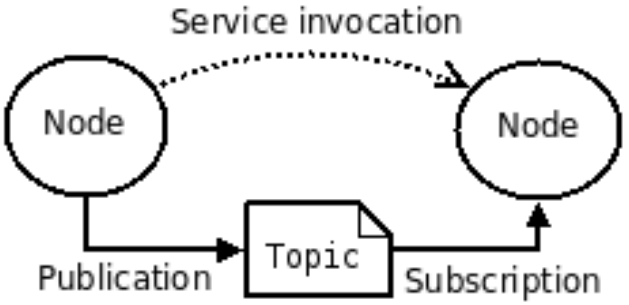


Structure of the ROS Framework



- Computation Graph Level (continued)
  - Messages
    - Nodes communicate with each other by passing messages
    - You have to deal with the different kinds of these
  - Topics
    - Messages are routed via a transport system with ***publish*** / ***subscribe*** semantics
    - You have to deal with this





Tutorials

1) Get Ubuntu!
2) Installation (jade)
3) Understanding Nodes
4) Understanding Topics
5) Basic Workflow
   a) Creating workspace with catkin
   b) Create a package (one for each exercise)
   c) Add/Edit source files of a package
   d) Build the workspace
   e) Load this workspace
   f) Run your package
   g) Create a launcher (to start multiple packages at once)
6) Using the Stage Simulator
7) Using rViz visualization

Getting Ubuntu

- Get VM (preferred) or install Ubuntu for dual boot
  - VM Virtualbox: http://www.oracle.com/technetwork/server-storage/virtualbox/downloads/index.html
- I suggest using Xubuntu, a lightweight flavour of Ubuntu that is similar to the windows 7 design https://xubuntu.org/getxubuntu/
  - “How to make Ubuntu Linux look like Windows 7” http://www.pcworld.com/article/2028896/how-to-make-ubuntu-linux-look-like-windows-7.html
  - The step ‘sudo apt-get install xubuntu-desktop’ can be skipped if you downloaded Xubuntu directly

ROS Installation (Jade)

- Get Ubuntu (14.04 or newer): http://www.ubuntu.com/download/desktop
- Windows → Oracle VM VirtualBox: http://www.virtualbox.org
- Follow Guide: http://wiki.ros.org/jade/Installation/Ubuntu
- Linux Terminal Code:

```
sudo apt-get install ros-jade-desktop-full
sudo rosdep init
rosdep update
echo "source /opt/ros/jade/setup.bash" >> ~/.bashrc
source ~/.bashrc
sudo apt-get install python-rosinstall
sudo apt-get install ros-jade-map-server
```

- ros-jade-map-server is optional
- For kinetic (newest ros): http://wiki.ros.org/kinetic/Installation/Ubuntu
- Restart the terminal after installation

Understanding Nodes

Prerequisites

```
sudo apt-get install ros-jade-ros-tutorials
```

Enter in terminal 1:

```
roscore
```

Enter in terminal 2:

```
rosnode list
```

Enter in terminal 3:

```
rosrun turtlesim turtlesim_node
```

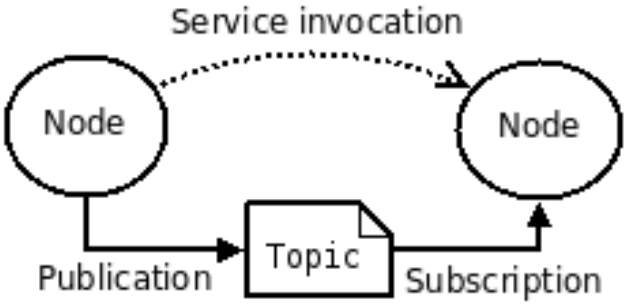
Enter in terminal 2:

```
rosnode list
```

Understanding Topics

Prerequisites

```
sudo apt-get install ros-jade-ros-tutorials
```



Enter in terminal 1:

```
roscore
```

Enter in terminal 2:

```
rosrun turtlesim turtlesim_node
```

Enter in terminal 3:

```
rosrun turtlesim turtle_teleop_key
```

Enter in terminal 4:

```
rosrun rqt_graph rqt_graph
```

Enter in terminal 5:

```
rostopic pub -1 /turtle1/cmd_vel geometry_msgs/Twist -- '[2.0, 0.0,
0.0]' '[0.0, 0.0, 1.8]'
```

Basic Workflow

1) Create a workspace (once)
2) Create a package (one for each exercise)
3) Add/Edit source files of a package (to solve the exercises)
4) Build the workspace (after you changed something)
5) Load the workspace (each time you restart the PC or something changes in the workspace)
6) Run your package (to see results)
7) Create a launcher (more convenient way to run your package)

Creating a Workspace with catkin

- Creating a workspace using catkin (you do this once)

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

Create a package

- Prerequisite: Valid catkin workspace
- Create one for each exercise
- Go to the src folder of your catkin workspace

```
cd ~/catkin_ws/src
```

- Create new packages (this generates the *manifest* and *CmakeList):*

```
catkin_create_pkg tarefa1 std_msgs rospy roscpp
catkin_create_pkg wavefront std_msgs rospy roscpp
catkin_create_pkg tarefa3 std_msgs rospy roscpp
```

Add/Edit source files of a package

- Prerequisites: Valid catkin workspace and one or more generated packages
- Your work directory is located inside the package, e.g. **`~/catkin_ws/src/tarefa1/src`**
- Here you add code files like “Stopper.cpp” or “run_stopper.cpp” and header files like “Stopper.h”
- Edit respective lines in ~/catkin_ws/src/tarefa1/`CmakeLists.txt`
  - `For the example add this two lines`

```
...
add_executable(stopper src/Stopper.cpp src/run_stopper.cpp)
target_link_libraries(stopper ${catkin_LIBRARIES})
```

Building a Workspace with catkin

- If you program in C++
- Build the workspace (you do this every time you change the code)
  - This builds your package(s) in `~/catkin_ws/src` to `~/catkin_ws/build`

```
cd ~/catkin_ws/

catkin_make
```

Load the Workspace

- Each time you restarted the PC
- Once before you start the ROS core/nodes

```
source ~/catkin_ws/devel/setup.bash
```

- Check if the path variable is set correctly now

```
echo $ROS_PACKAGE_PATH
```

- Should output

/home/*youruser*/catkin_ws/src:/opt/ros/jade/share:/opt/ros/jade/stacks

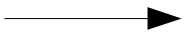
Run your package (to see results)

- To run your compiled code
- Prerequisites: Workspace, package + code, compiled, nothing ROS related is running
- Terminal 1:

```
roscore
```

- Terminal 2:

```
cd ~/catkin_ws
source ./devel/setup.bash
rosrun tarefa1 stopper
```



alex:catkin_ws$ rosrun tarefa1 stopper
[INFO] [1490198512.526653851]: Start moving

The simulator is missing, but the node works!

Create a launcher

- Launch file for launching both the Stage simulator and the stopper node
  - example for **~/catkin_ws/src/tarefa1/launch/my_launcher.launch**

```
<launch>

  <node name="stage" pkg="stage_ros" type="stageros" args="$(find stage_ros)/world/willow-
erratic.world"/>

  <node name="stopper" pkg="tarefa1" type="stopper" output="screen"/>

</launch>
```

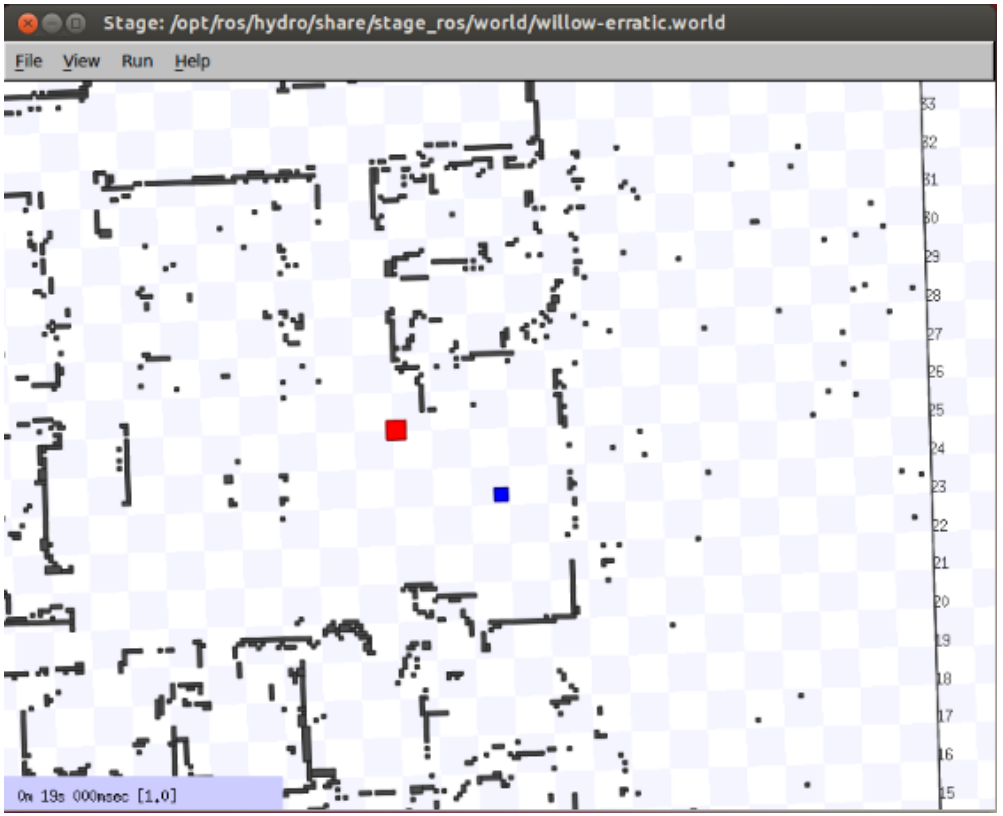
- To execute in a free terminal:

```
source ~/catkin_ws/devel/setup.bash

roslaunch tarefa1 my_launcher.launch
```

Stage Simulator

- http://wiki.ros.org/stage
- A 2D simulator that provides a virtual world populated by mobile robots, along with various objects for the robots to sense and manipulate



Stage Simulator

- Is installed with ROS jade
- To run Stage with an existing world file:

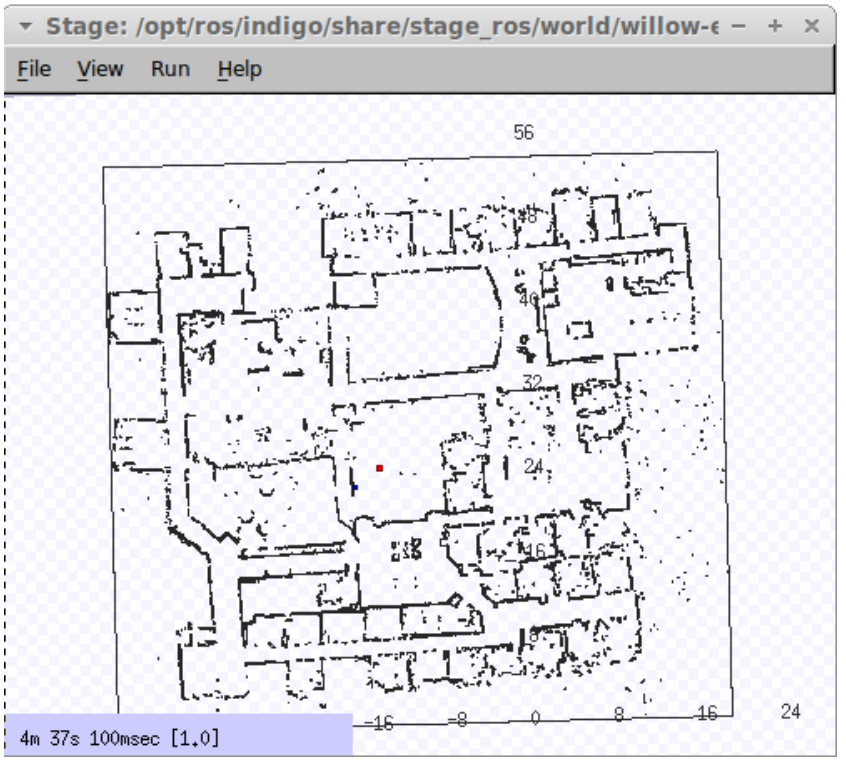
```
rosrun stage_ros stageros $(rospack find stage_ros)/world/willow-erratic.world
```

- Stage World Files:
  - `nameofworld.pgm` contains **occupancy grid** (image)
  - `myworld.world` contains **description** of entities
- Good class on the stage simulator

http://u.cs.biu.ac.il/~yehoshr1/89-685/ROS_Lesson4.pdf

.world File Example

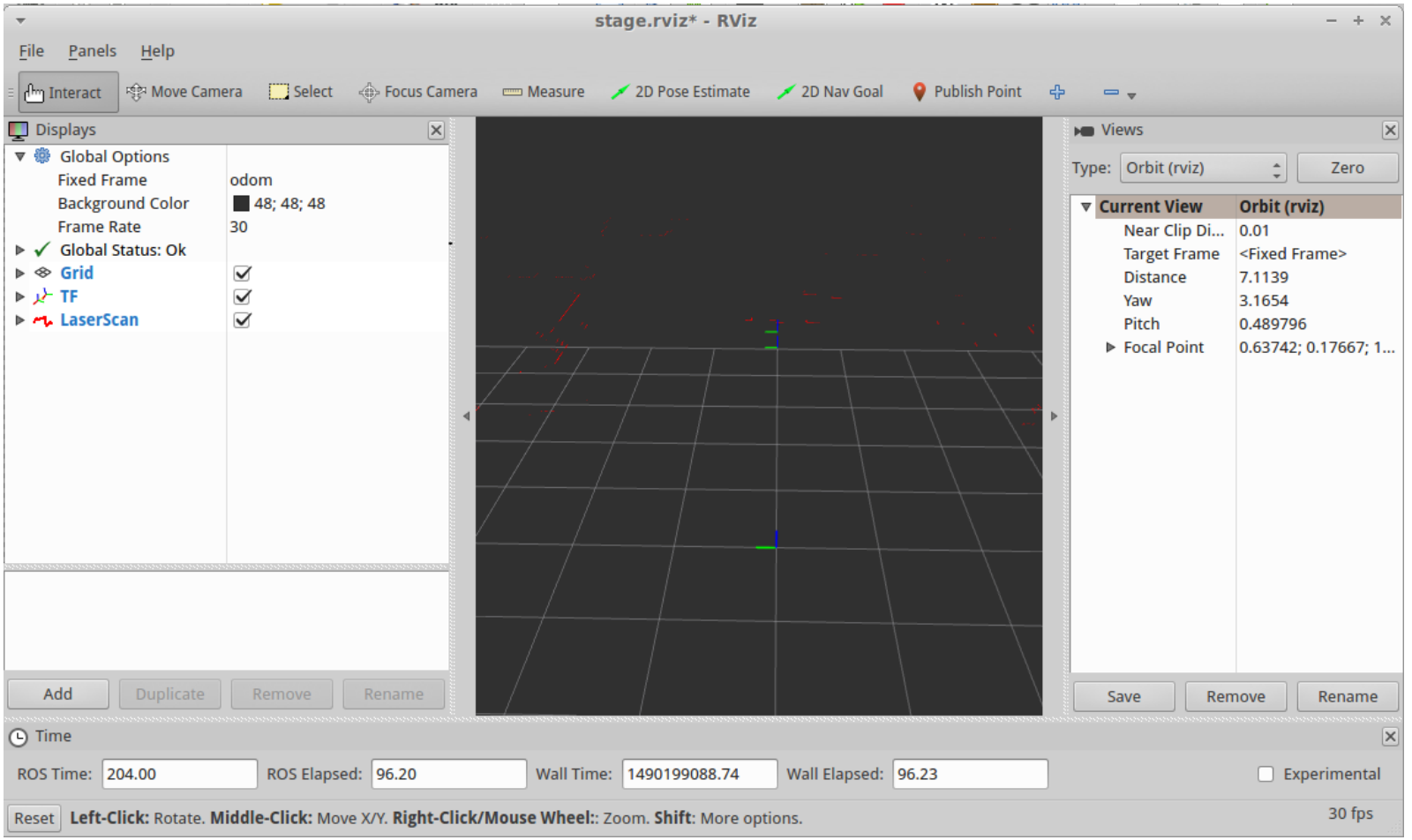
- You need to create your own, more simple world
- See an example in the folder `/opt/ros/jade/share/stage_ros/world/`
  - willow-full.pgm
  - willow-erratic.world



rViz Visualisation

- To see what the robot 'sees'

```
rosrun rviz rviz -d `rospack find stage_ros`/rviz/stage.rviz
```



Demo of Exercises

- Wavefront
- Tangent Bug
- Potential

Python and ROS

- Prerequisites: Valid catkin workspace and one or more generated packages
- Create a folder in the package directory, e.g. ‘tarefa1’

```
mkdir ~/catkin_ws/src/tarefa1/scripts
```

- Generate Python code in that ‘scripts’ folder, e.g. ‘listener.py’
  - And make the python file(s) executable, e.g.: `chmod +x listener.py`
- Compile the folder anyways (indirectly necessary) through `catkin_make` as described earlier
- Execute in an initialized terminal
  - Check first your terminal: `echo $ROS_PACKAGE_PATH`
  - Should output:
    /home/*youruser*/catkin_ws/src:/opt/ros/jade/share:/opt/ros/jade/stacks
  - Run: `rosrun tarefa1 listener.py`

Debugging Python PyCharm with ROS

- I suggest PyCharm, free and highly configurable
  - https://www.jetbrains.com/pycharm/download/#section=linux
  - The Professional version is 12 months free for students (I use it myself)
- PyCharm needs the environment variables from the workspace and ROS to function correctly
- To start PyCharm correctly with the environment variables:

```
source /opt/ros/indigo/setup.bash
source ~/catkin_ws/devel/setup.bash
pycharm-community
```

- Create a new python 2.7 project inside the ‘scripts’ folder (slide before) of the respective package
- Run/Debug your *.ppy files within the IDE

Useful Links

- ROS wiki http://wiki.ros.org/
- Search the ROS wiki for: "concepts", "command line tools", …
- ROS Q&A Forum http://answers.ros.org/questions/
- ROS code repositories at github https://github.com/ros