

# Robot Programming with Lisp

## 1. Introduction, Setup

Gayane Kazhoyan, Benjamin Brieber

Institute for Artificial Intelligence  
Universität Bremen

5<sup>th</sup> April, 2016

---

# Outline

Introduction

Assignment

Introduction

Assignment

Gayane Kazhoyan, Benjamin Brieber

5<sup>th</sup> April, 2016

Robot Programming with Lisp

2

# General Info

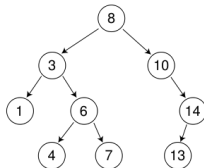
- Lecturers: Gaya, Benjamin (PhD students at IAI)
- Correspondence: `gaya@cs.uni-bremen.de`, `bbrieber@cs.uni-bremen.de`
- Dates: Tuesdays, 16:15 - 17:45
- Language: English (and German)
- Credits: 4 ECTS (2 SWS)
- Course type: practical course
- Course number: 03-BE-710.98d
- Location: TAB Building, Room 0.31 EG

# Course content

## Common Lisp



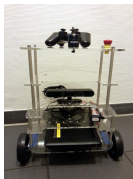
## Artificial Intelligence



## Robot Operating System (ROS)



## Robot platform





# Common Lisp

- Full-featured industry-standard programming language

# Common Lisp

- Full-featured industry-standard programming language
- Means for functional programming
- Means for imperative programming
- Means for OOP

# Common Lisp

- Full-featured industry-standard programming language
- Means for functional programming
- Means for imperative programming
- Means for OOP
- Fast prototyping through read-eval-print loop and dynamic typing

# Common Lisp

- Full-featured industry-standard programming language
- Means for functional programming
- Means for imperative programming
- Means for OOP
- Fast prototyping through read-eval-print loop and dynamic typing
- Compiles into machine code

# Common Lisp

- Full-featured industry-standard programming language
- Means for functional programming
- Means for imperative programming
- Means for OOP
- Fast prototyping through read-eval-print loop and dynamic typing
- Compiles into machine code
- Best choice for symbolic processing (AI, theorem proving, etc.)

# Common Lisp

- Full-featured industry-standard programming language
- Means for functional programming
- Means for imperative programming
- Means for OOP
- Fast prototyping through read-eval-print loop and dynamic typing
- Compiles into machine code
- Best choice for symbolic processing (AI, theorem proving, etc.)
- Good choice for writing domain-specific programming languages (e.g., robot programming languages)

# Common Lisp

- Full-featured industry-standard programming language
- Means for functional programming
- Means for imperative programming
- Means for OOP
- Fast prototyping through read-eval-print loop and dynamic typing
- Compiles into machine code
- Best choice for symbolic processing (AI, theorem proving, etc.)
- Good choice for writing domain-specific programming languages (e.g., robot programming languages)

Applications using / written in dialects of Lisp: Emacs, AutoCAD, Mirai, Google ITA, DART, Maxima, AI and robotics frameworks, ...

# ROS

- Middleware for communication of the components of a robotic system



# ROS

- Middleware for communication of the components of a robotic system
- Meta-Operating System for programming robotics software (configuring, starting / stopping, logging etc. software components)

# ROS

- Middleware for communication of the components of a robotic system
- Meta-Operating System for programming robotics software (configuring, starting / stopping, logging etc. software components)
- Powerful build system (based on CMake), with a strong focus on integration and documentation

# ROS

- Middleware for communication of the components of a robotic system
- Meta-Operating System for programming robotics software (configuring, starting / stopping, logging etc. software components)
- Powerful build system (based on CMake), with a strong focus on integration and documentation
- Language-independent architecture (C++, Python, Lisp, Java, JavaScript, ...)

# ROS

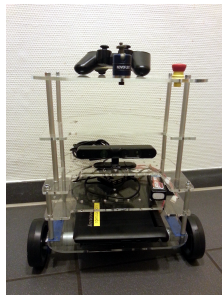
- Middleware for communication of the components of a robotic system
- Meta-Operating System for programming robotics software (configuring, starting / stopping, logging etc. software components)
- Powerful build system (based on CMake), with a strong focus on integration and documentation
- Language-independent architecture (C++, Python, Lisp, Java, JavaScript, ...)
- According to ROS 2014 Community Metrics Report,
  - More than 1 million pageviews of `wiki.ros.org` a month
  - About 3.5 million downloads of `.deb` packages a month

# ROS

- Middleware for communication of the components of a robotic system
- Meta-Operating System for programming robotics software (configuring, starting / stopping, logging etc. software components)
- Powerful build system (based on CMake), with a strong focus on integration and documentation
- Language-independent architecture (C++, Python, Lisp, Java, JavaScript, ...)
- According to ROS 2014 Community Metrics Report,
  - More than 1 million pageviews of `wiki.ros.org` a month
  - About 3.5 million downloads of `.deb` packages a month
- *De facto* standard in modern robotics

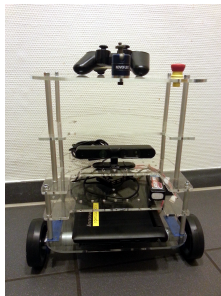
# TortugaBot

- 2 controllable wheels



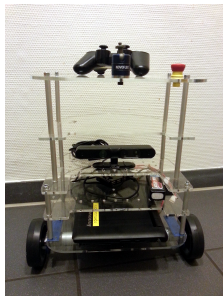
# TortugaBot

- 2 controllable wheels
- 2.5D vision sensor



# TortugaBot

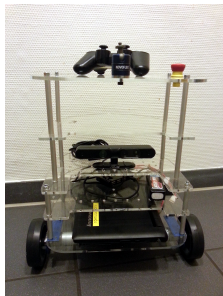
- 2 controllable wheels
- 2.5D vision sensor
- Asus Eee PC with bluetooth





# TortugaBot

- 2 controllable wheels
- 2.5D vision sensor
- Asus Eee PC with bluetooth
- Optional basket in the top part



# Why Lisp for robots?

- ROS supports a number of languages: C++, Python, Lisp and Java

# Why Lisp for robots?

- ROS supports a number of languages: C++, Python, Lisp and Java
- Lisp is good for rapid prototyping

# Why Lisp for robots?

- ROS supports a number of languages: C++, Python, Lisp and Java
- Lisp is good for rapid prototyping
- It is more suitable for symbolic reasoning and AI

# Why Lisp for robots?

- ROS supports a number of languages: C++, Python, Lisp and Java
- Lisp is good for rapid prototyping
- It is more suitable for symbolic reasoning and AI
- There are existing robot programming languages in Lisp that automate decision making

# Rough schedule

- Introduction, Setup
  
  - Lisp basics
  - Functional programming
  - OOP

---

  - ROS, ROS Lisp API (*roslisp*)
  - *roslisp*, 2D world of *turtlesim*
  - coordinate frames, *tf*
- TortugaBot, navigation
  - Collision avoidance
  - Project scenario

---

  - Project
  - *Lab visit*, project
  - The big day: **competition**

# Software requirements

Bringing a *personal laptop* is encouraged.

OS:	Ubuntu 14.04 LTS (other Ubuntu versions might work but with no guarantee)
IDE:	Emacs 24
Version control:	Git
Packaging system:	ROS
Lisp software:	SBCL compiler, ASDF build system, Emacs plugin for Common Lisp

# Homework assignments and the project

- Homework assignments will mostly consist of filling in the missing gaps in the already existing code.



# Homework assignments and the project

- Homework assignments will mostly consist of filling in the missing gaps in the already existing code.
- That code will be hosted on GitHub (<https://github.com>) and university GitLab (<https://gitlab.informatik.uni-bremen.de/>).

# Homework assignments and the project

- Homework assignments will mostly consist of filling in the missing gaps in the already existing code.
- That code will be hosted on GitHub (<https://github.com>) and university GitLab (<https://gitlab.informatik.uni-bremen.de/>).
- The code you write should be uploaded to GitHub / GitLab as well.

# Homework assignments and the project

- Homework assignments will mostly consist of filling in the missing gaps in the already existing code.
- That code will be hosted on GitHub (<https://github.com>) and university GitLab (<https://gitlab.informatik.uni-bremen.de/>).
- The code you write should be uploaded to GitHub / GitLab as well.
- Homework is due in one week, one more week gives 50% penalty

# Homework assignments and the project

- Homework assignments will mostly consist of filling in the missing gaps in the already existing code.
- That code will be hosted on GitHub (<https://github.com>) and university GitLab (<https://gitlab.informatik.uni-bremen.de/>).
- The code you write should be uploaded to GitHub / GitLab as well.
- Homework is due in one week, one more week gives 50% penalty
- Course final grade = 50 points homework + 50 points group project.

# Homework assignments and the project

- Homework assignments will mostly consist of filling in the missing gaps in the already existing code.
- That code will be hosted on GitHub (<https://github.com>) and university GitLab (<https://gitlab.informatik.uni-bremen.de/>).
- The code you write should be uploaded to GitHub / GitLab as well.
- Homework is due in one week, one more week gives 50% penalty
- Course final grade = 50 points homework + 50 points group project.
- To participate in the project you need at least 20 points from the homeworks, otherwise it's a fail.

# Homework assignments and the project

- Homework assignments will mostly consist of filling in the missing gaps in the already existing code.
- That code will be hosted on GitHub (<https://github.com>) and university GitLab (<https://gitlab.informatik.uni-bremen.de/>).
- The code you write should be uploaded to GitHub / GitLab as well.
- Homework is due in one week, one more week gives 50% penalty
- Course final grade = 50 points homework + 50 points group project.
- To participate in the project you need at least 20 points from the homeworks, otherwise it's a fail.
- Final grade: 50 points - 4.0, 100 points - 1.0

# Homework assignments and the project

- Homework assignments will mostly consist of filling in the missing gaps in the already existing code.
- That code will be hosted on GitHub (<https://github.com>) and university GitLab (<https://gitlab.informatik.uni-bremen.de/>).
- The code you write should be uploaded to GitHub / GitLab as well.
- Homework is due in one week, one more week gives 50% penalty
- Course final grade = 50 points homework + 50 points group project.
- To participate in the project you need at least 20 points from the homeworks, otherwise it's a fail.
- Final grade: 50 points - 4.0, 100 points - 1.0
- Bonus points for very good homework solutions

# Bottom line

You will learn / improve your skills in the following:

- Linux
- Git
- Emacs
- Functional programming
- Common Lisp, of course
- ROS (for future roboticists)

...and get to play with a real little robot!



# Outline

Introduction

Assignment

Introduction

Assignment

# Assignment goals

Set up your working environment    Set up your GitHub account



Get comfortable with Emacs



## Task 1: Install Ubuntu 14.04

- Find out your processor architecture (32 vs. 64 bit).

*Hint:* unless your computer is very old, it's most likely 64 bit.

In Windows 8-, holding the Windows Key press R, type dxdiag, press Enter and find the info you need.

## Task 1: Install Ubuntu 14.04

- Find out your processor architecture (32 vs. 64 bit).  
*Hint:* unless your computer is very old, it's most likely 64 bit.  
In Windows 8-, holding the Windows Key press R, type dxdiag, press Enter and find the info you need.
- Download Ubuntu 14.04 installation .iso:  
<http://www.ubuntu.com/download/desktop>

## Task 1: Install Ubuntu 14.04

- Find out your processor architecture (32 vs. 64 bit).  
*Hint:* unless your computer is very old, it's most likely 64 bit.  
In Windows 8-, holding the Windows Key press R, type dxdiag, press Enter and find the info you need.
- Download Ubuntu 14.04 installation .iso:  
<http://www.ubuntu.com/download/desktop>
- Burn the .iso onto a DVD or create a boot USB.  
*Hint:* for a bootable USB, in Windows use the Universal USB installer:  
<http://www.pendrivelinux.com/universal-usb-installer-easy-as-1-2-3/>;  
and in Linux you could, e.g., use the `unetbootin`.

# Task 1: Install Ubuntu 14.04

- Find out your processor architecture (32 vs. 64 bit).  
*Hint:* unless your computer is very old, it's most likely 64 bit.  
In Windows 8-, holding the Windows Key press R, type dxdiag, press Enter and find the info you need.
- Download Ubuntu 14.04 installation .iso:  
<http://www.ubuntu.com/download/desktop>
- Burn the .iso onto a DVD or create a boot USB.  
*Hint:* for a bootable USB, in Windows use the Universal USB installer:  
<http://www.pendrivelinux.com/universal-usb-installer-easy-as-1-2-3/>;  
and in Linux you could, e.g., use the `unetbootin`.
- Install Ubuntu 14.04 (aka Trusty).  
Dual boot installation with default settings is a one click thing.

# Task 1: Install Ubuntu 14.04: FAQ

- *How do I boot from CD?*

You should enter either “Boot Menu” or “BIOS Menu” during reboot

<https://www.desertcrystal.com/bootkeys>

## Task 1: Install Ubuntu 14.04: FAQ

- *How do I boot from CD?*

You should enter either “Boot Menu” or “BIOS Menu” during reboot

<https://www.desertcrystal.com/bootkeys>

- *Windows 8+ doesn't let me into “BIOS Menu”!*

You should restart into the “Boot Options Menu” of your Windows:  
hold down “Shift” while pressing “Restart”.

<http://www.makeuseof.com/tag/how-to-access-the-bios-on-a-windows-8-computer/>



# Task 1: Install Ubuntu 14.04: FAQ

- *How do I boot from CD?*

You should enter either “Boot Menu” or “BIOS Menu” during reboot

<https://www.desertcrystal.com/bootkeys>

- *Windows 8+ doesn't let me into “BIOS Menu”!*

You should restart into the “Boot Options Menu” of your Windows:  
hold down “Shift” while pressing “Restart”.

<http://www.makeuseof.com/tag/how-to-access-the-bios-on-a-windows-8-computer/>

- *My BIOS supports UEFI, Ubuntu won't install!*

It should work but if you can't get it to run turn off the UEFI mode:  
restart into the “Boot Options Menu” of your Windows,  
choose “Troubleshoot”, then “UEFI Firmware Settings”

<http://www.makeuseof.com/tag/how-to-access-the-bios-on-a-windows-8-computer/>

# Task 1: Install Ubuntu 14.04: FAQ

- *How do I boot from CD?*

You should enter either “Boot Menu” or “BIOS Menu” during reboot

<https://www.desertcrystal.com/bootkeys>

- *Windows 8+ doesn't let me into “BIOS Menu”!*

You should restart into the “Boot Options Menu” of your Windows:  
hold down “Shift” while pressing “Restart”.

<http://www.makeuseof.com/tag/how-to-access-the-bios-on-a-windows-8-computer/>

- *My BIOS supports UEFI, Ubuntu won't install!*

It should work but if you can't get it to run turn off the UEFI mode:  
restart into the “Boot Options Menu” of your Windows,  
choose “Troubleshoot”, then “UEFI Firmware Settings”

<http://www.makeuseof.com/tag/how-to-access-the-bios-on-a-windows-8-computer/>

- *It still doesn't work!*

Write an email to [gaya@cs.uni-bremen.de](mailto:gaya@cs.uni-bremen.de)

## Task 2: Install ROS

Consult the official installation instructions for troubleshooting:  
<http://wiki.ros.org/indigo/Installation/Ubuntu>

In short, it boils down to executing the following in the terminal  
(*hint*: to open a fresh terminal press <Ctrl>+<Alt>+t):

- Add ROS repositories to your sources list:

```
sudo sh -c 'echo "deb http://packages.ros.org/ros/ubuntu trusty main" > /etc/apt/sources.list.d/ros-latest.list'
```

## Task 2: Install ROS

Consult the official installation instructions for troubleshooting:  
<http://wiki.ros.org/indigo/Installation/Ubuntu>

In short, it boils down to executing the following in the terminal (*hint*: to open a fresh terminal press <Ctrl>+<Alt>+t):

- Add ROS repositories to your sources list:

```
sudo sh -c 'echo "deb http://packages.ros.org/ros/ubuntu trusty main" > /etc/apt/sources.list.d/ros-latest.list'
```

- Add their key to your trusted public keys:

```
sudo apt-key adv --keyserver hkp://pool.sks-keyservers.net --recv-key 0xB01FA116
```

## Task 2: Install ROS

Consult the official installation instructions for troubleshooting:  
<http://wiki.ros.org/indigo/Installation/Ubuntu>

In short, it boils down to executing the following in the terminal  
(*hint*: to open a fresh terminal press <Ctrl>+<Alt>+t):

- Add ROS repositories to your sources list:

```
sudo sh -c 'echo "deb http://packages.ros.org/ros/ubuntu trusty main" > /etc/apt/sources.list.d/ros-latest.list'
```

- Add their key to your trusted public keys:

```
sudo apt-key adv --keyserver hkp://pool.sks-keyservers.net --recv-key 0xB01FA116
```

- Update your Debian package index:

```
sudo apt-get update
```

## Task 2: Install ROS

Consult the official installation instructions for troubleshooting:  
<http://wiki.ros.org/indigo/Installation/Ubuntu>

In short, it boils down to executing the following in the terminal (*hint*: to open a fresh terminal press <Ctrl>+<Alt>+t):

- Add ROS repositories to your sources list:

```
sudo sh -c 'echo "deb http://packages.ros.org/ros/ubuntu trusty main" > /etc/apt/sources.list.d/ros-latest.list'
```

- Add their key to your trusted public keys:

```
sudo apt-key adv --keyserver hkp://pool.sks-keyservers.net --recv-key 0xB01FA116
```

- Update your Debian package index:

```
sudo apt-get update
```

- The version of ROS distributed with Ubuntu 14.04 is **ROS Indigo**.  
Install the **desktop** package. Say <No> if asked about hddtemp.

```
sudo apt-get install ros-indigo-desktop
```

## Task 2: Install ROS

Consult the official installation instructions for troubleshooting:  
<http://wiki.ros.org/indigo/Installation/Ubuntu>

In short, it boils down to executing the following in the terminal (*hint*: to open a fresh terminal press <Ctrl>+<Alt>+t):

- Add ROS repositories to your sources list:

```
sudo sh -c 'echo "deb http://packages.ros.org/ros/ubuntu trusty main" > /etc/apt/sources.list.d/ros-latest.list'
```

- Add their key to your trusted public keys:

```
sudo apt-key adv --keyserver hkp://pool.sks-keyservers.net --recv-key 0xB01FA116
```

- Update your Debian package index:

```
sudo apt-get update
```

- The version of ROS distributed with Ubuntu 14.04 is **ROS Indigo**.  
Install the **desktop** package. Say <No> if asked about hddtemp.

```
sudo apt-get install ros-indigo-desktop
```

- Install the workspace management tools:

```
sudo apt-get install python-rosinstall && sudo apt-get install python-wstool
```

## Task 3: Setup ROS

Consult the official installation instructions for troubleshooting:  
<http://wiki.ros.org/indigo/Installation/Ubuntu>

In short, it boils down to executing the following in the terminal:

- Setup rosdep:

```
sudo rosdep init && rosdep update
```



## Task 3: Setup ROS

Consult the official installation instructions for troubleshooting:  
<http://wiki.ros.org/indigo/Installation/Ubuntu>

In short, it boils down to executing the following in the terminal:

- Setup rosdep:

```
sudo rosdep init && rosdep update
```

- Initialize the ROS environment for this particular terminal:

```
source /opt/ros/indigo/setup.bash
```

## Task 3: Setup ROS

Consult the official installation instructions for troubleshooting:  
<http://wiki.ros.org/indigo/Installation/Ubuntu>

In short, it boils down to executing the following in the terminal:

- Setup rosdep:

```
sudo rosdep init && rosdep update
```

- Initialize the ROS environment for this particular terminal:

```
source /opt/ros/indigo/setup.bash
```

- Create a directory where the code you'll write will be stored (the name `ros_ws` and the location `~` can be changed):

```
mkdir -p ~/ros_ws/src
```

## Task 3: Setup ROS

Consult the official installation instructions for troubleshooting:  
<http://wiki.ros.org/indigo/Installation/Ubuntu>

In short, it boils down to executing the following in the terminal:

- Setup rosdep:

```
sudo rosdep init && rosdep update
```

- Initialize the ROS environment for this particular terminal:

```
source /opt/ros/indigo/setup.bash
```

- Create a directory where the code you'll write will be stored (the name `ros_ws` and the location `~` can be changed):

```
mkdir -p ~/ros_ws/src
```

- Initialize the workspace:

```
cd ~/ros_ws && catkin_make
```

## Task 3: Setup ROS

Consult the official installation instructions for troubleshooting:  
<http://wiki.ros.org/indigo/Installation/Ubuntu>

In short, it boils down to executing the following in the terminal:

- Setup rosdep:

```
sudo rosdep init && rosdep update
```

- Initialize the ROS environment for this particular terminal:

```
source /opt/ros/indigo/setup.bash
```

- Create a directory where the code you'll write will be stored (the name `ros_ws` and the location `~` can be changed):

```
mkdir -p ~/ros_ws/src
```

- Initialize the workspace:

```
cd ~/ros_ws && catkin_make
```

- Update your bash startup script and make sure it worked:

```
echo -e "\n# ROS\nsource $HOME/ros_ws/devel/setup.bash\n" >> ~/.bashrc && tail ~/.bashrc && source ~/.bashrc
```

## Task 4a: Git and GitHub

- Create an account on GitHub if you don't have one yet and request a private repository student discount for it (use an Uni Bremen email):

<https://github.com/join>

[https://education.github.com/discount\\_requests/new](https://education.github.com/discount_requests/new)

## Task 4a: Git and GitHub

- Create an account on GitHub if you don't have one yet and request a private repository student discount for it (use an Uni Bremen email):  
`https://github.com/join      https://education.github.com/discount\_requests/new`
- Create a new empty repository called `lisp_course_material` in your GitHub account and make it private once possible.

## Task 4a: Git and GitHub

- Create an account on GitHub if you don't have one yet and request a private repository student discount for it (use an Uni Bremen email):  
<https://github.com/join>      [https://education.github.com/discount\\_requests/new](https://education.github.com/discount_requests/new)
- Create a new empty repository called `lisp_course_material` in your GitHub account and make it private once possible.
- Add gaya- (mind the dash!) as a collaborator to that repo.

## Task 4a: Git and GitHub

- Create an account on GitHub if you don't have one yet and request a private repository student discount for it (use an Uni Bremen email):
- Create a new empty repository called `lisp_course_material` in your GitHub account and make it private once possible.
- Add gaya- (mind the dash!) as a collaborator to that repo.
- Install Git:

<https://github.com/join>      [https://education.github.com/discount\\_requests/new](https://education.github.com/discount_requests/new)

```
sudo apt-get install git
```



## Task 4a: Git and GitHub

- Create an account on GitHub if you don't have one yet and request a private repository student discount for it (use an Uni Bremen email):  
`https://github.com/join`     `https://education.github.com/discount_requests/new`
- Create a new empty repository called `lisp_course_material` in your GitHub account and make it private once possible.
- Add gaya- (mind the dash!) as a collaborator to that repo.

- Install Git:

```
sudo apt-get install git
```

- Download the course material into your ROS workspace:

```
roscd && cd ../src && git clone https://github.com/code-iai/lisp_course_material.git && ll
```

## Task 4a: Git and GitHub

- Create an account on GitHub if you don't have one yet and request a private repository student discount for it (use an Uni Bremen email):  
`https://github.com/join`     `https://education.github.com/discount_requests/new`
- Create a new empty repository called `lisp_course_material` in your GitHub account and make it private once possible.
- Add gaya- (mind the dash!) as a collaborator to that repo.

- Install Git:

```
sudo apt-get install git
```

- Download the course material into your ROS workspace:

```
roscd && cd ../src && git clone https://github.com/code-iai/lisp_course_material.git && ll
```

- Define a remote target with the address of your new GitHub repo:

```
cd lisp_course_material && git remote add my-repo https://github.com/YOUR_GITHUB_USERNAME/lisp_course_material.git
```

## Task 4a: Git and GitHub

- Create an account on GitHub if you don't have one yet and request a private repository student discount for it (use an Uni Bremen email):  
`https://github.com/join`    `https://education.github.com/discount_requests/new`
- Create a new empty repository called `lisp_course_material` in your GitHub account and make it private once possible.
- Add gaya- (mind the dash!) as a collaborator to that repo.

- Install Git:

```
sudo apt-get install git
```

- Download the course material into your ROS workspace:

```
roscd && cd ../src && git clone https://github.com/code-iai/lisp_course_material.git && ll
```

- Define a remote target with the address of your new GitHub repo:

```
cd lisp_course_material && git remote add my-repo https://github.com/YOUR_GITHUB_USERNAME/lisp_course_material.git
```

- Upload the files to your new GitHub repo:

```
git push -u my-repo master
```

## Task 4b: Git and GitLab

- Log into university GitLab with your LDAP / TZI account:

<https://gitlab.informatik.uni-bremen.de/>

## Task 4b: Git and GitLab

- Log into university GitLab with your LDAP / TZI account:  
`https://gitlab.informatik.uni-bremen.de/`
- Click on “+ New Project”, call the project `lisp_course_material` and make sure it is private.

## Task 4b: Git and GitLab

- Log into university GitLab with your LDAP / TZI account:  
`https://gitlab.informatik.uni-bremen.de/`
- Click on “+ New Project”, call the project `lisp_course_material` and make sure it is private.
- Once created, in “Members” tab add “Gayane Kazhoyan” as a collaborator to the project. “Project Access” should be master.

## Task 4b: Git and GitLab

- Log into university GitLab with your LDAP / TZI account:

```
https://gitlab.informatik.uni-bremen.de/
```

- Click on “+ New Project”, call the project `lisp_course_material` and make sure it is private.
- Once created, in “Members” tab add “Gayane Kazhoyan” as a collaborator to the project. “Project Access” should be master.
- Install Git:

```
sudo apt-get install git
```

## Task 4b: Git and GitLab

- Log into university GitLab with your LDAP / TZI account:

```
https://gitlab.informatik.uni-bremen.de/
```

- Click on “+ New Project”, call the project `lisp_course_material` and make sure it is private.
- Once created, in “Members” tab add “Gayane Kazhoyan” as a collaborator to the project. “Project Access” should be master.
- Install Git:

```
sudo apt-get install git
```

- Download the course material into your ROS workspace:

```
roscd && cd ../src && git clone https://github.com/code-iai/lisp_course_material.git && ll
```



## Task 4b: Git and GitLab

- Log into university GitLab with your LDAP / TZI account:

```
https://gitlab.informatik.uni-bremen.de/
```

- Click on “+ New Project”, call the project `lisp_course_material` and make sure it is private.
- Once created, in “Members” tab add “Gayane Kazhoyan” as a collaborator to the project. “Project Access” should be master.
- Install Git:

```
sudo apt-get install git
```

- Download the course material into your ROS workspace:

```
roscd && cd ../src && git clone https://github.com/code-iai/lisp_course_material.git && ll
```

- Define a remote target with the address of your new GitHub repo:

```
cd lisp_course_material
```

```
git remote add my-repo https://gitlab.informatik.uni-bremen.de/YOUR_GITHUB_USERNAME/lisp_course_material.git
```

## Task 4b: Git and GitLab

- Log into university GitLab with your LDAP / TZI account:

```
https://gitlab.informatik.uni-bremen.de/
```

- Click on “+ New Project”, call the project `lisp_course_material` and make sure it is private.
- Once created, in “Members” tab add “Gayane Kazhoyan” as a collaborator to the project. “Project Access” should be master.
- Install Git:

```
sudo apt-get install git
```

- Download the course material into your ROS workspace:

```
roscd && cd ../src && git clone https://github.com/code-iai/lisp_course_material.git && ll
```

- Define a remote target with the address of your new GitHub repo:

```
cd lisp_course_material
```

```
git remote add my-repo https://gitlab.informatik.uni-bremen.de/YOUR_GITHUB_USERNAME/lisp_course_material.git
```

- Upload the files to your new GitHub repo:

```
git push -u my-repo master
```

## Task 5: Install the IDE

- Install the editor itself (Emacs), the Common Lisp compiler (SBCL), the linker (ASDF) and the Emacs Common Lisp plugin (Slime):

```
sudo apt-get install ros-indigo-roslisp-repl
```

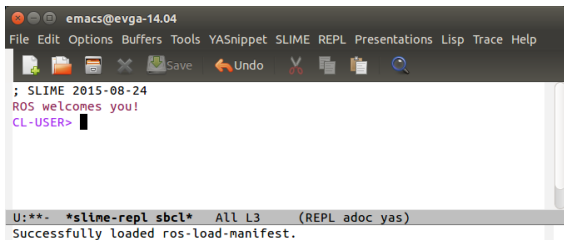
## Task 5: Install the IDE

- Install the editor itself (Emacs), the Common Lisp compiler (SBCL), the linker (ASDF) and the Emacs Common Lisp plugin (Slime):

```
sudo apt-get install ros-indigo-ros-lisp-repl
```

- Start the editor (after compilation is finished you'll see the Lisp shell):

```
roslisp_repl &
```



```
emacs@evga-14.04
File Edit Options Buffers Tools YASnippet SLIME REPL Presentations Lisp Trace Help
; SLIME 2015-08-24
ROS welcomes you!
CL-USER>
U:**- *slime-repl sbcl* ALL L3 (REPL adoc yas)
Successfully loaded ros-load-manifest.
```

## Task 6: Get familiar with Emacs

The following notation is used in Emacs for keyboard shortcuts:

- C for <Ctrl>
- M for <Alt>
- - for when two keys are pressed together (e.g. C-x for <Ctrl>+x)
- SPC for <Space>
- RET for <Enter>

The basic shortcuts you will need are listed below:

- C-x C-f opens a file
- C-x 3 or C-x 2 opens a new tab, C-x 0 closes it, C-x 1 maximizes
- C-x o switches between tabs
- C-x b switches buffers, C-x C-b lists all open buffers, C-x k kills
- C-g cancels a command half-way, C-x C-c yes exits Emacs

## Task 6: Get familiar with Emacs

The following notation is used in Emacs for keyboard shortcuts:

- C for <Ctrl>
- M for <Alt>
- - for when two keys are pressed together (e.g. C-x for <Ctrl>+x)
- SPC for <Space>
- RET for <Enter>

The basic shortcuts you will need are listed below:

- C-x C-f opens a file
- C-x 3 or C-x 2 opens a new tab, C-x 0 closes it, C-x 1 maximizes
- C-x o switches between tabs
- C-x b switches buffers, C-x C-b lists all open buffers, C-x k kills
- C-g cancels a command half-way, C-x C-c yes exits Emacs

Open the file with your first assignment and follow the instructions:

ROS\_WORKSPACE/src/lisp\_course\_material/assignment\_1/src/orc-battle.lisp  
Introduction

Assignment

## Task 7: Get familiar with Git

- Once done editing `orc-battle.lisp`, setup colorful output for Git and check what's new in your local repo (the one on your hard drive):

```
git config --global color.ui true && cd ROS_WORKSPACE/src/lisp_course_material && git status
```

## Task 7: Get familiar with Git

- Once done editing `orc-battle.lisp`, setup colorful output for Git and check what's new in your local repo (the one on your hard drive):

```
git config --global color.ui true && cd ROS_WORKSPACE/src/lisp_course_material && git status
```

- To see which exactly lines changed ask for the diff (`q` to exit):

```
git diff
```



## Task 7: Get familiar with Git

- Once done editing `orc-battle.lisp`, setup colorful output for Git and check what's new in your local repo (the one on your hard drive):

```
git config --global color.ui true && cd ROS_WORKSPACE/src/lisp_course_material && git status
```

- To see which exactly lines changed ask for the `diff` (`q` to exit):

```
git diff
```

- The red files are the new untracked ones, the green ones are already in the Git index. To add new files to the index use

```
git add .
```

## Task 7: Get familiar with Git

- Once done editing `orc-battle.lisp`, setup colorful output for Git and check what's new in your local repo (the one on your hard drive):

```
git config --global color.ui true && cd ROS_WORKSPACE/src/lisp_course_material && git status
```

- To see which exactly lines changed ask for the diff (*q* to exit):

```
git diff
```

- The red files are the new untracked ones, the green ones are already in the Git index. To add new files to the index use

```
git add .
```

- If you deleted some files, to remove them from the index use

```
git add -u
```

## Task 7: Get familiar with Git

- Once done editing `orc-battle.lisp`, setup colorful output for Git and check what's new in your local repo (the one on your hard drive):

```
git config --global color.ui true && cd ROS_WORKSPACE/src/lisp_course_material && git status
```

- To see which exactly lines changed ask for the diff (*q* to exit):

```
git diff
```

- The red files are the new untracked ones, the green ones are already in the Git index. To add new files to the index use

```
git add .
```

- If you deleted some files, to remove them from the index use

```
git add -u
```

- Once you're sure the changes are final, commit locally:

```
git commit -vm "A meaningful commit message."
```

## Task 7: Get familiar with Git

- Once done editing `orc-battle.lisp`, setup colorful output for Git and check what's new in your local repo (the one on your hard drive):  

```
git config --global color.ui true && cd ROS_WORKSPACE/src/lisp_course_material && git status
```
- To see which exactly lines changed ask for the diff (*q* to exit):  

```
git diff
```
- The red files are the new untracked ones, the green ones are already in the Git index. To add new files to the index use  

```
git add .
```
- If you deleted some files, to remove them from the index use  

```
git add -u
```
- Once you're sure the changes are final, commit locally:  

```
git commit -vm "A meaningful commit message."
```
- Finally, to upload your local commits to the GitHub server, push the changes upstream:  

```
git push # or git push my-repo master
```

# Links

- Emacs cheat sheet:

[http://www.ic.unicamp.br/~helio/disciplinas/MC102/Emacs\\_Reference\\_Card.pdf](http://www.ic.unicamp.br/~helio/disciplinas/MC102/Emacs_Reference_Card.pdf)

- Git reference book:

<http://git-scm.com/book/de>

# Info summary

## Assignment:

- Due: 12.10, Tuesday, 08:00
- Points: 3 of 50
- For questions: write an email to `gaya@cs.uni-bremen.de`, use the StudIP forum or come by at TAB 1.57

## Next class:

- Date: 12.10
- Time: 16:15 (16:00 - 16:15 for questions)
- Place: same room (TAB 0.31)

# Q & A

Thanks for your attention!