

# Course Format and Tools

---



# The Course

**This is the second module of the Lab of AI application course**

...Where we are mostly going to talk about **Machine Learning**

- Before we can start in earnest, however
- ...We need to familiarize with a few tools, because...

**The module will be delivered as a series of tutorials**

- We will discuss theoretical content when needed
- ...But mostly we will learn about ML by solving simple use cases

**Exercises will be mostly about playing around with the tutorial code**

- Nobody expects you to become an expert
- ...But the ability to write simple code will be required for the exam



# Jupyter

**The first tools we'll use throughout the module is called Jupyter**

...Which is an open-source project about **interactive computation environments**

- The project offers a few tools to write code
- ...Inspect the results
- ...And the modify the code until you are satisfied with the outcome

**The project provides three main tools**

- Jupyter **console**, i.e. an interactive terminal
- Jupyter **notebook**, i.e. an interactive web tool combining text and code
- Jupyter **lab**, which builds over the notebook pushing it close to an IDE

Multiple programming languages are supported



# Notebook di Jupyter

## In particular, we will use Jupyter notebooks

...Which work by relying on three main processes:

- A process (**server**) allows access the notebooks as web pages
- Your **browser** displays the pages and an editing UI
- An interpreter (**kernel**) handles code execution

## A notebook is split into **cells** (displayed as boxes)

There two types of cells, and both can be **run**

### ■ **Text** cells

- When run they are rendered as rich text (fonts, pictures, etc.)

### ■ **Code** cells

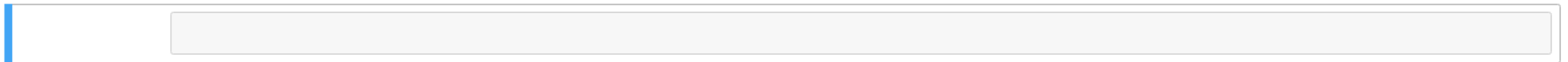
- When they are run, the code they contain is sent to the kernel
- ...And the results are displayed immediately underneath



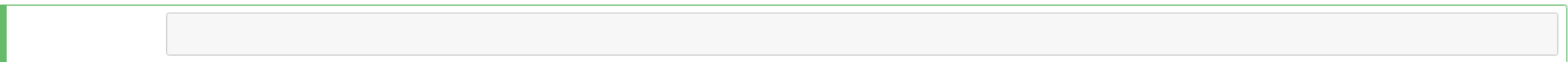
# Selecting Cells

**You can select a cell by simply clicking over the box**

- With a single click, you'll open the cell in **normal mode**
- ...Which can be recognized from the **blue** left-border









- With a double-click, you'll open the cell in **edit mode**
- ...Which can be recognized from the **green** left-border
- You can also press `enter` with the cell selected in normal mode



# Editing Cells

## You can modify cells in more than one way

- First, in edit mode you can **change their content** by just writing
- You can also add new cells with 
- You can cut, copy, and paste cells with   
- You can change the type of a cells with 
- You can save the notebook with 



# Running Cells

**You can control cell execution with these buttons:**



- The first button runs a cell
- The second interrupts execution
  - ...Which is useful when you accidentally start a long operation
- The third restarts the kernel
  - ...Which will kill all variables, functions, and modules
- The last one will restarts the kernel and runs the whole notebook

A list of more advanced commands can be accessed by clicking 



# Text Cells

Text cells are written in markdown format

...Which is plain text with some simple conventions

**# A single "hash" means a title**

**## Two hashes denote a subtitle, and so on**

\* You can use stars for bullet lists

- Dashes are fine, too

**\*\*A double star (or underscore) is used for bold text\*\***

*\_A single underscore (or star) denotes emphasized text\_*

- You can open any text cell in this notebook in edit mode and play around





# Text Cells

## When you run a text cell

- The markdown source is translated to HTML
- ...Which is the language used for web pages
- Your browser can then render the result in an esthetically pleasing format

## Try to make some changes to the following cell:

- Change this text as you wish!
- If you then get nostalgic about its previous look
- ...You can still press ctrl+z to undo the last action
- ...If that fails, you can still re-download the lecture ;-)



# Code Cells

## Our code cells will be written in Python

The tool can be configured to support Julia an R as well

- Code cells and sent to the kernel for execution
- ...And the results are then display (in HTML format, in some cases)

## Here's a simple code cell:

```
In [1]: print('Hello, world!')
```

```
Hello, world!
```

## ...And, if you are curious:

- I am using this plugin to show the notebook in presentation mode
- We won't cover that in the course, but it's not super complicated to use



# Other Requirements

**We will also use a number of Python modules**

Most of these will be well-established modules for Data Science

- `numpy` for vector computation
- `scipy` for numerical algorithms
- `pandas` for dataset management and inspection
- ...

**Each lecture contains a `requirements.txt` file**

- Inside the file you'll find which modules are needed for the lecture
- ...And you'll be able to install them using:
  - `conda` if you are using Anaconda Python
  - ...Or `pip` for a regular Python distribution



## Other Requirements

We will also use a number of **Python modules**

We will also use **custom modules**

- In particular in many lectures you'll find a folder called `util`
- ...Which will contain a file called `util.py`

**This is a module built ad-hoc for the given lecture**

- Longer code section will not be included in the notebooks directly
- ...But they will instead be provided in the module
- If you are curious about their inner working
- ...You can check the code directly (it's a standard Python source file)



# Other Requirements

## Due to the use of custom modules

...We will often start our notebooks with these Jupyter directives:

```
In [1]: %load_ext autoreload
        %autoreload 2
```

- These are not Python instructions, but directives for Jupyter
- The first one loads an extension (called `autoreload`)
- The second one tells the extension to **reload all modules** when a cell is run
- With this trick, if we make a change to a module...
- ...The change will immediately affect the notebook

Normally, modules are reloaded **only if we restart the notebook**

