System Setup

We will dive right into coding in the very first lecture. You have different option on how to prepare your development environment for the class.

A: Install Git and Close (or Fork) the Class Repository

a. Install a Git Client

Git clients are available for all relevant OS. You can use the official installers (https://git-scm.com/). If you are running Windows, TortoiseGit is also a good option (https://tortoisegit.org/).

b. Decide on the <class directory>

Decide in which directory (we will be calling this the <class directory>) to keep the files ("notebooks") you are working on.

c. Close the class repository

Open a terminal and change into the <class directory>. Clone the class repository with

```
$ git clone https://inf-git.fh-rosenheim.de/brm/data_aai_sose23.git
```

If you want to use Git as your source code version control system, you may want to fork the class repository instead.

B: Setup Jupyter Environment

We will be working extensively with Jupyter Notebooks, so you need an environment to execute them in. All notebooks have been tested with defined versions of Python and the relevant libraries, so you should make sure to use a system with these exact same versions. The recommendation for this class is to use a docker-based setup, but if you want to, you can use local install (preferably based on Anaconda for version control) or a cloud-based setup (version control depends on the cloud provider) as well.

Here are the steps for the docker-based Environment.

a. Install Docker (if not already installed)

See https://docs.docker.com/install/

b. Test you docker installation

Test your docker installation by running the Docker Hello World container with `docker run hello-world`:

```
$ docker run hello-world
...
Hello from Docker!
This message shows that your installation appears to be working correctly.
...
```

Note: You may have to start the demon "Docker Desktop" or "Docker Demon" first.

c. Build the docker image

For this class, we will be using a docker container based on the official `jupyter/datascience-notebook` hosted on docker hub. Unfortunately, it does not include all libraries we will be using (e.g., XGBoost), thus you will find a `Dockerfile` in the git-repo, which uses it as a base and installs the missing libraries.

Run

```
docker build -t datascience .
```

from the root of the git repository to build the `datascience` image (do not forget the trailing . in the command!)

d. Run the docker container

- As you cannot store documents inside a docker container (containers/images get destroyed all the time), you will have to "map" a persistent directory to store your files in into the container. We will be using the <class directory> as persistent directory and map it into the container as "/home/jovyan/work" (this path is predefined by the container) with the `-v` option.
- We also need to specify a port for the web-server to listen to, we will be using 8888 (`-p`),
- it makes sense to clean up the filesystem after running the container (`--rm`),
- and to give it a clear name (`--name`).

Examples:

Open a terminal, change into your <class directory> and run the container using

- On Linux or MacOS, execute

```
$ docker run -p 8888:8888 -v "$PWD":/home/jovyan/work --rm --name datascience datascience
```

- On Windows running storing your files in `C:\temp\`, execute

```
$ docker run -p 8888:8888 -v C:\temp\:/home/jovyan/work --rm --name datascience datascience
```

If this works properly, it will show you how to access the Notebook-Server through a web browser, this should look like this (your token will be different every time you start the Notebook-Server):

```
  To access the notebook, open this file in a browser:
        file:///home/jovyan/.local/share/jupyter/runtime/jpserver-7-open.html
    Or copy and paste one of these URLs:
        http://ee4d6332eca1:8888/?token=1ae2a6f5cae0fbad59dd8eda6c1c8346bdcd4ddbd5055c89
     or http://127.0.0.1:8888/?token=1ae2a6f5cae0fbad59dd8eda6c1c8346bdcd4ddbd5055c89
```

Copy-and-Paste the last URL (the one starting with `http://127.0.0.1:8888/...`) into a web browser (Chrome is recommended) on your computer. You should now see the JupyterLab UI.

WINDOWS USERS

In the Windows-Docker implementation, you may get an error when you try to map non-local drives into a container. E.g., the example above using `C:\temp` as <class directory> will work but using `Z:\data\markus` if `Z:` is a network volume will not work. This may be fixed by the time you read this – give it a try. If you get an error, try copying your <class directory> to a local directory, and copy it back at the end of the lecture (yes, this is annoying).