This document should help to prepare your computer and yourself for the programming exercises of "Data Science 1". We use `Python` as programming language, as it is the most popular programming language for data science. In particular, we use the packages `pandas` for data handling and `scikit-learn` for ML algorithms. In theory, you can also solve the tasks in another programming language (e.g., R, which we used for the exercises in previous years). However, the published solutions are in Python and hints in the tasks (on packages/functions) target at that language as well.

1 Installing Python

You can obtain Python from its website. The solutions to the exercises use Python 3.8. However, as we stick to Python's basics, you should be fine with slightly older and newer versions as well. (But please don't use versions of Python 2.)

Instead of obtaining a bare Python installation, you can also install one of the Python distributions Anaconda or Miniconda. Besides providing Python, both allow to easily setup environments with specific Python versions, as we discuss in Section 4. While `Miniconda` uses less disk space and already is sufficient for the exercises, Anaconda comes with lots of pre-installed Python packages, a pre-installed IDE (`Spyder`) and a (IMHO not really user-friendly) GUI for package management.

2 Installing an IDE

The published solutions will all use `Jupyter Notebooks`, which allow to mix code snippets with explanatory text in Markdown format. Such notebooks are very popular in data science, as they allow explaining your analyses and communicating results easily. Using `Jupyter Notebooks`, you don't need another IDE for the exercises. You can view, edit, and run notebooks in a web browser. See Section 6 for installation instructions.

If you still want to write classical source-code files and benefit from features like automatic style and error checking etc., you should obtain a proper IDE. One simple option is `Spyder`, which comes with `Anaconda`, but you can also install it as a standalone program or as a Python package as well.

3 Learning Python

In the exercises, we use Python to solve concrete data-science tasks. The goal is not to systematically learn Python as a language. Most of the code focuses on using particular data-science libraries anyway instead of writing large programs from scratch. For example, though we'll implicitly use object orientation, you don't need to write classes, overwrite methods etc. Some of the relevant syntactical elements are:

- `if` statements
- `for` loops and list comprehensions
- elementary data types as well as collections (dictionaries, lists, tuples, and sets)
- functions
- comments

You can find a short intro into various language elements of Python in the W3Schools tutorial. Also, Chapters 1-5 of the official tutorial might help you to get familiar with the basics. The official documentation might be helpful if you need a deep dive into certain aspects. Generally, we advise to not spend too much time grasping details of Python beforehand, but rather to learn things on the fly.

4 Preparing a Virtual Environment

We'll heavily use third-party libraries (= packages), which need to be installed first. A virtual environment allows storing all your Python dependencies for these exercises at a dedicated location, independent from the packages you want to use with your main Python installation. If you don't need to store other versions of the exercise's packages in parallel (e.g., if you are only using Python for these exercises anyway), you don't need to set up a virtual environment.

4.1 Alternative 1: `virtualenv` Environment

If you only have a bare Python without `conda`, we recommend using `virtualenv` to set up a dedicated environment. First, install `virtualenv` with

```
python -m pip install virtualenv
```

To set up an environment at a certain destination on your hard drive, run

```
python -m virtualenv -p <path/to/python/executable> <path/to/env/dest>
```

Activate the environment in Linux with

```
source <path/to/env/destination>/bin/activate
```

Activate the environment in Windows (note the back-slashes) with

```
<path\to\env\destination>\Scripts\activate
```

After activation, you can simply install packages from the Python Package Index into the environment like

```
python -m pip install pandas
```

To leave the environment, run

```
deactivate
```

4.2 Alternative 2: `conda` Environment

If you have `Anaconda` or `Miniconda`, use `conda` to set up an environment:

```
conda create --name ds-2021 python=3.8
conda activate ds-2021
```

You might replace `ds-2021` with another name. Now you can install packages like this:

```
conda install pandas
```

If packages are not listed in `Anaconda`'s package repo, you can also install them with `pip` as described above. To leave the environment, run

```
conda deactivate
```

5 Installing a List of Packages

We provide a file `requirements.txt` on ILIAS, listing all required packages (and their versions) for the exercises. To install all these dependencies at once, simply activate your environment and run

```
python -m pip install -r requirements.txt
```

If you make changes to the environment (e.g., add packages, update packages) and you want to persist these changes to the requirements file, run

```
python -m pip freeze > requirements.txt
```

We also created the original requirements file this way.

6 Using Notebooks

To run or create `Jupyter Notebooks` from the environment, you need to install the package `notebook` into the environment (after activating it!), e.g., with

```
python -m pip install notebook
```

If you have `Anaconda`, you can also directly use the version of `notebook` shipped with it. Next, you need to install a kernel for the environment:

```
ipython kernel install --user --name=ds-2021-kernel
```

The kernel allows using your environment in notebooks. Thus, you should see (and be able to select) the kernel in the menu after opening a notebook. Start `Jupyter Notebook` with

```
jupyter notebook
```

Note that you only get access to files in the directory from which you started `Jupyter Notebooks` or in sub-directories.
For an introduction into the functionality of `Jupyter Notebooks`, you can have a look at the official documentation or this tutorial.