To set up an environment to work with Python, Tensorflow, and Keras on Jupyter Notebook, you have two options:
1) Use Google Colab if your computer is rather slow. You will need a Gmail account.
2) Follow the steps below in order to work directly on your computer.

Step A: Setting up Python, Tensorflow 2.12, Keras, and other dependencies

A1. Download and install Anaconda or Miniconda.

A2. Open a terminal application and use the default bash shell. Go to the directory "`AI4REG_workshop`" in which you have downloaded, via the terminal.

A3. Create a Python environment with conda by typing this on the terminal:

```
conda create -n ai4reg python=3.10
```

Here, we create the conda environment called "ai4reg" using Python 3.10. Using other versions of Python is not recommended as it requires different versions of dependencies.

A4. Activate the created environment by typing

```
conda activate ai4reg
```

A5. Install `pip` which is the standard package manager for Python, to install and manage libraries and dependencies

```
conda install pip
```

A6. Install other dependencies including Tensorflow, Keras, and Scikit-learn by running

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

A7. To check if all the dependencies are installed correctly, type the following commands after typing "python" in your terminal (check if the right version of Python is shown). All should run successfully, although a warning message may pop up (see **B6** on page 2).

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.dates as mdates
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import mean_squared_error, r2_score
from keras.models import Sequential
from keras.layers import LSTM
from keras.layers import Dense
from keras.layers import Dropout
from keras import optimizers
import tensorflow as tf
from datetime import datetime
```

Note that if you want to run Tensorflow on GPU (if you have it on your computer), see the detailed instruction here: https://docs.anaconda.com/working-with-conda/applications/tensorflow/.

Step B: Setting up Jupyter Notebook and using the virtual environment on Notebook

B1. With the "ai4reg" environment activated, install `ipykernel`, which is a tool that allows you to execute Python code within Jupyter Notebooks, by typing

```
conda install ipykernel
```

B2. Add "ai4reg" as a kernel for Jupyter Notebooks

```
ipython kernel install --user --name=ai4reg
```

B3a. If you have Jupyter Notebook installed already, you can run it directly via the system installation or via terminal by typing

```
jupyter notebook
```

B3b. If you don't have it yet, you can install it in a new terminal. Alternatively, in the same terminal, deactivate the "ai4reg" environment first via

```
conda deactivate
```

Then install Jupyter Notebook via conda:

```
conda install jupyter
```

B4. You can now run Jupyter Notebook (see **B3a**). Open the file "Test-dependencies-installation.ipynb" in the directory "`AI4REG_workshop`".

B5. Navigate to 'Kernel' on the toolbar of Jupyter Notebook, change the kernel to "ai4reg".

B6. Execute the first cell. This should run without any problem. You may receive the following message, which is totally normal for those who have only CPUs on their computer.

```
2024-11-10 13:40:04.931001: I
tensorflow/core/platform/cpu_feature_guard.cc:182] This
TensorFlow binary is optimized to use available CPU
instructions in performance-critical operations.
To enable the following instructions: AVX2 AVX512F AVX512_VNNI
FMA, in other operations, rebuild TensorFlow with the
appropriate compiler flags.
```

B7. We also have the "TensorBoard" extension on Jupyter Notebook that will allow us to track the progress of the neural network training processes. Execute the second cell and see if you can also open the TensorBoard via the web browser at http://localhost:6006/#.