# Installation instructions for Software Environment of Machine Perception & Intelligent Vehicles practica

You can find the installation instructions for Windows, Linux, MacOS and Docker (on Linux) in the following sections.

## Windows

If Anaconda3 is already installed (like on the TU Delft lab computers), you can proceed to step 3 directly.

1. Download the Anaconda3 Python installer from https://www.anaconda.com/distribution/
2. Install Anaconda3 Python. Choose the option to install for the current user only, in order to avoid needing admin permissions.
3. Download the practicum archive file from brightspace. Save the file in, for example `H:\Desktop\mpiv`. Make sure that your directory does NOT contain whitespace characters. (H indicates the drive, if you are working on your own laptop instead of the practicum hall systems, this is most likely your `C` drive so `C:\Desktop\mpiv`)
4. Windows allows extracting tar.gz files using the command prompt, but it does not always support it from the right-click menu. Therefore, open the command prompt, in `Start menu -> Command Prompt` (Dutch: opdrachtprompt).
5. Navigate to the practicum directory in the command prompt, by first changing to the right drive. In case of `H:\Desktop\mpiv`: type `H:` and press ENTER. Then navigate to the right directory with `cd Desktop\mpiv`.
6. Extract the tar file by executing `tar -xvzf practicum1-<hash>.tar.gz`.
7. Download `data.tar.gz` from brightspace to your practicum directory (`H:\Desktop\mpiv\practicum1-<hash>\data-<hash>.tar.gz`) and extract the file using `tar -xvzf data.tar.gz`.
8. Double check to make sure that the `data` directory is next to the `release` directory. These two directories should be at the same level.
9. Open **Anaconda Prompt (Anaconda3)** from the Start menu (do NOT use Anaconda Powershell Prompt).
10. In the terminal use `cd` and navigate to the practica materials directory.
11. Run `conda env create -f environment.yml`. This creates the conda environment `mp-iv` with the packages defined in `environment.yml` and can take a while. There is a known issue with terminal output not logging to the screen. If the output seems to hang, check a workaround here.
12. Run `01-activate.bat`. This will select the conda environment `mp-iv` and set the needed environment variables (`PYTHONPATH`)
13. Run `jupyter notebook`
14. Now open the notebook (`ipynb`) file of the practicum by selecting the file

in the browser.
15. Happy coding!

Every time that we want to start to work again we have to: 1. Open **Anaconda Prompt (Anaconda3)** in the Start menu 2. Go to the practica materials directory (for example `cd Desktop/mpiv`) 3. Type `01-activate.bat` (this will also activate the environment mp-iv, that you see in parenthesis before the directory and set needed environment variables (`PYTHONPATH`) 4. Type `jupyter notebook` (this will open the notebook server from where you can open the practica and assignment notebook files)

## Linux

The instructions were tested on Ubuntu 18.04.

Be aware that if conda is already installed on your system, the following steps may result in conflicts. Proceed to step 4 if conda is already installed.

1. Download **anaconda** 64-Bit (x86) Installer for Linux from https://www.anaconda.com/products/individual. A file named `Anaconda3-2022.05-Linux-x86_64.sh` (or similar) should be downloaded.

2. Move the downloaded file to your folder and execute the file.

```
mv ../Downloads/Anaconda3-2022.05-Linux-x86_64.sh ./
bash Anaconda3-2022.05-Linux-x86_64.sh
```

Accept the license terms by typing yes as in the following:

```
Do you accept the license terms? [yes|no]
[no] >>>  yes
```

Anaconda3 will now be installed into this location:

```
/home/$user$/anaconda3

Press ENTER to confirm the location
Press CTRL-C to abort the installation
Or specify a different location below

[/home/$user$/anaconda3] >>>

Press Enter and wait until conda is successfully installed.

Now initialize conda by typing yes as in the following:

Executing transaction: done
installation finished.
Do you wish the installer to initialize Anaconda3
```

```
by running conda init? [yes|no]
[no] >>> yes

==> For changes to take effect, close and re-open your current shell. <==
```

If you'd prefer that conda's base environment not be activated on startup, set the auto_activate_base parameter to false:

```
conda config --set auto_activate_base false
```

```
Thank you for installing Anaconda3!
```

Type the following and press Enter, depending on your shell:

```
source ~/.bashrc
```

or

```
source ~/.zshrc
```

Now run the following command to update the conda packaging tool to the latest version:

```
conda update -n base -c defaults conda
```

The following packages will be UPDATED:

```
conda             4.8.5-py37_0 --> 4.10.3-py37_0
```

```
Proceed ([y]/n)?
```

Press **Enter** and wait for the update process to finish.

Try the following commands to see if the installation worked for you.

```
conda --version
```

You should be able to see something like this: "conda 4.10.3"

3. Create the directory `~/mpiv` with `mkdir ~/mp-iv`.
4. Download the practicum archive and save it in `~/mpiv`. Navigate to the directory with:

```
cd `~/mp-iv`
```

5. Extract the archive with `tar -xvzf practicum1-<hash>.tar.gz`.
6. Download `data.tar.gz` from brightspace to `~/mpiv` and extract the archive with `tar -xvzf data.tar.gz`.
7. Create the mp-iv Environment

```
conda env create -f environment.yml
```

Next, activate the new environment:

```
source 01-activate.sh
```

Also try the following:

```
which python
```

You should be able to see an output similar to the following:

```
/home/$user$/anaconda3/envs/mp-iv/bin/python
```

8. Write the following command to open a jupyter notebook in your browser:

```
bash 02-start-notebook.sh
```

You now should be able to run the first practicum

Please make sure, that you will have to activate the environment (mp-iv) you just created every time you restart your terminal window (or computer) by executing `source 01-activate.sh`. So do not forget to do this before you start working on your practica with `02-start-notebook.sh`.

## MacOS

We do not have a MacOS system ourselves, so we cannot easily verify if installation on MacOS succeeds. Installation on systems with ARM-based chips (like the popular M1 chip) will *not* succeed. However, last years, it turns out that installation on MacOS with x86-based chip architectures usually works. Download the MacOS installer from https://www.anaconda.com/products/individual and follow the remaining steps (starting at step 3) from the Linux installation instructions.

## Docker (on Windows)

1. Install Docker with: https://docs.docker.com/desktop/windows/install/. During the installation it may say that some things can be updated, read that carefully and follow the steps that you get. After the full installation, restart your computer.
2. Make a folder named `mp-iv`.
3. Place the data folder and the contents of the practicum1 file inside the folder `mp-iv`.
4. Open a Windows PowerShell.
5. Navigate to the folder `mp-iv` with the Windows PowerShell, but adapt the commands to your directory:

```
cd C:\Users\Student\Desktop\mp-iv
```

6. Build the image with:

```
docker build -t mp-iv .
```

7. Execute the following commands in the Windows PowerShell, but adapt the commands to your directory:

```
docker run --rm -it -p 8888:8888 -v C:\Users\Student\Desktop\mp-iv:/mp-iv mp-iv bash
source 01-activate.sh
```

You'll see:

```
/opt/conda/condabin/conda
Adding /mp-iv/source, each practicum dir and common dir to PYTHONPATH
```

Execute:

```
jupyter notebook --ip 0.0.0.0 --no-browser --allow-root
```

8. Open browser and search the URL. For me this was (see last line of example output above):

```
http://127.0.0.1:8888/?token=f0903c6e28b2150e60cb57a9161a8d03120b5b62a9aecdf5
```

Note: If some Docker related steps do not work, just restart your computer and try again! If you want to remove Docker you can uninstall it like every other program.

Every time we want to start the notebook again, we have to open a Windows PowerShell. Do the following commands, but adapt the commands to your directory:

```
cd C:\Users\Student\Desktop\mp-iv
docker build -t mp-iv .
docker run --rm -it -p 8888:8888 -v C:\Users\Student\Desktop\mp-iv:/mp-iv mp-iv bash
conda activate mp-iv
source 01-activate.sh
Jupyter notebook --ip 0.0.0.0 --no-browser --allow-root
```

BROWSER: http://127.0.0.1:8888/?token=f0903c6e28b2150e60cb57a9161a8d03120b5b62a9aecdf5 (insert the token you see in the PowerShell)

## Docker (on Linux)

You could also run the notebook in a docker container.

1. Create the directory `~/mpiv` with `mkdir ~/mp-iv`.
2. Download the practicum archive and save it in `~/mpiv`. Navigate to the directory with:

```
cd `~/mp-iv`
```

3. Extract the archive with `tar -xvzf practicum1-<hash>.tar.gz`.
4. Download `data.tar.gz` from brightspace to `~/mpiv` and extract the archive with `tar -xvzf data.tar.gz`.

5. Build the `mp-iv` image with:

```
docker build -t mp-iv .
```

6. Start the container using:

```
docker run --rm -it -p 8888:8888 -v ~/mp-iv:/mp-iv mp-iv bash
```

7. Source the environment variables used in the practicum with:

```
source 01-activate.sh
```

8. Start the jupyter notebook with `--allow-root`, since inside the container you are running it as the root user:

```
jupyter notebook --ip 0.0.0.0 --no-browser --allow-root
```

9. Start a browser on the host (*not* inside the container) and navigate to the URL as shown in the terminal output. It will be something like:

```
http://127.0.0.1:8888/?token=<some_very_long_token>
```

10. You should now be able to open your practicum notebook file.

## Target file structure

After unzipping the files of all practica and assignments, the directory tree should look like this (`data` directory is not shown)

```
.
+-- 01-activate.bat
+-- 01-activate.sh
+-- 02-start-notebook.sh
+-- 82-assemble-data-for-students.filelist
+-- 84-check-data-folder-structure.sh
+-- 90-docker-build-image.bat
+-- 90-docker-build-image.sh
+-- 92-docker-start-notebook.sh
+-- Dockerfile
+-- environment.yml
+-- git-hash
+-- installation_instructions.md
+-- installation_instructions.pdf
+-- pyproject.toml
+-- release
    +-- assignment
    |   +-- fa_00_overview.ipynb
    |   +-- fa_01a_data_visualization.ipynb
    |   +-- fa_02a_3d_pedestrian_detection_single_camera.ipynb
    |   +-- fa_02b_mp_only_3d_pedestrian_detection_multiple_sensors.ipynb
```

```
|   +-- fa_04_iv_only_motion_planning.ipynb
|   +-- git-hash
|   +-- interfaces.py
|   +-- ground_planes.json
|   +-- iv_only_ts_newworld_cam.json
|   +-- linear.png
|   +-- piecewise.png
|   +-- planning_visualization.py
|   +-- solution_helpers.py
|   +-- validation_metrics.py
+-- common
|   +-- git-hash
|   +-- k3d_helpers.py
|   +-- sequence_loader.py
|   +-- visualization.py
+-- practicum1
|   +-- activation_fns.py
|   +-- BoundingBox.py
|   +-- evaluation.py
|   +-- git-hash
|   +-- helpers.py
|   +-- ImagePatchClassifier.py
|   +-- ImagePatch.py
|   +-- image_processing.py
|   +-- load_data.py
|   +-- media
|   |   +-- animation_for_features_and_classifier.mp4
|   |   +-- coord_systems_diagram.svg
|   |   +-- iou_equation.png
|   |   +-- MLP_classifier.drawio.svg
|   |   +-- MobileNetv2_architecture.svg
|   |   +-- modern_view_of_ML.png
|   |   +-- overlapping_bboxes.png
|   |   +-- proposal_box_example.mp4
|   +-- metrics.py
|   +-- NeuralNetClassifier.py
|   +-- pedestrian_classifier
|   |   +-- saved_model.pb
|   |   +-- variables
|   |       +-- variables.data-00000-of-00001
|   |       +-- variables.index
|   +-- practicum1.ipynb
|   +-- preprocessing_fns.py
+-- practicum2
|   +-- check_kf.py
|   +-- compare_measurements.py
```

```
|   +-- copy_sensor.py
|   +-- define_occupancy_map.py
|   +-- evaluate_tracker.py
|   +-- funcs.py
|   +-- git-hash
|   +-- hidden_test_25.pkl
|   +-- images
|   |   +-- assignment_mot.pdf
|   |   +-- assignment_mot.png
|   |   +-- assignment_self-localization.pdf
|   |   +-- assignment_self-localization.png
|   |   +-- vehicle-pf.pdf
|   |   +-- vehicle-pf.png
|   +-- Measure.py
|   +-- Object.py
|   +-- pf_init_around_state.py
|   +-- pf_init_freespace.py
|   +-- practicum2.ipynb
|   +-- run_multi_object_tracker.py
|   +-- run_multiple_kfs.py
|   +-- run_pf.py
|   +-- run_single_kf_gating.py
|   +-- run_single_kf.py
|   +-- selfloc_scenario.py
|   +-- Sensor.py
|   +-- trackeval
|   |   +-- datasets
|   |   |   +-- _base_dataset.py
|   |   |   +-- __init__.py
|   |   |   +-- mot_challenge_2d_box.py
|   |   +-- eval.py
|   |   +-- __init__.py
|   |   +-- metrics
|   |   |   +-- _base_metric.py
|   |   |   +-- count.py
|   |   |   +-- hota.py
|   |   |   +-- __init__.py
|   |   +-- _timing.py
|   |   +-- utils.py
|   +-- visualizations.py
+-- practicum3_iv
|   +-- animate_parking_manoeuvre.py
|   +-- animate_steering_profile_plot.py
|   +-- determine_occupied_cells.py
|   +-- git-hash
|   +-- make_discrete_space.py
```

```
|   +-- motion_planning_types.py
|   +-- parking_scenario.py
|   +-- planning_problems.py
|   +-- plot_functionality.py
|   +-- plot_setup_groundplane_2d.py
|   +-- plot_setup_trajectory_planning.py
|   +-- plot_setup_vehicle_configuration_space_3d.py
|   +-- practicum3_iv.ipynb
|   +-- reconstruct_via_backtracking.py
|   +-- resources
|   |   +-- a_star_pseudo_code.png
|   |   +-- best_first_pseudo_code.png
|   |   +-- candidate_trajectories.png
|   |   +-- spline_steering_profile.png
|   +-- setup_trajectory_scenario.py
|   +-- trajplanning_obstacle_scenario.py
|   +-- vertices_reachable_in_n_steps.py
+-- practicum3_mp
    +-- git-hash
    +-- practicum3_mp.ipynb
```

git-hash: a5b84ea7