1 Installation Instructions

This section will focus on installing Python 3.8 and *git*, which is the versioning system we will use. There are many ways to setup python. Here, we will use *Miniconda* since that is a painless way to do it.

Also, if you have never worked with *git* before, we suggest you take a look at this simple guide once you've installed it: http://rogerdudler.github.io/git-guide/

There is also an example in assignment 1 on how to use *git*.

1.1 Windows 10

1.1.1 Install Miniconda

- Download miniconda: https://docs.conda.io/en/latest/miniconda.html for Python 3.8. Usually the 64-bit version is correct.
- Open the *Anaconda Prompt (miniconda3)* you just installed.
- Run the command `python -V`. It should output `Python 3.8.3` or similar.

1.1.2 Install Git

Install *git* from here: https://git-scm.com/download

After installation, close and restart your miniconda prompt. Run `git --version`, this should show you your git version and verifies that the installation worked correctly.

1.2 Linux

1.2.1 Install Miniconda

- Download the Miniconda Linux installer for Python 3.8: https://docs.conda.io/en/latest/miniconda.html Usually the 64-bit version is correct.
- Open the terminal and `cd` into the folder containing *Miniconda3-latest-Linux-x86_64.sh*.
- Run `bash Miniconda3-latest-Linux-x86_64.sh`. Accept the default settings.
- Close and reopen your terminal window to make sure the changes take effect.
- Test your installation by running `conda --version` to see its version.

1.2.2 Install git

We will use the `apt` (Advance Package Tool), which is available in Debian, Ubuntu and related Linux distributions to install *git*. For other Linux distributions, you can find the installation instructions here: https://git-scm.com/download/linux

- Run the following commands in your terminal:

```
sudo apt-get update
sudo apt-get install git
```

- Close and reopen your terminal window to make sure the changes take effect.
- Test your installation by running `git --version` to see its version.

1.3 macOS

1.3.1 Install Miniconda

There are two ways to install Miniconda in macOS.

- Download `Miniconda3 MacOSX 64-bit pkg` for Python 3.8: https://docs.conda.io/en/latest/miniconda.html
- Open *Miniconda3-latest-MacOSX-x86_64.pkg* in the Finder and follow the instructions on screen.
- When installation is complete, test it by opening a terminal and running `conda --version` to see its version.

OR

- Download `Miniconda3 MacOSX 64-bit bash` for Python 3.8: https://docs.conda.io/en/latest/miniconda.html
- Open the terminal and `cd` into the folder containing *Miniconda3-latest-MacOSX-x86_64.bash*.
- Run `bash Miniconda3-latest-MacOSX-x86_64.bash`. Accept the default settings.
- Close and reopen your terminal window to make sure the changes take effect.
- Test your installation by running `conda --version` to see its version.

1.3.2 Install git

You can download the Binary installer for macOS from https://git-scm.com/download/mac

After installation, open the terminal and run `git --version`. This should show you your git version and verify that the installation worked correctly.

2 Set up your environment

Next we will set up conda with a new environment named `mydlenv` which uses Python 3.8. Run the following commands in your Miniconda prompt (Windows) or terminal (macOS, Linux):

```
conda --version
conda update conda -y
conda create --name mydlenv python=3.8 -y
conda activate mydlenv
conda env list
```

The last command should show you all installed environements (including `base` and the activated `mydlenv` environment). This way, you have a separate environment to install everything for this course and avoid possible interference with/from other python projects.

Note: Whenever you start the miniconda prompt/terminal for this course, run `conda activate mydlenv` to switch to the correct environment.

3 Run your first code

We have prepared a hello world git repository for you to test your installation.

- Create a folder to store your code.
- Open the Miniconda prompt (Windows) or terminal (macOS, Linux) and `cd` into the folder.
- If you don't have one yet, get a personal access token
- Clone the public repository: `git clone https://github.com/dl-classroom/ex00-helloworld`
- `cd` into it: `cd ex00-helloworld`
- Run `pip install -r requirements.txt` to install the required package.
- Run `python hello.py` and check the output.

4 Optional: Get an IDE

4.1 PyCharm

It is possible to edit your code in a simple text editor. However, that is a lot harder than using a good tool. One popular IDE which you can download for free is *PyCharm*.

Install *PyCharm Community Edition* from here: https://www.jetbrains.com/pycharm/
Start it and create a new python project.

Now you want to be able to debug in PyCharm, but also still run code in your command prompt. So you have to tell PyCharm to use your existing miniconda environment:

- First we have to choose the Python Interpreter:
  - *Windows, Linux*: Go to File → Settings → Project → Python Interpreter.
  - *macOS*: Go to PyCharm → Preferences → Project → Python Interpreter.
- With your project selected, click the wheel on the right of *interpreter* and select *Add...*
- Select *Conda Environment* on the left and *Existing environment* on the right.
- Click the 3 dots on the right of *interpreter* and input the path to the `python` interpreter of your `myenv` miniconda environment. To get the path:
  - *Linux, macOS*: Activate your conda environment in a terminal with `conda activate myenv` and then run `which python` to find the interpreter used by your environment.
  - *Windows*: Default path is `C:\Users\(yourname)\miniconda3\envs\myenv\python.exe`, unless you had changed it during setup.
- Also select *Make available to all projects* so you can use that environment everywhere.
- Confirm changes.

To find out how PyCharm works, try importing the `ex00-helloworld` as a new project. Make sure the project interpreter is set correctly to your `myenv` environment.

Now under Run → Edit Configurations, setup the `hello.py` file correctly and run it within PyCharm. (The environment and path variable don't need to be changed, but the script path needs to be changed to the location of hello.py) You can also try setting a breakpoint in the main function by clicking next to a line number and debug the script.

Note: It can be very helpful in the long run to tune the settings to your preferences and learn the important hotkeys to become a fast coder.

4.2 Visual Studio Code

Another popular IDE choice is *Visual Studio Code* (VS Code), which you can also download for free.

Install *Visual Studio Code* from here: https://code.visualstudio.com/

VS Code does not support a specific programming language. Instead, it provides a set of generic features which can be specialized by installing plugins. For Python support in VS Code, install the Microsoft Python extension:

- Open Extensions by clicking on the square icon in the right sidebar or by pressing *Ctrl+Shift+X*.
- Search for "Python" and click *Install* on the Python extension from Microsoft.

In VS Code, you can associate each project with a specific Python environment by following these steps:

- Open the command palette by clicking on "View" → "Command palette" or by pressing *Ctrl+Shift+P*.
- Type `Python: Select Interpreter` and press *Enter*.

- Select the desired *Miniconda* environment from the list.

To get some hands-on experience, open `ex00-helloworld` as a new project by clicking on "File" → "Open Folder". Then associate your Python enviroment with the project with the steps described above.

VS Code also provides a built-in terminal which you can open by clicking *Terminal* → *New Terminal*. If the setup was successful, the your Miniconda environment should be activated automatically.

Run the `hello.py` Python script in the terminal by typing `python hello.py`.

VS Code also provides a Python Debugger extension, that you can use to debug your code. You read more about it here.

We recommend PyCharm or Visual Studio Code, as these are in our opinion the best you can get for free.

4.3 Code Together

Another good way to code is doing it together with your teammates. Some IDEs have a built-in-solution, like PyCharm (code-with-me) and VS Code (live-share). A platform-independent solution would be to meet over Zoom or over BigBlueButton to discuss, but coding together is definitely more efficient than just discussing together.

5 How to send in pen and paper solutions

5.1 Optional: Using LaTeX

LaTeX is a professional way of creating documents with math in it. In case you want to use it, we provide a template document for you.

5.1.1 Installation

First, you will need to install a LaTeX compiler and package manager, e.g. MiKTeX. Make sure to install for your local user only, not for all users (this can cause problems).

Then, you will need some IDE to work in. If you're already working in PyCharm, you can use the TeXiFy-IDEA plugin. You can also use a standalone latex IDE like TeXworks (all OS) or TeXnicCenter (Windows only).

Here is a Tutorial to help you get started with LaTeX.

5.1.2 Overleaf

Overleaf allows you to collaboratively edit LaTeX documents in real-time.

5.2 Doing the math by hand

Another option would be to do the math on a piece of paper and send in the scan. Please note that we only allow solutions to be send in as a single PDF, so make sure to compile all your scans into a single PDF document. You can use Microsoft Word (Windows), LibreOffice (all OS) or any other software you like for that.