

Deep Learning (2022-2023) – Laboratory

Prof. Marco Cristani, Federico Girella

Abstract

In this document we will go through the preparation necessary to start the laboratory lessons. We will show you how to prepare your Development Interface, create a Virtual Environment and install the required libraries.

Note that this guide is written using examples taken from Windows operating system, as such, the Linux examples may be inaccurate.

Prerequisites

There are no specific requirements for the lessons, but your PC needs to be capable of running Python code.

Preparing your Development Interface

During this course we will work with Python, as such we need a Development Environment (IDE) capable of running AND debugging Python code. You can choose whichever IDE you prefer, but we will list a couple suggestions down below. Note that the rest of this guide will assume you chose one of these two IDEs.

PyCharm

The lessons will be taught using PyCharm [[Download Page](#)] as it is an IDE specialized in Python Development. The community edition is free for all users, however thanks to your University account, you can access the Professional Version for free, by registering to the website and linking your email address.

This is the recommended choice if you want to follow the lessons step by step and are unsure of how an IDE works.

Visual Studio Code

Visual Studio Code (also known as VSCode) is a free to use IDE developed by Microsoft [[Download Page](#)]. It's lightweight and general-purpose, though it requires some easy setup before being ready for python (namely installing the appropriate extension).

This is the recommended IDE if you want a single IDE for all your programming languages.

Installing a Virtual Environment

Different Python Projects may require different versions of Python and/or libraries that not always are compatible. To avoid this problem, we will create a Virtual Environment specifically for our projects. This way whenever you want to run our code, you will only need to activate it, without worrying about dependencies and compatibility problems.

There are different tools for the job, but the one we will use is MiniConda [[Download Page](#)], a lightweight version of Conda. VirtualEnv is also a viable option, but we will limit this guide to Conda (MiniConda).

Download and Install MiniConda

This section will be divided into two versions, depending on which operating system you are using: Windows or Linux-based (e.g. Ubuntu).

Windows

- Download the appropriate version from the [[Download Page](#)] and run the file.
- Select all the default options until you are asked to “initialize conda by running conda.init”.
- Write yes and enter.
- You are done!
- If everything worked correctly, you should have a new program install called “Anaconda Prompt (Miniconda)”
- Every time you want to edit your environments (installing new libraries for example) you will need to open this “Anaconda Prompt”, we will refer to this as the Conda Terminal
- Run the `conda` command inside the Conda Terminal to confirm everything is working

Linux-Based

- Download the appropriate version from the [[Download Page](#)]
- Open the Terminal and navigate to the downloaded file (to navigate folders, use `ls` to see the current directory content and `cd <folder name>` to access a directory)
- If you are in the correct folder, running the `ls` command should display the downloaded `<Miniconda_file_name>.sh` file
- Assuming the name of the downloaded file is `Miniconda_installer.sh` run the following command

```
sh ./Miniconda_installer.sh
```
- Select all the default options until you are asked to “initialize conda by running conda.init”.
- Write yes and enter
- Once the installation is complete, restart the terminal (close all instances and re-open)
- You should see a `(base)` prefix before the usual location on the terminal.
- Run the `conda` command to confirm everything is working
- When we will refer to the Conda Terminal, we refer to a terminal which is able to run Conda commands. This should now mean all terminal instances that have the `(base)` prefix

Create a Virtual Environment

A virtual environment allows us to create localized instances of Python, each independent from the others, with their own Python versions and libraries. This is particularly handy when working on multiple projects with different requirements.

Say for example that we have a Deep Learning project that requires Python 3.8, and a Visual Intelligence project that requires Python 2.7. We can create two different environments, one called `deeplearning` with python 3.8, and one called `visualintelligence` with python 2.7. When we want to code for Deep Learning, we activate the `deeplearning` environment. When we want to code for Visual Intelligence we activate the `visualintelligence` environment. Easy!

With this brief explanation of what virtual environments do, let's create our own.

- Open a Conda Terminal (remember, for Windows user this means opening the Anaconda Prompt, for Linux users it simply means opening a terminal)
- Type the following command to create a new environment named `'deepLearning'` with python 3.8

```
conda create -n deepLearning Python=3.8
```

- Activate the environment to get ready to install dependencies and/or code
`conda activate deepLearning`

Install the dependencies

In the folder you found this file in, there should be a `requirements.txt` file.

This is a summary of all the libraries we need to install in our environment to be able to run our code.

- Open a Conda Terminal
- Activate our environment

```
conda activate deepLearning
```

- Navigate to the folder containing the `requirements.txt` file
- Run the following command to install the libraries

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

- Done

Install PyTorch

Pytorch is a particular library we will need to do machine learning and it cannot be installed through the `requirements` file.

- Open the [PyTorch Get Started Page](#)

- Select the appropriate options in the table. Here is an example if you run windows:

PyTorch Build	Stable (1.12.1)		Preview (Nightly)		LTS (1.8.2)
Your OS	Linux		Mac		Windows
Package	Conda	Pip		LibTorch	Source
Language	Python			C++ / Java	
Compute Platform	CUDA 10.2	CUDA 11.3	CUDA 11.6	ROCm 5.1.1	CPU
Run this Command:	conda install pytorch torchvision torchaudio cpuonly -c pytorch				

- As the table suggests, we now need to copy the command found at the bottom
- Open a Conda Terminal
- Activate our environment
`conda activate deepLearning`
- Paste the command in the terminal and press enter.
Note: if your PC has a dedicated GPU, you could consider installing a compatible CUDA toolkit to run PyTorch on your GPU. This would speed up some operations, but it's not required for this course

Configure our IDE

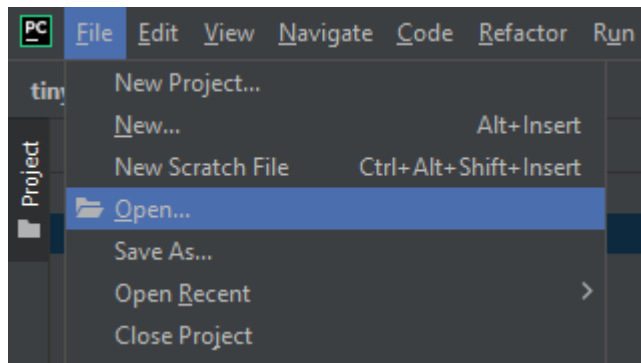
We now have all the prerequisites for starting our project, however a quick setup of our IDE is needed to ensure that everything works fine. This section will be separated in two subsections, one for each possible IDE we suggested at the start.

Before we get into the specifics, we need to create our Project Folder. This folder will contain all the code for our future lessons, as well as images and models trained in the process. As such, you want to create the Project Folder in a location (hard drive) that has enough space and that you are ok to fill with files (don't use the Desktop!)

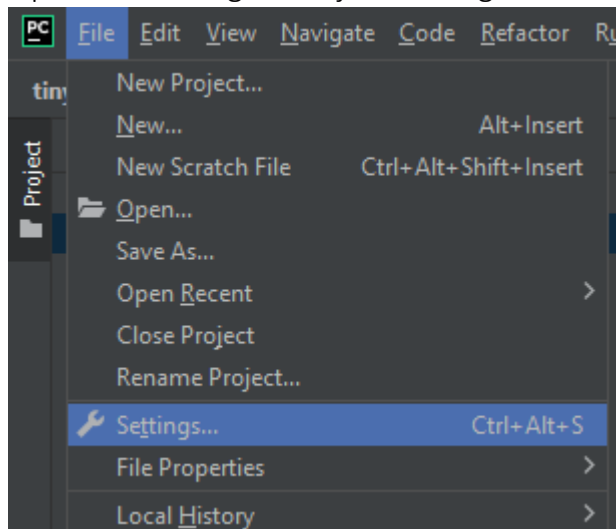
From now on we are going to refer to this folder as the Project Folder, and for simplicity assume it is called `deepLearning`

PyCharm

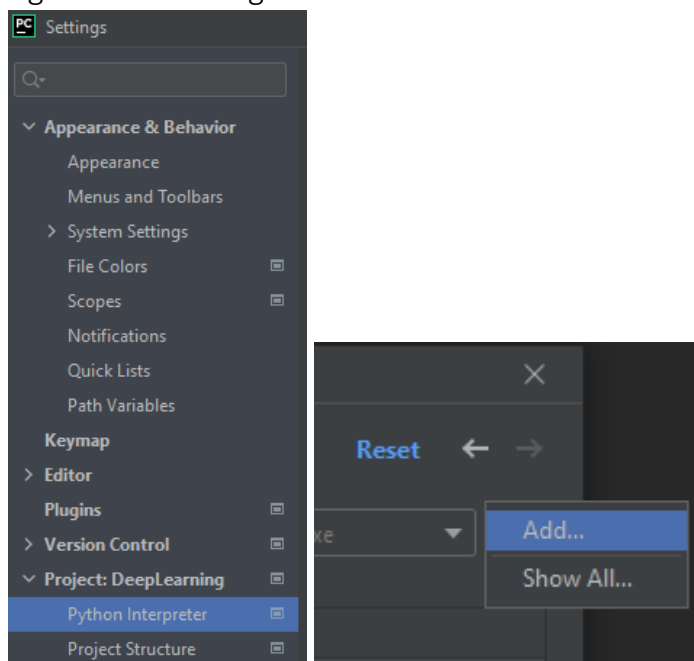
- Open the Project Directory



- Once the Project is opened, we need to tell PyCharm to use our newly created Conda Environment (deepLearning) to run our code.
- Open the Settings > Project Settings



- Add the environment to the available ones by clicking on the Cog in the top right and selecting Add



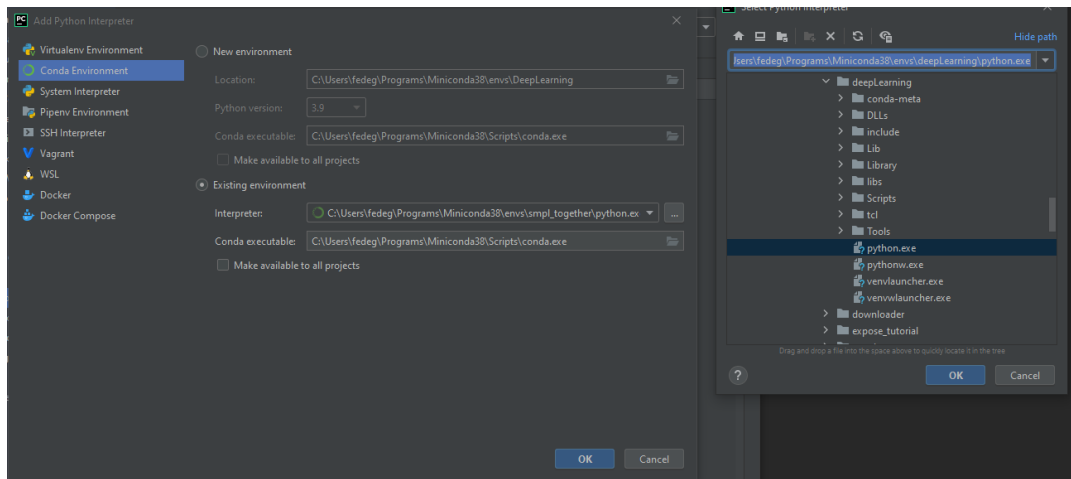
- Click on the [...] button next to Existing Environment in the Conda section.

- Navigate to the location where you have created the environment, select the python file and click OK.
- By default in Windows it's located in:

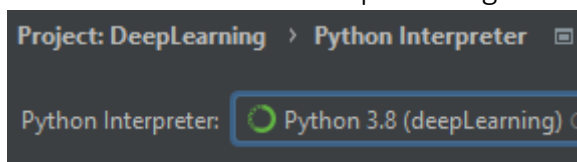
`C:\Users\<your_user>\Programs\Miniconda38\envs\deepLearning\python.exe`

- In Linux it is in

`~/miniconda3/envs/deepLearning/bin/python`



- Choose the deepLearning environment as the project default and click OK




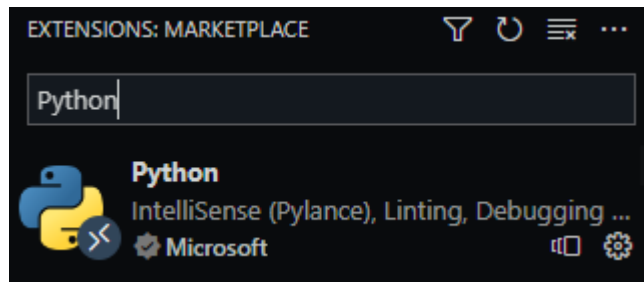
- You are done!

From now on this project will run using the environment we created in the previous steps

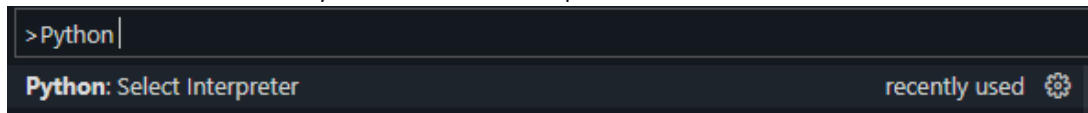
VSCode

VSCode by default it's not compatible with Python. Luckily there is an official extension for it. Let's install it and set the correct environment.

- With VSCode, open our Project Folder
- Open the Extension Page (this icon  on the left bar) and look for Python (the official one from Microsoft) and install it



- Once it's done, press CTRL+SHIFT+P to open the command palette
- Search for Python Select Interpreter and click Enter



- Select the `deepLearning` environment from the available ones
- You are Done! From now on all the code in this Project Folder will run using our environment

Problems and questions

Every system is a bit different from every other, and it's possible that this general guide will not be sufficient. Most problems can be solved by looking on the internet, but in case there are any difficulties that you can't solve on your own, feel free to write to Federico Girella federico.girella@univr.it

NOTE: this is a guide for the academic year 2022/2023. If you are looking at this file in the future, avoid contacting Federico and contact the current tutor.