

# Homework 01

IANNwTF

October 26, 2022

This weeks goal is mostly to set up everything needed for the course, starting Python and Tensorflow on your machine. We put together some instructions to help guide you through the process.

While this homework is a guide only, you are still required to hand in an empty repository. This makes sure you have successfully created the repository and found a group to work with during the semester, while also giving you the opportunity to give feedback.

You may also want to brush up on your programming and math skills; check out the exercises below to get started.

- Homework submission date: Thursday 3.10. (23.59)
- Submission link: <https://forms.gle/ApAZ5ubY8ewgNmJA9>
- This week no homework reviews are required, as there is no previous week to review.

## Contents

<b>1</b>	<b>Setup guide</b>	<b>2</b>
1.1	On the topic of operating systems . . . . .	2
1.2	Conda . . . . .	2
1.3	Other Python distributions . . . . .	3
1.4	Virtual environments . . . . .	3
1.5	TensorFlow . . . . .	4
1.6	Extras . . . . .	5
<b>2</b>	<b>Coding exercises</b>	<b>6</b>
2.1	Objects and modules . . . . .	6
2.2	List comprehension . . . . .	6
2.3	Generators . . . . .	6
2.4	NumPy . . . . .	7
<b>3</b>	<b>Math</b>	<b>7</b>

# 1 Setup guide

**!! NOTE: As with any other guide, make sure that you have read the instructions in their entirety and that you know what you are about to do before executing any of the steps described below!!**

The sooner you get started with setting up your computer, the more time you will have to deal with any problems that you might encounter throughout the process. You *need* to have a functional setup by the time you receive your first real assignment next week. If you need help, please come to one of coding support sessions.

## 1.1 On the topic of operating systems

It is not a requirement for this course per se, but we recommend that you make the jump to Linux if you have not done so already. It is less of a hassle to get Tensorflow running than in native Windows and in an upcoming version of Tensorflow native Windows support will be sunset altogether.

If you are opposed to using Linux or want to get going immediately you could also take a look at [Google Colab](#).<sup>1</sup> It is a **web hosted Jupyter notebook environment** that requires no setup and makes it easy to share your work with others. Even if you would like to use Colab we still advise you strongly to at least try to **setup and use Tensorflow on your own machine at least once**, as it gives you more control and a better understanding of how it works.

## 1.2 Conda

For starters, conda is a package manager for Python. Installing it is going to make your life easier in ways which should become apparent soon. You should do so in one of two ways:

1) Anaconda. This is a Python distribution which comes bundled with conda, as well as hundreds of popular data science packages. With Anaconda, chances are that if you need something you already have it installed on your computer. However, all those packages take up a lot of storage, and most of them you do not even need. You can find instructions on how to install Anaconda [here](#)<sup>2</sup>.

2) Miniconda. If have limited storage available or if you simply like to decide for yourself which packages to install on your computer, then you may want to opt for Miniconda instead. Miniconda provides a basic Python installation, conda, and nothing else. Get Miniconda [here](#)<sup>3</sup>.

Make your choice and follow the installation instructions for your operating

---

<sup>1</sup><https://blog.tensorflow.org/2018/05/colab-easy-way-to-learn-and-use-tensorflow.html>

<sup>2</sup><https://docs.anaconda.com/anaconda/install>

<sup>3</sup><https://docs.conda.io/en/latest/miniconda.html>

system. Once you are done, your command line prompt should now look a bit differently.

A terminal window showing the prompt `(base) [tutor@uos ~]$`. The text is white on a black background.

Figure 1: Conda configures your shell to display the currently active environment.

### 1.3 Other Python distributions

If you are taking this course, chances are that you've worked with Python before and that you already have it installed on your computer. If you are on Linux, Python is probably installed by default. **You do not have to get rid of your existing Python installation to install Anaconda/Miniconda!** When you install Python through conda, that version of Python supersedes the one that is already on your computer, but it does not break or alter it in any way. Your old Python installation is still going to be there and you will be able to switch back to it if you like. Once you have conda installed, your shell will load up your new 'base' environment (more on that in a bit) instead of your system's Python whenever you open up a terminal. You can change this behavior by configuring conda, or you could just run this command

```
$ conda deactivate
```

While your base environment is active to switch back temporarily.

### 1.4 Virtual environments

Conda can do more than just manage packages. It also provides you with virtual environments on top of your 'base' Python installation. Think of virtual environments as containers into which to install your packages. The environments isolate their contents from each other, allowing you to have conflicting packages, different versions of the same package, or even different versions of Python installed on your computer at the same time without it causing any problems. This is useful for a number of reasons:

- Some of your projects might require packages which are incompatible with each other.
- Some of your projects might require different versions of the same package.
- Some of your projects might require older versions of Python
- You can generate a file from which you can then easily recreate your environment on another machine. Or can share that same file with your friends so that they can recreate your setup.

With conda installed, execute the following commands to get a feeling for how virtual environments work in practise:

```
$ conda list
```

Shows you a list of all the packages you have installed in your current environment; try it out! Generally, you should keep your base environment from getting too cluttered.

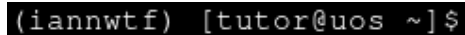
```
$ conda create --name iannwtf
```

Creates a new environment with the name 'iannwtf', and

```
$ conda activate iannwtf
```

Activates it. Your command line prompt should once again update to reflect the change. If you ask conda to list your packages now there should be nothing there. Return to your base environment with

```
$ conda deactivate
```

A terminal window snippet showing the prompt '(iannwtf) [tutor@uos ~]\$' in a dark background with light-colored text. This indicates that the 'iannwtf' environment is currently active.

```
(iannwtf) [tutor@uos ~]$
```

Figure 2: The command prompt helps you remember which environment is currently active.

## 1.5 TensorFlow

Now is the time to finally install TensorFlow. Make sure to activate your new environment once again with

```
$ conda activate iannwtf
```

Then, use conda to install the 'tensorflow'<sup>4</sup> package into your virtual environment, like this:

```
$ conda install tensorflow
```

Now, use the Python script up on Stud.IP to check if everything is working as it should. The script requires the Matplotlib package to work, so you should first install that one:

```
$ conda install matplotlib
```

Download the Python script if you have not already, navigate to it using the command line, and make sure that the correct environment is active before executing the script

```
$ python test_tensorflow.py
```



Figure 3: Running the test script with and without GPU support (the GPU support makes it go faster). Small differences quickly add up when working on larger projects

The script might take a while to finish depending on your hardware, but if it concludes without errors you are good to go!

## 1.6 Extras

Consider using an integrated development environment to code in. For Python we recommend PyCharm Community Edition or VSCode/VSCodium.

You have the choice between handing in your homework as Python scripts (\*.py) or as Jupyter Notebooks (.ipynb). Notebook organizes your code into cells, which you can execute independently of each other, and displays their output underneath. It also supports Markdown, giving you the option to display your code along with formatted text. Jupyter Notebook is automatically installed with Anaconda, but needs to be installed manually if you went with Miniconda instead. Take a look at [this guide](#)<sup>5</sup> to learn more.

Google Colaboratory is a browser-based cloud computing service for running your Python code. It's free, and it provides you with a GPU to speed things up with. Don't support Google if you want to, but for the time being consider making an exception if your hardware is too slow for your liking. Needless to say, you need an internet connection to use Colaboratory. See [here](#)<sup>6</sup> for an overview of its features. In general, we recommend that you split up your projects into scripts and modules as that is considered best practice. Jupyter Notebooks are a nice way to play around with code, to visualize and explain stuff to others; however, larger projects quickly turn unwieldy if kept within a Notebook. We thus recommend sticking to Python modules for actual coding and only using Notebooks for the interactive parts.

---

<sup>4</sup>If you have a CUDA-enabled GPU, you may be able to use it to speed up your deep learning code! Try it out by installing the 'tensorflow-gpu' package instead. If you like you could even create a separate environment to compare the two.

<sup>5</sup><https://realpython.com/jupyter-notebook-introduction/>

<sup>6</sup>[https://colab.research.google.com/github/shranith/Colab-intro/blob/master/Colab\\_intro.ipynb#scrollTo=YQSRkNm-6hlf](https://colab.research.google.com/github/shranith/Colab-intro/blob/master/Colab_intro.ipynb#scrollTo=YQSRkNm-6hlf)

## 2 Coding exercises

### 2.1 Objects and modules

If you are new to Python, you should familiarize yourself with objects and modules. Put simply, a Python module is just a Python file, i.e. any file with the ".py" extension. Create two such files named "cat.py" and "kittyconcert.py" and open them in your editor of choice. Define a "cat" class in "cat.py". This class should have:

- A constructor, which lets you name a "cat".
- A method to make your cats greet each other by first introducing themselves and then addressing a different cat by name, e.g.:  
"Hello I am Kittosaurus Rex! I see you are also a cool fluffy kitty Snowball IX, let's together purr at the human, so that they shall give us food".

When you are done, import "cat.py" into "kittyconcert.py" and create two cats and have them both greet each other.

### 2.2 List comprehension

Understanding list comprehensions lets you define long lists in a manner that is both both elegant and saves you a lot of time. The first time you see a list comprehension it may seem daunting, but they're relatively simple once you understand how they work.

Here's a challenge for you: Using a single line of code, get a list of the squares of each number between 0 and 100! Then do it again, but only include those squares which are even numbers.

Search the internet for "python list comprehension" to get an idea for how to do this.

### 2.3 Generators

Generators are an important tool for training deep neural networks. because they let you iterate over data without cluttering your computer's memory all that much. Implement a generator function which returns a meow the first time you call it, and then twice the number of meows on each consecutive call; e.g.

```
1 Meow
2 Meow Meow
3 Meow Meow Meow Meow
4 Meow Meow Meow Meow Meow Meow Meow Meow
```

etc.. If you are unfamiliar with the concept of generators in Python, search the internet for "python yield" and "python generator" to learn more about it.

## 2.4 NumPy

NumPy is an important library for working with multidimensional arrays, matrices and doing math in general. If you don't have NumPy, use conda to install it now; though if you followed our setup guide, NumPy should have been installed into your virtual environment along with it TensorFlow. See if you can

1. Create a  $5 \times 5$  NumPy array filled with normally distributed (i.e.  $\mu = 0$ ,  $\sigma = 1$ )
2. If the value of an entry is greater than 0.09, replace it with its square. Else, replace it with 42.
3. Use slicing to print just the fourth column of your array.

Check [NumPy's official documentation](https://numpy.org/doc/stable/)<sup>7</sup> for help.

## 3 Math

Both the sigmoid function and its derivative are going to be an important part of this course going forward. Familiarizing yourselves with them now is going to pay off as early as next week. Look up the sigmoid function and try to calculate its derivative. Once you are done, consider the following multivariate function

$$f(x, z, a, b) := y = (4ax^2 + a) + 3 + \sigma(z) + (\sigma(b))^2 \quad (1)$$

Calculate the partial derivatives of  $f$  w.r.t. each of its variables. To be more precise, we want you to calculate

- $\frac{\partial y}{\partial x}$
- $\frac{\partial y}{\partial z}$
- $\frac{\partial y}{\partial a}$
- $\frac{\partial y}{\partial b}$

If you feel especially motivated, you could also look up gradients and the  $\nabla$  ('del'/'nabla') operator. With the results of your previous calculations in hand, you should be able to calculate the gradient  $\nabla f$  w.r.t  $[x, z, a, b]$ .

---

<sup>7</sup><https://numpy.org/doc/stable/>