# Setting up Python

This brief guide will help you set up your Python environment for the course. I have tried to keep it as simple but comprehensive as possible, but if you have any questions or run into any problems, please don't hesitate to ask for help as some of the steps might be different depending on your operating system.

For BIA4, we recommend using Python 3.7 or higher at the moment of writing, the latest version is 3.12.4.

This guide will allow you to:

- Install Python on your system (necessary for the course)
- Set up an editor for writing Python code (necessary for the course)
- Set up a Python environment to use in the course (optional, but recommended, as it is very good practice)

---

**Content of this guide**

---

## Installing Python

The specific steps to install Python depend on your operating system. Below, you will find instructions for Windows, MacOS, and Linux.

- Windows
- MacOS
- Linux

### *Windows*

You can download Python from the Python website.

Once downloaded, run the installer. Make sure to check the box that says "Add python.exe to PATH" before clicking "Install Now".
Follow the instructions in the installer. The safest thing is to leave the default settings as they are; you will

be asked if you want to install several optional features (pip, tcl/tk, etc.), and you should say yes to all of them. pip is probably the most important one, as it will allow you to install additional packages later on.

Once installed you can check the version of Python installed on your system by opening a command prompt (Start -> cmd) and typing the following command:

```
python --version
```

This should return the version of Python you have installed (e.g. Python 3.12.4).

### *MacOS*

You can download Python from the Python website.

Once downloaded, run the installer and follow the instructions within. The safest thing is to leave the default settings as they are.

Once installed you can check the version of Python installed on your system by opening a terminal (Applications -> Utilities -> Terminal) and typing the following command:

```
python --version
```

### *Linux*

Most Linux distributions come with Python pre-installed. You can check the version of Python installed on your system by opening a terminal and typing the following command:

```
python --version
```

If you don't have Python installed, you can install it using your package manager. For example, on Ubuntu, you can install Python by running the following command:

```
sudo apt-get install python3
```

or on Fedora:

```
sudo dnf install python3
```

## Testing your Python installation

Open a terminal (Command Prompt on Windows, Terminal on MacOS, or any terminal emulator on Linux).

You can now test that Python is working by typing the following command:

```
python
```

This should open the Python interpreter, which will look something like this:

```
Python 3.12.4 (<some version info>)
Type "help", "copyright", "credits" or "license" for more information.

>>>
```

Type this:

```
print("This is my first Python program!")
```

And press Enter. You should see the following output:

```
This is my first Python program!
```

Ok, not very exciting, but it means Python is working!

To exit the Python interpreter, type:

```
exit()
```

## Setting up an editor

You can write Python code in any text editor, but it is recommended to use an editor that is specifically designed for writing code.

When installing Python, you should have also installed IDLE, which is a simple editor that comes with Python. You can find it in the Start menu on Windows, or in the Applications folder on MacOS (IDLE is not installed by default on Linux, but you can install it using your package manager).

While IDLE is a good editor for beginners, it is not as powerful as some other editors that are available. Some popular editors include:

- Visual Studio Code - note that Visual Studio Code is a general-purpose editor that can be used for many programming languages, not just Python, so you will have to install the Python extension to get Python support.
- PyCharm
- Sublime Text

The website for each editor will have instructions on how to install it on your system and how to set it up for Python development.

## Packages

Python programming relies heavily on packages, which are collections of code that provide additional functionality. For example, in BIA4 we will make extensive use of the `scikit-image` package, which provides tools for image processing and the `numpy` package, which provides tools for numerical computing.

Python packages are installed using a package manager called `pip`. You can install packages by running the following command in a terminal:

```
pip install <package-name>
```

Sometimes you might encounter errors when installing packages. This can happen for a variety of reasons, but one common reason is that you don't have the necessary permissions to install packages system-wide (this will depend on how Python was installed on your system). In this case, you can install packages in your user directory by adding the `--user` flag to the `pip install` command:

```
pip install <package-name> --user
```

## Setting up a Python environment (optional but recommended)

A Python environment is a self-contained directory that contains a specific version of Python and any packages you need for your project. This is useful because it allows you to have different environments for different projects, each with its own set of packages.

For example, say you are working on two projects, one that uses version 1.0 of a package and another that uses version 2.0. Version 2.0 of the package introduces new functionalities you'd like to use, but it is not backward compatible with version 1.0, so all the code you wrote for the first project will break if you upgrade the package system-wide. By using environments, you can create one environment for each project, each with the appropriate version of the package.

So your projects are isolated from each other, and you can avoid conflicts between different versions of packages.

There are several tools available for managing Python environments, the two most popular ones are `virtualenv` and `conda`. My personal preference is `venv`, which is included with Python.

Creating a new environment with `venv` is easy. Open a terminal (MacOs, Linux) or command prompt (Windows) and navigate to the directory where you want to create the environment.
This can be done by using the `cd` command:

```
cd path/to/directory
```

Now run the following command:

```
python -m venv my_project_env
```

This will create a new directory called `my_project_env` (of course you can use any name you like!) that contains a Python interpreter and a `pip` executable.
If you are using a visual editor you might also be able to create a new environment from within the editor. For example in Visual Studio Code you can create a new environment by opening the command palette (Ctrl+Shift+P) and typing `Python: Create New Environment`.

You now need to activate the environment in order to tell your system to use the Python interpreter and packages in the environment and not the system-wide one. This can be done by running the following command:

On MacOS and Linux:

```
source my_project_env/bin/activate
```

On Windows:

```
my_project_env\Scripts\activate
```

In Visual Studio Code you can select the environment by opening the command palette (Ctrl+Shift+P) and typing `Python: Select Interpreter`.

You should now see the name of the environment in your terminal prompt, indicating that the environment is active. You can now install packages in this environment using `pip` as described above.

## Sharing your environment

If you want to share your environment with someone else, you can export a list of the packages installed in the environment by running the following command:

```
pip freeze > requirements.txt
```

This will create a file called `requirements.txt` that contains a list of all the packages installed in the environment. You can then share this file with someone else, and they can recreate the environment on their system by running the following command:

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

I have created a `requirements.txt` file that lists all the packages you will need for the course workshops. You can download it from Learn.