

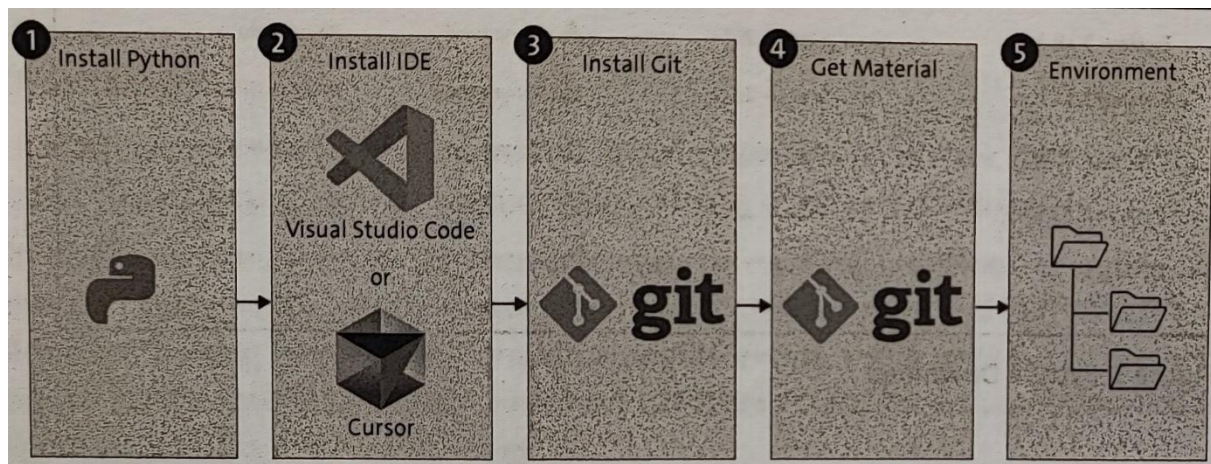
Workshop AI & Machine Learning: System Setup

Contents

Workshop AI & Machine Learning: System Setup	1
Python Installation	2
IDE Installation	2
Visual Studio Code (VS Code)	2
Cursor	2
Git Installation	2
Getting the Source Material.....	3
Setting up Your Local Environment	3
Test your environment.....	4

The figure below shows the steps for setting up your system so you can work with the coding material and run it on your local system.

By following these steps, the Python scripts and notebooks can be run on your local computer. This preparation might seem like a lot but consider that this is a one-time effort and shouldn't take more than 30 minutes. The steps are as follows:



1. Python installation
2. Integrated development environment (IDE) installation
3. Git installation
4. Downloading the course materials
5. Setting up your local environment

Python Installation

The Python installation step is straightforward and can be accomplished through multiple methods. We recommend downloading Python from the official website at <https://www.python.org/downloads>. If you scroll down the download page, you'll find multiple different versions.

The scripts and code in this course were developed on Python version 3.12.11. To avoid any issues, you should stick to the same version. Click the Download button and follow the installation instructions. Nothing specific needs to be considered, so you can go with the default settings.

IDE Installation

An IDE is essential for efficiently writing and managing code in Python projects. Some popular choices include the following:

Visual Studio Code (VS Code)

This lightweight and highly customizable IDE supports many different programming languages, which is one reason why it is a favorite. VS Code is open source and free at <https://code.visualstudio.com>. Its large community provides thousands of extensions.

Cursor

Cursor is a fork of VS Code, which means that it is very close relative. But the main difference with Cursor is that it already has AI power included. With this AI-enhanced code editor, you become much faster and much efficient in coding. Cursor leverages generative AI capabilities, and you can get coding assistance via OpenAI's Codex, GPT, or Anthropic's Claude.

Cursor features code completion, code explanation, bug fixing, refactoring, and interactive chatting to speed you up. It has a free version and a paid version. You can learn more at <https://www.cursor.com>.

If you already have a preference and experience with VS Code, I won't stop you. But if you're a beginner and looking for good advice, I recommend going with Cursor.

Git Installation

Git is an essential tool for version control, allowing developers to track changes in their code, collaborate with others, and maintain a history of project development. Git helps you manage your codebase efficiently by enabling branching, merging, and reverting changes. Git integrates seamlessly with platforms like GitHub, GitLab, and Bitbucket, making remote collaboration straightforward. You'll need Git in the next step to get the course's material. Download Git from <https://git-scm.com/downloads>. Select the

download corresponding to your operating system. Many choices are available, but you can go with the default settings during installation.

Getting the Source Material

The notebooks, images and datasets for this course are hosted on GitHub. You can find the material at the following link:

https://github.com/jdecorte/Workshop_AI_MachineLearning

When you navigate to this page, you should see all the material, as shown in the figure below. You can get the code in two ways by clicking the green Code button.

- Download as ZIP
- Clone using web URL

If you have no experience with Git, go with the ZIP option. Download the ZIP file and extract it to a folder of your choice on your computer. But if you had points of contact with Git, you could clone the repository by running the following from the command line in a folder of your choice:

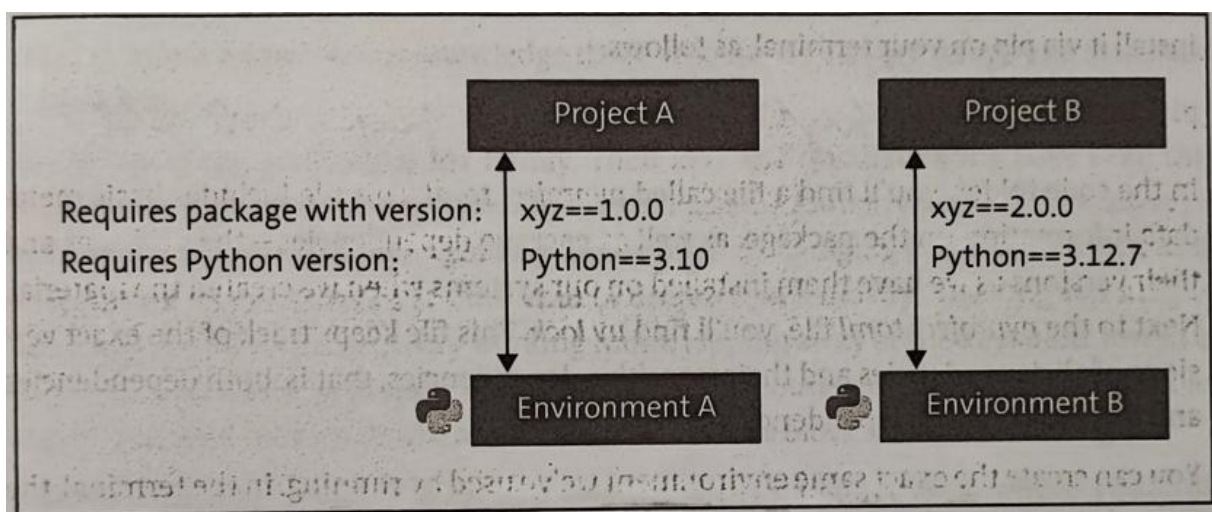
```
git clone https://github.com/jdecorte/workshop_AI_MachineLearning.git
```

Possibly you will have to switch MS Defender temporarily off.

This command will clone the repository to your computer.

Setting up Your Local Environment

A Python environment is an isolated workspace where you can install and manage dependencies for a specific project. This environment acts as a sandbox, ensuring that the libraries and tools used in one project don't interfere with others on your system.



While Project A requires a specific package xyz version 1.0.0 and Python version 3.10, you might also need to work on a different project with completely different requirements.

Let's say, for instance, that Project B requires the same package xyz, but in the newer version 2.0.0, and a different Python version, maybe 3.12.7 instead. If you've only set up a single environment for all your projects, you would run into issues because of the incompatibility between the two projects.

The gold standard is to set up an environment for each project you work on. This approach consumes more space on your hard disk, but you'll be on the safe side and avoid negative side effects between your projects. Different tools for managing your Python environments exist but we go for the modern uv.

uv is a less popular but emerging tool, it focuses on creating ultralight virtual environments. Due to its implementation in Rust, it is blazing fast compared to other environmental tools.

You can find many different options to install uv on the developer page <https://docs.astral.sh/uv/getting-started/installation/#standalone-installer>. You can install it via pip on your terminal, as follows:

```
pip install uv (or python -m pip install uv if that doesn't work).
```

In the code folder of the repository, you'll find a file called `pyproject.toml`. This file includes basic meta-data information on the package, as well as package dependencies - the packages and their versions as I have them installed on my system when I created the material.

Next to the `pyproject.toml` file, you'll find `uv.lock`. This file keeps track of the exact versions of all dependencies and their transitive dependencies, that is, both dependencies and dependencies of dependencies.

You can create the exact same environment I've used by running, in the terminal, the following command:

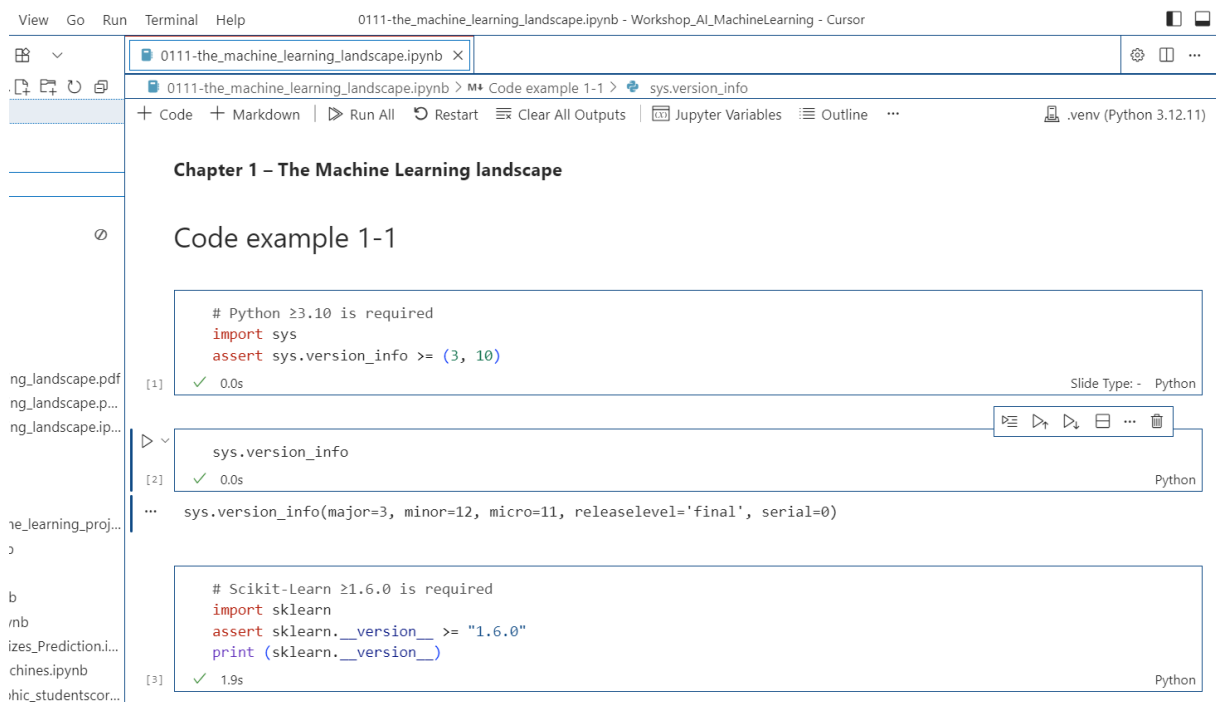
```
uv sync
```

This command will download and install all dependencies into a subfolder named `.venv`, all at once. If for whatever reason it does not work on your computer, don't worry. Open the `pyproject.toml`, look for the section on dependencies, and install the packages manually. You can install these packages by running

```
uv add packagename
```

Test your environment

You can now open your `Workshop_AI_MachineLearning` folder in cursor. Navigate to the file `0111-the_machine_learning_landscape.ipynb` :



Choose `.venv` (Python 3.12.11) as the Python kernel and click “Run All”. All cells in the notebook should now be executed.

Any problems that you experience while executing these instructions can be discussed during the workshop.