Managing a Python project and its *Python Virtual Environment* (PVE) with UV

Learning outcomes:
- Learn how to manage a Python project and its Python Virtual Environment (PVE) using the project manager **uv**.

Expected duration:
- 30 minutes (depending on your Internet connection).

Summary

Interest

The state of the art in Python programming (Data processing, Machine Learning…) is to develop each project within a **Python Virtual Environment** (PVE) which provides a dedicated and persistent environment containing a specific installation of Python:

- independent of other Python installations likely to coexist on the same machine,
- independent of computer updates.

A PVE is based on a dedicated disk tree that houses the desired version of the Python interpreter and all the specific versions of the Python modules needed for the project. You can create, duplicate, delete and recreate a PVE very easily, without impacting other Python installations possibly present on your computer.

Tools

The most often used tools until know to create PVE were:

- the `conda` command, available if you have installed miniconda or Anaconda on your computer
- the `venv` Python module (see venv)
- the `poetry` tool (see poetry.org).

Recently a new tool has been unveiled to manage Python project within a PVE : uv.

It is considerably faster and easier to use than the previous tools: so let’s see how to use **uv** for our Python project.

▸ How to install **uv** on your computer

▸ 1 – Windows

Copy/paste the command bellow in a powershell windows:

```
powershell -ExecutionPolicy ByPass -c "irm https://astral.sh/uv/install.ps1 | iex"
```

▸ 2 – Mac OS & GNU/Linux

Copy/paste the command bellow in a terminal:

```
curl -LsSf https://astral.sh/uv/install.sh | sh
```

If you don't have `curl` installed on your system, then you can use `wget` as shown below:

```
wget -qO- https://astral.sh/uv/install.sh | sh
```

▸ 3 – Upgrading to the Latest uv version

If **uv** is already installed on your computer and you would like to be up to date with the latest version, then you can run the following command:

```
$ uv self update
info: Checking for updates…
success: Upgraded uv from v0.8.0 to v0.8.5! https://github.com/astral-sh/uv/releases/
tag/0.8.5
```

▸ Install the “AI-ML_at_ENSPIMA” project with **uv**

▸ 1 – get the “AI-ML_at_ENSPIMA” GitHub repository

- Open the Git repository https://github.com/cjlux/AI-ML_at_ENSPIMA.
- Download the ZIP archive with the button 

.
- Extract the directory AI-ML_at_ENSPIMA-master from the ZIP archive somewhere in your working tree.
- Rename the directory **AI-ML_at_ENSPIMA-master** as **AI-ML_at_ENSPIM**A.
- With the file manager, copy the path of the directory **AI-ML_at_ENSPIMA** on your computer : we will denote this path in the rest of the document as **<path_of_AI-ML_at_ENSPIMA>**.

▸ 2 – Install the PVE & the python modules with uv

Open a console (Windows: a *powershell* window, Mac/Linux: a *terminal*) and go to the directory of the project with the command cd (*change directory*):

```
cd <path_of_AI-ML_at_ENSPIMA>
```

Then install all the required Python modules with the command:

```
uv sync
```

that's all. This will:

- Create the PVE in the subdirectory **.venv**
- Install all the Python modules listed in the file **pyproject.toml**, including all the dependencies

▸ 3 – Windows post-installation

For the Windows platform you must complete the installation with this command:

```
uv pip install tensorflow==2.16.*
```

It will install an operational version of the tensorflow module.

▸ Useful uv commands

command	description
uv pip list	Lists the python packages of the virtual environmet
uv sync	Installs all the Python packages as listed in the **pyproject.toml** file.
uv self update	Updates uv to the mast bersion.