

# Create a *Python Virtual Environment* (PVE)

## Learning outcomes:

- Learn how to create and use a Python Virtual Environment (PVE) with the *conda* command.

## Expected duration:

- 30 minutes (depending on your Internet connection).

## Summary

► Interest	1
► Tools	1
► Understanding how to create a PVE with conda	2
► Install a PVE under Windows	4
► Install a PVE under macOS	6
► Install a PVE under GNU/Linux	8
► Useful commands	10

### ► Interest

The state of the art in Python programming (Data processing, Machine Learning...) is to work within a **Python Virtual Environment** (PVE) to encapsulate each project in 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 version of the Python interpreter and specific versions of the Python modules that you need for your project. You can create, delete and re-create a PVE very easily, without impacting other Python installations possibly present on your computer.

### ► Tools

The two most often used tools to create PVE are:

- the **conda** command, available if you have installed [miniconda](#) (or [Anaconda](#)) on your computer
- the **venv** Python module (see [venv](#)).

The advantage of *miniconda* for numerical computation is that it transparently installs the [MKL](#) library which provides Intel processors optimization for linear algebra libraries ([BLAS](#), [LAPACK](#) ...) that determine the performance modules like *numpy*, *tensorflow*...

Another advantage of *miniconda* is that you can create PVE of any version : for example with the last *miniconda Python 3.10* version you can create a “Python 3.8 PVE”, or a “Python 3.7 PVE” or even a “Python 2.7 PVE”... and they can all coexist on the same computer!

## ► Understanding how to create a PVE with conda

Prior to the creation of a PVE, you will have to install the *miniconda* package on your computer. You will see this later in the present document, in the section [\[Work do to\]](#) for your operating system (Linux, macOS or Windows).

>>> **For now the goal is just to understand the steps for creating a PVE with the *conda* command.**

With the *conda* command you can create & configure as many PVE as you want following the 4 steps procedure explained bellow.

Don't do the job **now**!

Just understand the commands below, do the job **later** for real in the section [\[Work do to\]](#).

### 1/ How to create a PVE

```
conda create -n <pve_name> pip python=<version>
```

- **<pve\_name>** is the (free) name of your PVE, often a mnemonic like *pyml* (for *Python machine learning*) or *tf2* (for a project under tensorflow2) or *drl* ...
- **<version>** is the version of Python you want to install in your PVE (for example 3.6 or 3.6.8 or 3.8...)

### 2/ How to activate a PVE

```
conda activate <pve_name>
```

Activating a PVE named **drl** results in the prompt being prefixed with the string (**<pve\_name>**) :

- [Windows] : a prompt like **C:\Users\LOGNAME>** becomes: **(drl) C:\Users\LOGNAME>**.
- [GNU/Linux]: a prompt like **logname@host \$** becomes **(drl) logname@host \$**.
- [macOS]: a prompt like **Mac:~ LOGNAME\$** becomes **(drl) Mac:~ LOGNAME\$**.

### 3/ How to install the desired Python modules in a PVE with a YAML file

You can use a YAML file listing the modules to install (and possibly their version), for example :

```
name: <pve_name>
channels:
- defaults
dependencies:
- python=3.8
- tensorflow==2.9.*
- pandas
- matplotlib
- opencv
- jupyter
- notebook
```

to install all the Python modules listed in the file use the command:

```
conda env update -n <pve_name> - -file myfile.yml
```

## 4/ How to add “manually” Python modules to the PVE

The important point is that the PVE concerned must already be activated.

In case you miss one particular module in your activated PVE, you can install a modules by hand, with the PVE activated by typing:

```
conda install <module>
```

```
conda install <module==version>
```

if you want to force the module version

- this command downloads and installs the Python module named **<module>** within the activated PVE. Not all the Python modules have a **conda** version.

```
pip install <module>
```

```
pip install <module==version>
```

if you want to force the module version

- this command downloads and installs the Python module named **<module>** within the activated PVE. All the Python modules are available with the **pip** installer.

The drawback is that mixing **conda install ...** and **pip install ...** can lead to conflicts between the installed modules.

Now you can jump to the section corresponding to your OS, Windows, macOS or GNU/Linux.

## ► Install a PVE under Windows

### ► How does a Virtual Environment work

The installation of the *miniconda3* package under Windows creates a specific version of the terminal (aka the “*command windows*”) named the “*Anaconda Prompt*”. In this terminal, the `PATH` environment variable is modified to first reference the directory containing the conda executable like `C:\Users\LOGNAME\Miniconda3\condabin` for example.

When you activate the PVE `pve_name`, The `PATH` environment variable is modified again to first reference the PVE root directory, like `C:\Users\LOGNAME\Miniconda3\envs\pve_name`:

- all the Python-related commands (*python*, *pip* ...) are first searched under the PVE root directory tree,
- any installation of a Python module with *conda* or *pip* installs the module under the PVE root directory.

### ► Do I need to install miniconda ?

If you don’t find the *Anaconda Prompt* application when searching “*anaconda prompt*” in the Windows search bar, then you need to install miniconda3 else you just have to update the *conda* command.



To update conda of an existing installation, open an *Anaconda Prompt* window and type:

```
conda update -n base -c defaults conda -y
```

Now you can jump to the section ►2 – Create a PVE dedicated to the DRL practical work.

## Work to do in 3 steps

### ► 1 – Installation of miniconda3

If you don’t find the *Anaconda Prompt* application, download the last version of *miniconda3* from [doc.conda.io](https://docs.conda.io) (take care to choose the 64 bits version, unless your PC is a 32 bits one... which is very unlikely). Pay attention to these points for installing *miniconda3*:

- the Miniconda3 directory path (installation directory) must contain no space nor accented character (the default path on Windows is often: `C:\Users\LOGNAME\Miniconda3\`). See this frequently asked question [here](#) :

**In what folder should I install Anaconda on Windows?**

We recommend installing Anaconda or Miniconda into a directory that contains only 7-bit ASCII characters and no spaces, such as `C:\anaconda`. Do not install into paths that contain spaces such as `C:\Program Files` or that include `Unicode` characters outside the 7-bit ASCII character set. This helps ensure correct operation and no errors when using any open-source tools in either Python 3 or Python 2 conda environments.

- Install *miniconda3* “*just for me*”.
- Keep unchecked the option “*Add Miniconda to my PATH environment variable*”.
- If you don’t have any other version of Python installed on your computer you can check the option “*Register Miniconda3 as my default Python ...*” else uncheck this option.

- At the end of the installation answer *yes* to the question “*Do you wish the installer to initialize Miniconda3 by running conda init? [yes | no]*”
- Advice: you can disable the automatic launch of the PVE (**base**) by typing the command in a terminal (an **Anaconda Prompt** window):

```
conda config --set auto_activate_base false
```

Now it's done. If you want to check your installation type the command ‘**conda info**’: you should get no error in return and see information on your *miniconda3* installation displayed on the screen.

## ► 2 – Create a PVE dedicated to the DRL practical work


With *conda* available on your computer, create and activate the PVE named **drl** to work with Python 3.10 :

```
(base) C:\Users\LOGNAME> conda create -n drl pip python=3.10 -y
...some stuff...

(base) C:\Users\LOGNAME> conda update -n base -c defaults conda -y
...some stuff...

(base) C:\Users\LOGNAME> conda activate drl
(drl) C:\Users\LOGNAME>
```

Now install the Python modules required for the practical work:

- Open the Git repository [https://github.com/cjlux/DRL\\_at\\_ENSEIRB-MATMECA](https://github.com/cjlux/DRL_at_ENSEIRB-MATMECA)
- Download the ZIP archive with the button 
- Extract the directory **DRL\_at\_ENSEIRB-MATMECA-master** from the ZIP archive somewhere in your working tree.
- Rename **DRL\_at\_ENSEIRB-MATMECA-master** as **DRL\_at\_ENSEIRB-MATMECA**.
- Install all the modules with these commands:

```
(drl) C:\Users\LOGNAME> cd <path_of_folder DRL_at_ENSEIRB-MATMECA>
(drl) <path_of_DRL_at_ENSEIRB-MATMECA> conda env update -n drl --file drl.yml
...some stuff...
```

Now you can run **jupyter notebook** or **jupyter lab** to work with the notebooks...

## ► Install a PVE under macOS

If you cannot run the `miniconda` command in a terminal then you need to install the `miniconda3` package else you can skip the `miniconda3` installation.

## ► How does a Virtual Environment work

The installation of `miniconda3` modifies the `.bashrc` file in your home directory. The `PATH` environment variable is modified to mention first the directory containing the `conda` command: `/Users/<logname>/opt/miniconda3/condabin`).

When you activate the PVE `pve_name`, The `PATH` variable is modified again to reference first the PVE root directory `/Users/<logname>/opt/miniconda3/envs/pve_name`:

- all the Python-related commands (`python`, `pip` ...) are first searched under the PVE root directory tree,
- any installation of a Python module with `conda` or `pip` installs the files under the PVE root directory.

## Work to do in 3 steps

### ► 1 – Installation of `miniconda3`

If the `conda` command does not work in a terminal, download and install miniconda on your computer from <https://docs.conda.io/en/latest/miniconda.html>. Pay attention to these points:

- the *installation path* of the `miniconda3` directory must not contain any spaces or accentuated character. (the default installation path on MacOS is: `/Users/<logname>/opt/miniconda3`)
- At the end of the installation answer yes to the question “Do you wish the installer to initialize Miniconda3 by running `conda init`? [yes | no]”
- Start a new terminal or type the command `source ~/.bashrc` to inherit changes from your `.bashrc` file: the `conda` command now becomes available in the terminal.
- Advice: you can disable the automatic launch of the PVE (`base`) by typing the command:

```
conda config --set auto_activate_base false
```

Now it's done. If you want to check your installation, launch a new terminal and try the command `conda info`: you should get no error in return and see information on your `miniconda3` displayed on the screen.

### ► 2 – Create a PVE dedicated to the DRL practical work


With `conda` available on your computer, create and activate the PVE named `drl` to work with Python 3.10 :

```
(base) Mac:~ logname$ conda create -n drl pip python=3.10 -y
...some stuff ...

(base) Mac:~ logname$ conda update -n base -c defaults conda -y
... some stuff...

(base) Mac:~ logname$ conda activate drl
(drl) Mac:~ logname$
```

Now install the Python modules required for the practical work:

- Open the Git repository [https://github.com/cjlux/DRL\\_at\\_ENSEIRB-MATMECA](https://github.com/cjlux/DRL_at_ENSEIRB-MATMECA)
- Download the ZIP archive with the button 
- Extract the directory [DRL\\_at\\_ENSEIRB-MATMECA-master](#) from the ZIP archive somewhere in your working tree.
- Rename [DRL\\_at\\_ENSEIRB-MATMECA-master](#) as [DRL\\_at\\_ENSEIRB-MATMECA](#).
- Install all the modules with these commands:

```
(drl) Mac:~ logname$ cd <path_of_folder DRL_at_ENSEIRB-MATMECA>
(drl) Mac:~ logname$ conda env update -n drl --file drl.yml
...some stuff...
```

Now you can run [jupyter notebook](#) or [jupyter lab](#) to work with the notebooks...

## ► Install a PVE under GNU/Linux

If you cannot run the `miniconda` command in a terminal then you need to install the `miniconda3` package else you can skip the installation.

## ► How does Virtual Environment work

The installation of `miniconda3` modifies the `.bashrc` file in your home directory. The `PATH` environment variable is modified to first reference the directory containing the `conda` command: `/home/<logname>/miniconda3/condabin` on Ubuntu.

When you activate the PVE `pve_name`, The `PATH` variable is modified again to first reference the PVE root directory `/home/<logname>/miniconda3/envs/pve_name`:

- all the Python-related commands (`python`, `pip` ...) are first searched under the PVE root directory tree,
- any installation of a Python module with `conda` or `pip` installs the files under the PVE root directory.

## Work to do in 3 steps

### 1 – Install miniconda

If the `conda` command does not work in a terminal, download and install miniconda on your computer from <https://docs.conda.io/en/latest/miniconda.html>. Pay attention to these points:

- The *installation path* for the `miniconda3` directory must not contain spaces or accentuated characters (the default installation path on Ubuntu is: `/home/<logname>/miniconda3`)
- At the end of the installation answer yes to the question “Do you wish the installer to initialize Miniconda3 by running `conda init`? [yes | no]”
- Start a new terminal or type the command `source ~/.bashrc` to inherit changes from your `.bashrc` file: the `conda` command now becomes available in the terminal.
- Advice: you can disable the automatic launch of the PVE (`base`) by typing the command:

```
conda config --set auto_activate_base false
```

Now it's done. If you want to check your installation launch, a new terminal and try the command `conda info`: you should get no error in return and see a information on `miniconda3` displayed on the screen.

### ► 2 – Create a PVE dedicated to the DRL practical work


With `conda` available on your computer, create and activate the PVE named `drl` to work with Python 3.10 :

```
(base) logname@host $ conda create -n drl pip python=3.10 -y
...some stuff ...
...
(base) logname@host $ conda update -n base -c defaults conda -y
... some stuff...

(base) logname@host $ conda activate drl
(drl) logname@host $
```



Now install the Python modules required for the practical work:

- Open the Git repository [https://github.com/cjlux/DRL\\_at\\_ENSEIRB-MATMECA](https://github.com/cjlux/DRL_at_ENSEIRB-MATMECA)
- Download the ZIP archive with the button 
- Extract the directory `DRL_at_ENSEIRB-MATMECA-master` from the ZIP archive somewhere in your working tree.
- Rename `DRL_at_ENSEIRB-MATMECA-master` as `DRL_at_ENSEIRB-MATMECA`.
- Install all the modules with these commands:

```
(drl) logname@host $ cd <path_of_folder DRL_at_ENSEIRB-MATMECA>
(drl) logname@host $ conda env update -n drl --file drl.yml
... some stuff...
```

Now you can run [jupyter notebook](#) or [jupyter lab](#) to work with the notebooks...

When running PyBullet if you get an error like “**libGL error: MESA-LOADER: failed to open iris**” you need to change the library `libstdc++.so` in the `drl` PVE :

```
(drl) logname@host $ cd <path_of_miniconda>/envs/drl/lib
(drl) logname@host $ mkdir backup
(drl) logname@host $ mv libstdc++* backup/
(drl) logname@host $ cp /usr/lib/x86_64-linux-gnu/libstdc++.so.6.0.30 libstdc++.so.6
(drl) logname@host $ ln -s libstdc++.so.6 libstdc++.so
(drl) logname@host $ ln -s libstdc++.so.6 libstdc++.so.6.0.30
```

>>> Adjust the last number of `libstdc++.so.6.0.xx` to whatever is the number ‘xx’ of your `libstdc++` library in `/usr/lib/x86_64-linux-gnu/`.

## ► Useful commands

<i>command</i>	<i>description</i>
<code>conda info</code>	Display informations about <i>conda</i>
<code>conda env list</code>	List the PVEs known by <i>conda</i>
<code>conda deactivate</code>	Deactivate the currenttly activated PVE
<code>conda activate &lt;pve_name&gt;</code>	Activate the PVE named <code>&lt;pve_name&gt;</code>
<code>conda list</code>	conda view of the list of installed packages <b>for the activated PVE</b>
<code>pip list</code>	pip view of the list of installed packages for the activated PVE
<code>conda search &lt;name&gt;</code>	Find versions of the Python module named <code>&lt;name&gt;</code> compatible with the activated PVE
<code>conda env remove -n &lt;pve-name&gt;</code>	Removes all the files of the PVE <code>&lt;pve-name&gt;</code>