Spring 2025: Programming for chemistry

(lec18: using Python CLI & WEB)

Davide Ceresoli (davide.ceresoli@cnr.it)

Instituto di Scienze e Tecnologie Chimiche "G. Natta" (CNR-SCITEC)

December 11, 2025

Outline

Virtual environments

- Python has a huge collection (∼300,000) of thirdy-part packages, listed at `http://pypi.org`
- Many are actively maintained, other not...
- ...dependency hell: packages might require the installation of **specific** NumPy/SciPy versions
- **Virtual environments** keep packages and their dependencies separated!
- You might want to try different versions of the same package (i.e. stable vs development).

Virtual environments

Both `conda` and standard Python offer the possibility to create *venvs*:

conda

- create venv: `conda create -n prog4chm`
- activate venv: `conda activate prog4chem`
- install packages: `conda install packages...`
- deactivate venv: `conda deactivate`

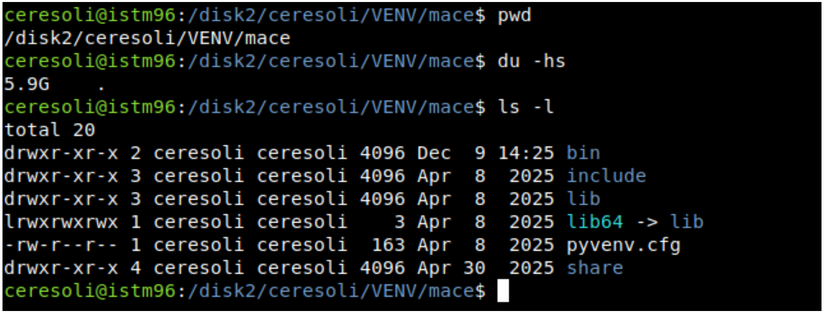
Python

- create venv: `python -m venv prog4chm`
- activate venv: `. ./prog4chem/bin/activate`
- install packages: `pip install packages...`
- deactivate venv: `deactivate`

Virtual environments

- Of course, it is necessary to type this commands in the terminal!
- In Linux/MacOS/WSL the terminal runs a *Unix shell*, i.e. Bash or Zsh
- In Windows, the terminal runs either the DOS or the Powershell
- Exercise: create a `prog4chem` environment and install the following packages: `numpy`, `scipy`, `matplotlib`, `streamlit`
- Get familiar with the CLI (Command Line Interface), create/activate/deactivate venvs, etc...

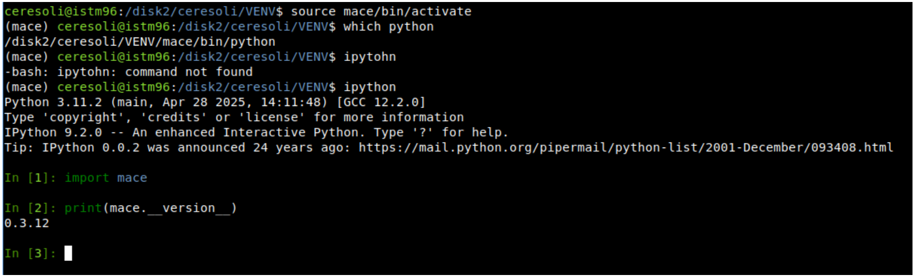
What's in a virtual environment



```
ceresoli@istm96:/disk2/ceresoli/VENV/mace$ pwd
/disk2/ceresoli/VENV/mace
ceresoli@istm96:/disk2/ceresoli/VENV/mace$ du -hs
5.9G    .
ceresoli@istm96:/disk2/ceresoli/VENV/mace$ ls -l
total 20
drwxr-xr-x 2 ceresoli ceresoli 4096 Dec  9 14:25 bin
drwxr-xr-x 3 ceresoli ceresoli 4096 Apr  8  2025 include
drwxr-xr-x 3 ceresoli ceresoli 4096 Apr  8  2025 lib
lrwxrwxrwx 1 ceresoli ceresoli    3 Apr  8  2025 lib64 -> lib
-rw-r--r-- 1 ceresoli ceresoli  163 Apr  8  2025 pyvenv.cfg
drwxr-xr-x 4 ceresoli ceresoli 4096 Apr 30  2025 share
ceresoli@istm96:/disk2/ceresoli/VENV/mace$
```

What's in a virtual environment

- Once you activate a venv, the prompt changes and indicates the active one
- In the `bin` directory you will find a link to the Python interpreter
- Packages are searched in the lib/python-3.11/site-packages directory



```
ceresoli@istm96:/disk2/ceresoli/VENV$ source mace/bin/activate
(mace) ceresoli@istm96:/disk2/ceresoli/VENV$ which python
/disk2/ceresoli/VENV/mace/bin/python
(mace) ceresoli@istm96:/disk2/ceresoli/VENV$ ipytohn
-bash: ipytohn: command not found
(mace) ceresoli@istm96:/disk2/ceresoli/VENV$ ipython
Python 3.11.2 (main, Apr 28 2025, 14:11:48) [GCC 12.2.0]
Type 'copyright', 'credits' or 'license' for more information
IPython 9.2.0 -- An enhanced Interactive Python. Type '?' for help.
Tip: IPython 0.0.2 was announced 24 years ago: https://mail.python.org/pipermail/python-list/2001-December/093408.html

In [1]: import mace

In [2]: print(mace.__version__)
0.3.12

In [3]: 
```

Python scripts

- Python scripts are text files with `.py` extension, as opposed to Jupyter notebooks which have the `.ipynb` extension
- They are executed from the first to the last line as a whole
- The input source is the keyboard *(standard input)*
- The output is the terminal *(standard output*)
- Both can be substituted by files using *redirection*

Python scripts

- Conventionally, the first line must be with: `#!/usr/bin/env python`
- In Linux/MacOS you can make a script *executable* with:
  `chmod 0755 script.py`
  then launch it simply with:
  `./script.py parameters...`
- In Windows you must type: `python script.py parameters...`
- CLI parameters are found in the `sys.argv[]` list of strings

Exercise

Let's make scripts to solve a second order polynomial

- 1st version: use `input()` to ask for the three coefficients
- 2nd version: get the three coefficients from the command line

Note: `sys.argv[0]` is the name of the script, `sys.argv[1]` will be the first CLI parameter. In both scripts, convert strings to floating point numbers using `float()`

Input/output redirection

- Both scripts can get their input from a file using `<file`...
- The CLI version can output easily the results in a file using `>file` and `>>file`...
- For instance:

```
./solve_interactive.py <coefficients.dat
./solve_cli.py 1 3 -7 >roots.dat
```

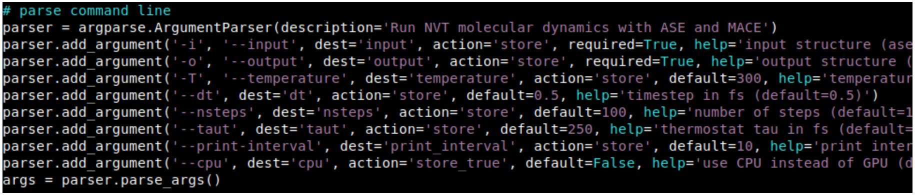
- If you remove every printing message you can create a *filter* script:

```
./solve_filter.py <coefficients.dat >root.dat
```

Argument parsing

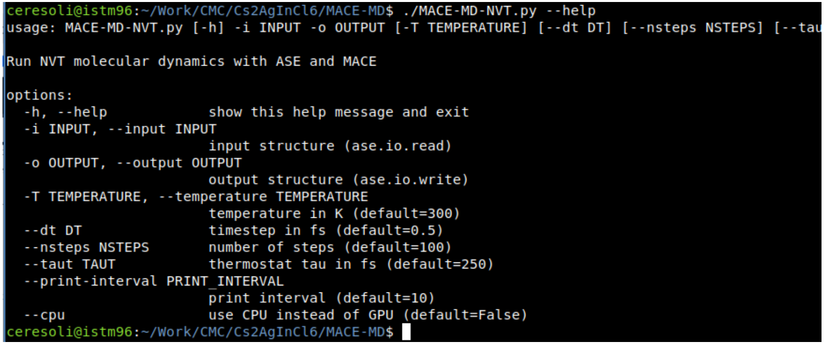
For a more flexible CLI, use the powerful `argparse` module

Let's look at the example script `MACE-MD-NVT.py`



```
# parse command line
parser = argparse.ArgumentParser(description='Run NVT molecular dynamics with ASE and MACE')
parser.add_argument('-i', '--input', dest='input', action='store', required=True, help='input structure (ase
parser.add_argument('-o', '--output', dest='output', action='store', required=True, help='output structure (
parser.add_argument('-T', '--temperature', dest='temperature', action='store', default=300, help='temperatur
parser.add_argument('--dt', dest='dt', action='store', default=0.5, help='timestep in fs (default=0.5)')
parser.add_argument('--nsteps', dest='nsteps', action='store', default=100, help='number of steps (default=1
parser.add_argument('--taut', dest='taut', action='store', default=250, help='thermostat tau in fs (default=
parser.add_argument('--print-interval', dest='print_interval', action='store', default=10, help='print inter
parser.add_argument('--cpu', dest='cpu', action='store_true', default=False, help='use CPU instead of GPU (d
args = parser.parse_args()
```

Argparse



```
ceresoli@istm96:~/Work/CMC/Cs2AgInCl6/MACE-MD$ ./MACE-MD-NVT.py --help
usage: MACE-MD-NVT.py [-h] -i INPUT -o OUTPUT [-T TEMPERATURE] [--dt DT] [--nsteps NSTEPS] [--tau

Run NVT molecular dynamics with ASE and MACE

options:
  -h, --help            show this help message and exit
  -i INPUT, --input INPUT
                        input structure (ase.io.read)
  -o OUTPUT, --output OUTPUT
                        output structure (ase.io.write)
  -T TEMPERATURE, --temperature TEMPERATURE
                        temperature in K (default=300)
  --dt DT               timestep in fs (default=0.5)
  --nsteps NSTEPS       number of steps (default=100)
  --taut TAUT           thermostat tau in fs (default=250)
  --print-interval PRINT_INTERVAL
                        print interval (default=10)
  --cpu                 use CPU instead of GPU (default=False)
ceresoli@istm96:~/Work/CMC/Cs2AgInCl6/MACE-MD$
```

Interlude

High Performance Computing (HPC)

- CPU vs GPU (demo on workstation)
- non-interactive batch jobs (demo on Leonardo@CINECA)

Streamlit (`https://streamlit.io`)

- The `streamlit` package is a super-easy way to create Web applications from your scripts
- It provides *input()*- and *print()-like* functions that work in a browser
- Streamlit apps will be interactive and *reactive!*
- Create a `requirements.txt` file with the list of packages you need
- Then create a `streamlit_app.py` source file

Our first web app

- Activate the `prog4chem` environment, make sure `streamlit` is installed
- Go to the `second_order_webapp` directory...
- ...and type: `streamlit start`

Final exercise

Impress the experimental colleagues

Create a *webapp* to balance chemical reactions!

Conclusions