# Poetry

Man will Wahrheit, man will Wirklichkeit und verdirbt dadurch die Poesie. (Johann Wolfgang von Goethe)

Wim R.M. Cardoen
Email: $(prefix)[at]gmail[dot]com
where
prefix='wcardoen'

November 1, 2023

In the following paragraphs we discuss the `Poetry` tool [Eus18] which handles dependency management and packaging in Python. The installation of `Poetry` requires Python $>$ 3.8, but the tool is available for the `Linux`, `MacOS`, `Windows` OS. All code below was executed on an laptop running the `Ubuntu` 22.04 OS, Python 3.10.12 and `Lmod` 8.7.31.

## Introduction

`Poetry` is a tool which allows you to:

- perform dependency management
- build & package projects
- publish your projects

## I Installation of the Poetry tool

The installation of `Poetry` requires the retrieval of its installation script. By default the `Poetry` executable (`poetry`) will be installed in the directory `$HOME/.local/bin/`. You can install the `poetry` in another location as long as the environmental variable `POETRY_HOME` has been defined.

```
curl -sSL https://install.python-poetry.org >& driver_poetry.py
export POETRY_HOME=$HOME/software/pkg/poetry/1.6.1
python3 driver_poetry.py
```

If you decide to remove the `Poetry` tool, you can proceed as follows:

```
export POETRY_HOME=$HOME/software/pkg/poetry/1.6.1
python3 driver_poetry.py --uninstall
```

Subsequently, a `lua` module file (`LMOD`) was created to load `Poetry` :

```
sleipnir@x1:~$ module load poetry
sleipnir@x1:~$ which poetry
/home/sleipnir/software/pkg/poetry/1.6.1/bin/poetry
sleipnir@x1:~$ poetry --version
Poetry (version 1.6.1)
```

The invocation of the `poetry` executable without any subcommands/options lists all `Poetry` subcommands.

# II Creation of a new Project

Invoking the `poetry new` command followed by a string creates a new project bearing the name of the string.

```
sleipnir@x1:~$ poetry new chpc-demo
  Created package chpc_demo in chpc-demo
```

The new project has the following directory structure:

```
chpc-demo/
    chpc_demo/
         __init__.py
    pyproject.toml
    README.md
    tests/
         __init__.py
```

The newly created python project bears the name `chpc_demo` i.e. the hyphen in the project name has been replaced by an underscore (python does not support hyphen tokens in package names). The newly created project also contains two directories i.e. `chpc_demo` (source code properly) and the directory `tests` (potential test code). By adding the `--src` flag to the `python new` command, the source directory will bear the name `src`. In the latter case, the directory structure `chpc_demo` is to be found under `src`.

```
poetry new --src chpc-demo
```

The newly generated `toml`[TOM23] file `pyproject.toml` contains the minimal amount of information to create a working python package:

```
[tool.poetry]
name = "chpc-demo"
version = "0.1.0"
description = ""
authors = ["Your Name <you@example.com>"]
readme = "README.md"

[tool.poetry.dependencies]
python = "^3.10"

[build-system]
requires = ["poetry-core"]
build-backend = "poetry.core.masonry.api"
```

Some of the blocks in the `toml` file can be created/changed by `poetry` commands (vide infra). However, most fields/blocks can only be manually modified using an editor. In the `pyproject.toml` file the fields `name`, `version`, `description`, `authors` are mandatory. We further notice the field `python = "^3.10"`. `Toml` files adhere to a particular syntax (Semantic Versioning) [SEM23]. According to its rules, `^3.10` implies `3.10` and higher but **not** `4.00` (which currently does not exist). The field name `python` was initialized to `^3.10` (first found by on `$PATH`).

In case you want to use an existing pre-populated directory named `mycode` you need to invoke the `poetry init` command:

```
cd mycode
poetry init
```

# III Add external dependencies

After the creation or initialization of the project existing python packages can be added to the new project. These packages will be inserted to a virtual environment within the new project. First, make sure you have entered the

project directory. You can either explicitly create the virtual environment prior to the addition of the packages i.e. by using the command poetry `poetry env use python3` or by using the absolute path to the executable `poetry env use /usr/bin/python3`.

If you don't create a virtual environment prior to the addition process, a poetry virtual environment will be created associated to the project when you start to add packages. The command `poetry env info` displays information on the current `poetry` virtual environment.

```
sleipnir@x1:~/chpc-demo$ poetry env info
Virtualenv
Python:         3.10.12
Implementation: CPython
Path:           /home/sleipnir/.cache/pypoetry/virtualenvs/chpc-demo-5_MZDma8-
   py3.10
Executable:     /home/sleipnir/.cache/pypoetry/virtualenvs/chpc-demo-5_MZDma8-
   py3.10/bin/python
Valid:          True

System
Platform:   linux
OS:         posix
Python:     3.10.12
Path:       /usr
Executable: /usr/bin/python3.10
```

The command `poetry add` followed by the package names adds packages to the new project. If you do **not** specify the version of the package (using the syntax adhered to by `toml` files) `poetry` wil **always** insert the latest version of a package in the virtual environment and the `toml` file. Unfortunately, this behavior can have serious consequences later down the road for users who install your package. In what follows we decided that `numpy>=1.2` as well as the packages `black` and `mypy` were required. A succesful execution of the command `poetry add numpy@^1.2` required the modification of the block:

```
python = "^3.10"
```

into:

```
python = "<3.13,>=3.9"
```

Subsequently the following commands were invoked:

```
poetry add numpy@^1.2
poetry add black mypy --group=dev
```

The flag `--group=dev` implies that these packages will be added for the development dependency (which will generate a separate block in the `toml` file).

Finally, the command `poetry install` installs the packages in the virtual environment. The command `poetry show --tree` displays info on the installed packages in a tree form.

```
sleipnir@x1:~/chpc-demo$ poetry show --tree
black 23.10.1 The uncompromising code formatter.
├── click >=8.0.0
│   └── colorama *
├── mypy-extensions >=0.4.3
├── packaging >=22.0
├── pathspec >=0.9.0
├── platformdirs >=2
├── tomli >=1.1.0
```

```
└── typing-extensions >=4.0.1
mypy 1.6.1 Optional static typing for Python
├── mypy-extensions >=1.0.0
├── tomli >=1.1.0
└── typing-extensions >=4.1.0
numpy 1.26.1 Fundamental package for array computing in Python
```

Note that `numpy 1.26.1` was installed in the newly created virtual environment. However, the corresponding `toml` file contains:

```
numpy = "^1.2"
```

This requirement will ensure that the installation of our package will **not** overwrite existing installations of `numpy` as long as the installed version of `numpy>=1.2` and `numpy<2.0`.

The new package requires the existence of a binary `genmycdf`. In order to accomodate this feature the following block of code had to be manually inserted in the `toml` file:

```
[tool.poetry.scripts]
genmycdf = "chpc_demo.run_extwiener:run"
```

# IV Accessing the code in the package

The code in the newly created package can be accessed directly by spawning a shell (`poetry shell`):

```
sleipnir@x1:~/chpc-demo$ poetry shell
Spawning shell within /home/sleipnir/.cache/pypoetry/virtualenvs/chpc-demo-5
    _MZDma8-py3.10
. /home/sleipnir/.cache/pypoetry/virtualenvs/chpc-demo-5_MZDma8-py3.10/bin/
    activate
sleipnir@x1:~/chpc-demo$ . /home/sleipnir/.cache/pypoetry/virtualenvs/chpc-demo
    -5_MZDma8-py3.10/bin/activate
(chpc-demo-py3.10) sleipnir@x1:~/chpc-demo$ python3
Python 3.10.12 (main, Jun 11 2023, 05:26:28) [GCC 11.4.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import numpy as np
>>> np.__version__
'1.26.1'
```

or indirectly by invoking the `poetry run` command:

```
sleipnir@x1:~/chpc-demo$ poetry run python3 -c "import numpy as np; print(np.
    __version__)"
1.26.1
```

The executable `genmycdf` can also be tested:

```
sleipnir@x1:~/chpc-demo$ poetry shell
Spawning shell within /home/sleipnir/.cache/pypoetry/virtualenvs/chpc-demo-5
    _MZDma8-py3.10
sleipnir@x1:~/chpc-demo$ . /home/sleipnir/.cache/pypoetry/virtualenvs/chpc-demo
    -5_MZDma8-py3.10/bin/activate
(chpc-demo-py3.10) sleipnir@x1:~/chpc-demo$ python3
Python 3.10.12 (main, Jun 11 2023, 05:26:28) [GCC 11.4.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> quit()
```

```
(chpc-demo-py3.10) sleipnir@x1:~/chpc-demo$ genmycdf --help
usage: genmycdf [-h] --numinter NUMINTER --numpaths NUMPATHS [--kappa KAPPA] [--
    seed SEED] --outfile OUTFILE

Extended Wiener process:

options:
  -h, --help            show this help message and exit
  --numinter NUMINTER, -i NUMINTER
                        #Intervals in [0,1]
  --numpaths NUMPATHS, -p NUMPATHS
                        #Paths/trajectories (i.e. repeats)
  --kappa KAPPA, -k KAPPA
                        Value for kappa [0.35]
  --seed SEED, -s SEED  Seed for the number generator [12345]
  --outfile OUTFILE, -o OUTFILE
                        Output file (.npz)
(chpc-demo-py3.10) sleipnir@x1:~/chpc-demo$ exit
exit
```

# V Build the package

The building of a package is very common if you plan to share your code with the wider world. In order to perform the build, execute `poetry build`:

```
sleipnir@x1:~/chpc-demo$ poetry build
Building chpc-demo (0.1.0)
  - Building sdist
  - Built chpc_demo-0.1.0.tar.gz
  - Building wheel
  - Built chpc_demo-0.1.0-py3-none-any.whl
```

A compressed source code and a wheel file are built. Both can be used to do an installation, although the installation of a wheel file will be faster especially if the new package contains code which needs compilation.

# VI Publish the package

A lot of Python packages can be found on https://pypi.org/. Howver, there is a website to test Python package publishing: https://test.pypi.org. Prior to its use one needs to register and create a token for the package one wants to publish.

After the token has been created on the `https://test.pypi.org` website the current virtual environment needs the information to deliver its content.

```
sleipnir@x1:~/chpc-demo$ poetry config repositories.test-pypi https://test.pypi.
    org/legacy/
sleipnir@x1:~/chpc-demo$ poetry config pypi-token.test-pypi xyz
```

where `xyz` stands for the token provided by `https://test.pypi.org`.

The package can now be published as follows:

```
sleipnir@x1:~/chpc-demo$ poetry publish -r test-pypi
Publishing chpc-demo (0.1.0) to test-pypi
 - Uploading chpc_demo-0.1.0-py3-none-any.whl 100%
 - Uploading chpc_demo-0.1.0.tar.gz 100%
```

## VII Install the newly created package

We can now install our newly created package locally. Make sure you leave the `Poetry` environment.

```
sleipnir@x1:~$ module purge
sleipnir@x1:~$ module load anaconda3
sleipnir@x1:~$ pip install -i https://test.pypi.org/simple/ chpc-demo
Looking in indexes: https://test.pypi.org/simple/
Collecting chpc-demo
  Obtaining dependency information for chpc-demo from https://test-files.
     pythonhosted.org/packages/9c/c3/432
     e8df10a5db0e3d6b6963c41638f8b9e6b7b24d81fd759932e0dfa7d3e/chpc_demo-0.1.0-
     py3-none-any.whl.metadata
  Downloading https://test-files.pythonhosted.org/packages/9c/c3/432
     e8df10a5db0e3d6b6963c41638f8b9e6b7b24d81fd759932e0dfa7d3e/chpc_demo-0.1.0-
     py3-none-any.whl.metadata (664 bytes)
Requirement already satisfied: numpy<2.0,>=1.2 in ./software/pkg/anaconda3
   /2023.09/lib/python3.11/site-packages (from chpc-demo) (1.24.3)
Downloading https://test-files.pythonhosted.org/packages/9c/c3/432
   e8df10a5db0e3d6b6963c41638f8b9e6b7b24d81fd759932e0dfa7d3e/chpc_demo-0.1.0-py3
   -none-any.whl (3.4 kB)
Installing collected packages: chpc-demo
Successfully installed chpc-demo-0.1.0
```

From the installation output, it is obvious that the locally installed version of `numpy 1.24.3` satisfied the requirements in the `toml` file and therefore wasn't overwritten by a more recent version of the `numpy` package.

We are now ready to use the newly installed package:

```
sleipnir@x1:~$ python3
Python 3.11.5 (main, Sep 11 2023, 13:54:46) [GCC 11.2.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> from chpc_demo import extwiener
```

## References


[Eus18] Sébastien Eustace. Python packaging and dependency management made easy. https://python-poetry.org, 2018. Accessed: (10/27/2023).

[SEM23] Semantic versioning 2.0.0. https://semver.org/, 2023. Accessed: (11/01/2023).

[TOM23] `TOML`: Tom's Obvious Minimal Language. https://toml.io/en/, 2023. Accessed: (11/01/2023).