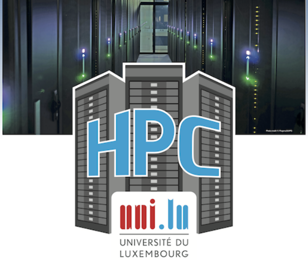


High Performance Computing & Big Data Services

hpc.uni.lu

hpc@uni.lu

@ULHPC

Uni.lu HPC School 2021

PS6: Python basics

Uni.lu High Performance Computing (HPC) Team

S. Peter

University of Luxembourg (UL), Luxembourg

http://hpc.uni.lu

Latest versions available on Github:

UL HPC tutorials:	https://github.com/ULHPC/tutorials
UL HPC School:	hpc.uni.lu/education/hpcschool
PS6 tutorial sources:	ulhpc-tutorials.rtfd.io/en/latest/python/basics/

Summary

1 **Introduction**

2 Using Python on the UL HPC clusters

3 Python packages

4 Virtual Environments

5 Conda

6 Example used in the tutorial

7 Packages used in the tutorial

Main Objectives of this Session

Basics

- Run Python code on the cluster.
- See the difference between Python **versions**.
- Install and use **Python packages**.
- Switch between different Python and package versions using a **virtual environment**.
- Speed up code using packages.
- Compile your code in **C** to have better performance.
- Create an independent Python installation with **conda**.

Summary

1 Introduction

2 **Using Python on the UL HPC clusters**

3 Python packages

4 Virtual Environments

5 Conda

6 Example used in the tutorial

7 Packages used in the tutorial

Python on the UL HPC Platform

You have two different ways of using Python on the UL HPC Platform:

- Use the **system** Python installed on the nodes
  - ↪ version **2.7** and **3** are installed under `/usr/bin/python` and `/usr/bin/python3`
- Rely on Environment Modules **once** inside a job on a computing node
  - ↪ then you can search for the avaiable versions of Python with `module avail lang/python`

Python on the UL HPC Platform

You have two different ways of using Python on the UL HPC Platform:

- Use the **system** Python installed on the nodes
  - ↪ version **2.7** and **3** are installed under `/usr/bin/python` and `/usr/bin/python3`
- Rely on Environment Modules **once** inside a job on a computing node
  - ↪ then you can search for the avaiable versions of Python with `module avail lang/python`

```
------------------ /opt/apps/resif/aion/2020b/epyc/modules/all ---------------------
   lang/Python/2.7.18-GCCcore-10.2.0    lang/Python/3.8.6-GCCcore-10.2.0 (D)
```

Python on the UL HPC Platform

- Make sure to always use the same Python version (and package versions) when running your code or workflow.
- The first thing you should always do is to load the version of Python that you need.
  - ↪ Your scripts will use the loaded module version to execute.
  - ↪ This version will be used inside your virtual environment.
- Python code is not necessarily compatible between versions 2 and 3.
- For many packages recent versions are only available for Python 3.

Examples of module usage

```
$(node)> module load lang/Python/3.8.6-GCCcore-10.2.0
$(node)> python --version
Python 3.8.6

$(node)> module load lang/Python/2.7.18-GCCcore-10.2.0
$(node)> python --version
Python 2.7.18

$(node)> module purge
$(node)> python --version
Python 2.7.18
```

Summary

1 Introduction

2 Using Python on the UL HPC clusters

3 **Python packages**

4 Virtual Environments

5 Conda

6 Example used in the tutorial

7 Packages used in the tutorial

pip: package installer for Python

```
$> python -m pip install --user <package>                    # install <package>
```

pip: package installer for Python

```
$> python -m pip install --user <package>                   # install <package>
```

```
$> python -m pip install --user -U <package>                # upgrade <package>
```

pip: package installer for Python

```
$> python -m pip install --user <package>                # install <package>
```

```
$> python -m pip install --user -U <package>             # upgrade <package>
```

Dump list of installed packages and their versions to a requirements file:

```
$> python -m pip freeze > requirements.txt
```

Summary

1. Introduction
2. Using Python on the UL HPC clusters
3. Python packages
4. **Virtual Environments**
5. Conda
6. Example used in the tutorial
7. Packages used in the tutorial

Virtualenv

The `virtualenv` package allows you to create an virtual environment for Python and isolate the installation of packages inside it.
You can have several virtual environments that are independent of each other and you can load them to use the packages installed inside.

Install `virtualenv` package

- For some versions of Python, you might have to **install** the `virtualenv` package locally in your home directory by using **pip**.

```
(node)$> python2 -m pip install --no-cache-dir --user virtualenv
(node)$> python2 -m virtualenv --version
virtualenv 20.10.0
```

- If you use a Python from the modules, it comes with `virtualenv` already installed.

```
(node)$> module load lang/Python/2.7.18-GCCcore-10.2.0
(node)$> python -s -m virtualenv --version
virtualenv 20.0.34
(node)$> module load lang/Python/3.8.6-GCCcore-10.2.0
(node)$> python -s -m virtualenv --version
virtualenv 20.0.34
```

Create virtual environment

- Create your own **virtual environment** to install packages in it:

```
(node)$> module load <your Python version>
(node)$> python -m virtualenv <name of your environment>
```

- If you do not load any module, the the **system** python will be used. It is a best practice to specify your Python version via module before running any script.
- Now you can **activate** the environment named `tutorial_env`, for example, with the following command:

```
(node)$> source tutorial_env/bin/activate
(tutorial_env)(node)$> # you are now inside your virtual environment
```

Python 3 virtual environments

Python **3** comes with a `venv` command already built-in, so you do not need to install the `virtualenv` package.

```
(node)$> module load lang/Python/3.8.6-GCCcore-10.2.0
(node)$> python3 -m venv tutorial_env
(node)$> source tutorial_env/bin/activate
(tutorial_env)(node)$> # you are now inside your virtual environment
```

Summary

1 Introduction

2 Using Python on the UL HPC clusters

3 Python packages

4 Virtual Environments

5 **Conda**

6 Example used in the tutorial

7 Packages used in the tutorial

Conda

- open source package and environment management system
- runs on Windows, macOS and Linux
- quickly installs, runs and updates packages and their dependencies
- easily creates, saves, loads and switches between environments on your local computer and the HPC cluster

Summary

1 Introduction

2 Using Python on the UL HPC clusters

3 Python packages

4 Virtual Environments

5 Conda

6 **Example used in the tutorial**

7 Packages used in the tutorial

Compute standard deviation

The naïve code used to compute the standard deviation of an array of random numbers (lst) is:

```
def mean(lst):
    return sum(lst) / len(lst)


def standard_deviation(lst):
    m = mean(lst)
    variance = sum([(value - m) ** 2 for value in lst])
    return math.sqrt(variance / len(lst))
```

The variable will be the size of the array on which we want to compute the standard deviation. The idea is to reduce the time used to compute this value by using libraries (numpy) or compile the code in C (with pythran).

Summary

1. Introduction
2. Using Python on the UL HPC clusters
3. Python packages
4. Virtual Environments
5. Conda
6. Example used in the tutorial
7. **Packages used in the tutorial**

NumPy

NumPy is the fundamental package for array computing with Python.
It provides:

- a powerful N-dimensional array object
- sophisticated (broadcasting) functions
- tools for integrating C/C++ and Fortran code
- useful linear algebra, Fourier transform, and random number capabilities
- and much more

Besides its obvious scientific uses, NumPy can also be used as an efficient multi-dimensional container of generic data. Arbitrary data-types can be defined. This allows NumPy to seamlessly and speedily integrate with a wide variety of databases.

Pythran

- Pythran is an ahead of time compiler for a subset of the Python language, with a focus on scientific computing.
- Compile your Python code in C++ for (hopefully) faster execution.

Thank you for your attention...

Questions?

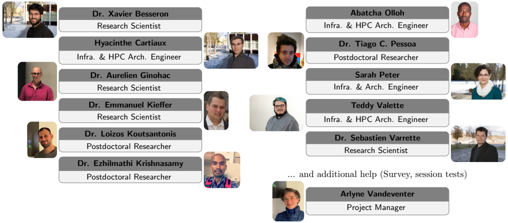
ulhpc-tutorials.rtfd.io/en/latest/python/basics/

High Performance Computing @ Uni.lu

University of Luxembourg, Belval Campus
Maison du Nombre, 4th floor
2, avenue de l'Université
L-4365 Esch-sur-Alzette
mail: hpc@uni.lu

1. Introduction
2. Using Python on the UL HPC clusters
3. Python packages
4. Virtual Environments
5. Conda
6. Example used in the tutorial
7. Packages used in the tutorial

Uni.lu HPC School 2021 Contributors



Dr. Xavier Besseron
Research Scientist

Hyacinthe Cartiaux
Infra. & HPC Arch. Engineer

Dr. Aurelien Ginohac
Research Scientist

Dr. Emmanuel Kieffer
Research Scientist

Dr. Loizos Koutsantonis
Postdoctoral Researcher

Dr. Ezhilmathi Krishnasamy
Postdoctoral Researcher

Abatcha Olloh
Infra. & HPC Arch. Engineer

Dr. Tiago C. Pessoa
Postdoctoral Researcher

Sarah Peter
Infra. & Arch. Engineer

Teddy Valette
Infra. & HPC Arch. Engineer

Dr. Sebastien Varrette
Research Scientist

... and additional help (Survey, session tests)

Arlyne Vandeventer
Project Manager

hpc.uni.lu